AffinityScanner

A tablet application to support the affinity diagramming process

Master Thesis
March 19, 2012

Lukas Keller
Zurich, Switzerland
Student-ID: 04-919-361
lukas.keller@access.uzh.ch

University of Zurich
Department of Informatics

People and Computing Lab
Professor: Elaine M. Huang, PhD, huang@ifi.uzh.ch
Supervisor: Gunnar Harboe, harboe@ifi.uzh.ch

Abstract

Affinity diagrams are a widely used technique to structure unstructured data. Data is segmented into pieces and the pieces are arranged as sticky notes on a wall. To support or substitute this traditional method, various electronic solutions have already been developed. But none of them has made a big impact in practice so far. This thesis follows a new approach, exploiting the possibilities of today's tablet computers. An application for tablet devices that displays additional information across the affinity diagram in an augmented reality style has been developed. Users can look at the affinity wall through the device and display context information, media data and annotations related to affinity notes in the diagram. The evaluation of the developed solution in several user tests has shown that the application is accepted by the users but still has some technical problems that negatively impact the handling. Nevertheless the application can help users to make better and more exact decisions in the affinity diagramming process based on the displayed context information.

Zusammenfassung

Affinitätsdiagramme sind eine häufig verwendete Technik um unstrukturierte Daten zu strukturieren. Die Daten werden dazu in Teile zerlegt, diese Teile auf Haftnotizzettel geschrieben und an eine Wand geklebt. Als Ersatz oder Unterstützung dieser Methode wurden bereits diverse elektronische Lösungen entwickelt. Aber keine dieser Lösungen hatte bis jetzt Erfolg in der Praxis. Die vorliegende Arbeit verfolgt einen neuen Ansatz, welcher die Möglichkeiten heutiger Tablet Computer ausschöpfen soll. Eine Tablet-Applikation, welche im „Augmented Reality“ Stil zusätzliche Informationen zum Affinitätsdiagramm anzeigt wurde entwickelt. Mit der entwickelten Applikation ist es dem Nutzer möglich, das Affinitätsdiagramm an der Wand durch die Kamera des Tablets zu betrachten und dabei den einzelnen Notizzetteln zugehörige Kontextinformationen, Mediendaten und Notizen anzuschauen. Die Evaluation in mehreren Benutzertests hat gezeigt, dass die entwickelte Applikation trotz einiger Probleme von den Nutzern als hilfreich eingeschätzt wurde und dazu führen kann, dass während der Erstellung des Affinitätsdiagrammes bessere und genauere Entscheidungen getroffen werden.

Acknowledgments

I would like to thank the following persons for their help in writing this master thesis:

Professor Elaine Huang for giving me the opportunity to write this thesis, my supervisor Gunnar Harboe for the time and effort he put into assisting my work and my user studies and for the great amount of helpful hints he gave me during the whole process, the ZPAC team for the nice atmosphere in the lab, Erich Reinhardt for his lecture service, my girlfriend Karin Reinhardt for her love and patience, Gelek Doksam who worked on his master thesis at the same time and gave me valuable inputs and my sister Christine.

Contents

Figures

1 Introduction

Affinity Diagrams are a way to create a hierarchically organized structure from unstructured data. The whole data is segmented into small pieces, written down on paper notes and stuck to a wall. These notes are then grouped into groups of similar notes and a hierarchy of similar groups is built. During this process, the data gets structured and the people working with it get deeper insight into the data set.

Traditionally, those notes are written on sticky paper notes and put on the wall. This approach however has some disadvantages. Even mid-sized diagrams take a lot of space on a wall. As the process often takes more than one day, the wall cannot be used otherwise during the diagramming phase. If the data is confidential (e.g. a user study), the whole room can only be used by the people who have permission to see the data. The fact that the data is stuck to a physical wall makes it nearly impossible to work together with people who are not in the same room. This can be problematic for teams operating apart. Once the diagram is finished, the question is what to do with it. A good idea is to archive it by taking a photo of the wall or stick the notes on multiple “butcher papers” (a long sheet of paper, format A2), so that they can be archived elsewhere. Both approaches have their drawbacks in case the diagram needs to be reused later. A digital photo is seldom watched in wall-size and the original view can therefore hardly be reconstructed again and to put the butcher papers back on the wall later requires some effort (Curtis, Heiserman, Jobusch, Notess, & Webb, 1999; Harboe, Minke, Ilea, & Huang, 2012; Judge, Pyla, McCrickard, & Harrison, 2008).
These drawbacks have led to solutions that replace the paper and pen with digital representations. A digital affinity diagram can address a lot of the problems of a paper-based approach, as digital representations are easily moved around in teams operating in different places and can be taken off a wall if not used and redisplayed when used again. But even though several digital solutions exist, the traditional paper and pen approach still seems to be favored by users (Harboe, Minke, et al., 2012; Judge et al., 2008). Reasons discussed in literature are that digital solutions negatively influence the interaction and collaboration in the group and that the tangibility and the spatial awareness of a real wall with real paper notes gets lost (Harboe, Doksam, Keller, & Huang, 2012).

A new approach followed in Harboe, Doksam, et al. (2012) is to support the affinity diagramming process by electronic solutions, without actually forcing the user to use it. The user should be able to move seamlessly between the traditional affinity diagram and the augmented system.

The necessary hardware should be available in the mass market and therefore not present a barrier in terms of costs. The approach currently focuses on the use of mobile devices and beamers to overlay the affinity diagram with more information and to enable the search for affinity notes. This master thesis focuses on one part of this approach. A tablet device is used to give the user more context information for notes in an augmented reality style. The tablet can be pointed to the wall and the user sees the wall directly on the display. The paper notes are shown as digital duplicates one-to-one in the display as they are stuck to the wall, but they are clickable and can show context information to the note data like photos, annotations or different language versions of the text (see tablet device in the foreground of Figure 1). The user develops the affinity diagram in the traditional paper and wall style, but can at anytime access an additional digital layer of the diagram by using the tablet app.

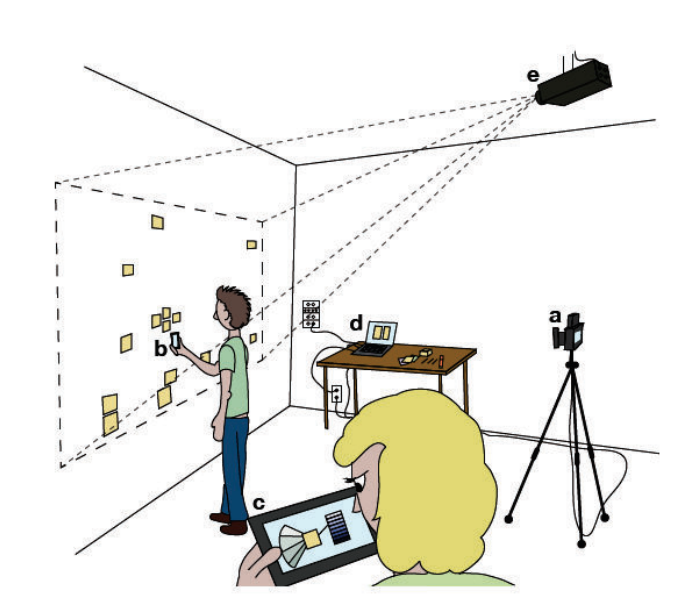


Figure 1: Augmented affinity diagram with beamer and mobile clients (taken from Harboe, Doksam, et al. (2012))

The objective with the tablet client is on the one hand to give the researchers more context information about the interview and about specific notes, on the other hand to let them enrich the diagram with additional information like annotations or attached photos on notes. Both types of information (context and enriched) can be used either during the diagramming process to enrich the process or by researchers that join the process later and first have to get an understanding of the diagram and the events afore.

The tablet application was developed with a strong focus on usability. Therefore a lot of effort has been put into designing, prototyping and testing the user interface to make it easy and efficient to use. This should seamlessly integrate the solution into the traditional affinity diagramming process, without giving the user the feeling of switching from the physical paper-wall world to the digital world.

Development started by designing the user interface of the tablet application with sketches and storyboards. Then, prototypes have been built to test multiple design ideas with users. Later, the final design has been implemented as an Android tablet application. Additionally a server has been developed that communicates with the tablet application and holds the data the tablet application displays to the user. At the end the whole solution has then been evaluated in another user test.

This thesis first portrays solutions already existing and gives an overview on the research that has been done with electronic representations of affinity diagrams. This is followed by the description of the design process which allowed to gather requirements for the tablet application and to figure out the best interface design. This is followed by parts including the technical implementation, the functionality and the evaluation of the application prototype. The thesis will finish by summarizing the findings of the evaluation.

2 Related Work

This section first describes the affinity diagramming process in more detail. In a second part, solutions to support or substitute affinity diagrams with digital means are presented. The most important of them are summarized and their key characteristics are described. Then, a quick summary of the Android mobile operating system that was used in the tablet is given. At the end, the previous work that led to this thesis is presented.

2.1 Affinity Diagram

As already mentioned in the Introduction, affinity diagrams are a way to create a hierarchically organized structure from unstructured data. A bottom up approach is used on the data. It is done by segmenting the whole data (normally in text form) into small pieces, write them down on paper notes and stick them to a wall. Notes on the wall are then grouped into groups of similar notes. Later, a hierarchy of similar groups can be built to build a tree-like group structure. The affinity diagramming process ends when most of the groups are connected to only a few high level groups. While building this hierarchy, the data gets structured and the people working with it get insight into and overview over the whole data set (Beyer & Holtzblatt, 1997, 1999).

Figure 2 shows an affinity diagram. The yellow notes are the notes with the data on it. The violet notes are group labels. The yellow notes under the violet group labels belong to the same group. At the top of the image bigger white notes can be seen. These are higher level group labels that summarize the violet groups below them. The bluish notes are annotations to the yellow notes.

Figure 2: Affinity diagram (taken from http://wiki.fluidproject.org/display/fluid/Affinity+Diagrams)

Affinity diagrams are often used to gain insight into data from unstructured interviews. Those interviews are often made in user studies to research the users' needs and to gain insight into the work processes of users. Typically users have different needs and different work processes and data is often dissimilar. Segmenting the interviews on paper notes and arranging them on a wall allows for multiple persons to sort together in a collaborative way similar needs or processes and to make groups out of them which abstract the individual needs and processes into higher level requirements (Holtzblatt, Wendell, & Wood, 2005).

There exist also different methods for creating affinity diagrams than the one used in this thesis. Apart from variations of wall affinity diagrams, there exist also versions where the affinity notes are laid out and grouped on a table. As the affinity diagram is easy to understand, most users modify the method a little bit to fit their needs.

2.2 Digital Solutions

As presented in the Introduction, the traditional affinity diagram with paper notes stuck on a wall has disadvantages. This led to the development of solutions that enhance the affinity

diagram by electronic means. Most of the solutions replace the traditional approach with an entirely electronic application. They simulate the arranging and structuring of the notes on a computer screen instead of on a wall. Some of them try to combine the traditional method with electronic devices. This normally means that the input of the data is still made by hand-written notes, but these notes are then converted into digital representations and the affinity diagram is made with the digital duplicates of the notes.

The advantage of most of the electronic solutions is that the diagram is electronic and therefore can easily be passed between different sites, can be archived and reused and also be looked at in different kind of views. Also the costs for printing the sticky notes can be reduced.

In the following part, some of the existing solutions are presented in a non-exhaustive manner. They are categorized into desktop and touch screen solutions.

2.2.1 Desktop Solutions

Desktop solutions are applications that run on normal desktop computers without touch screen displays. The normal way to interact with these applications is with the mouse and the keyboard. The main disadvantage of these solutions is that the affinity space is limited to the screen size of the desktop application. Another problem is that the haptic feedback and the spatial awareness of real notes is lost, because the manipulation with the mouse is very far from the traditional way the notes are arranged (Judge et al., 2008; Klemmer, Newman, Farrell, Bilezikjian, & Landay, 2001).

There exist a lot of desktop solutions. The presentation here is only a small excerpt of all the software made for supporting the affinity diagram method.

2.2.1.1 StickySorter

StickySorter[1] is an Office Lab Project of two Microsoft employees to support virtual collaboration in affinity diagrams in teams operating apart. The software is a fully implemented affinity diagramming system with simulated sticky notes that can be moved around with the computer mouse. Notes can be grouped, but no higher hierarchies can be constructed.

Convenient in this application is the possibility to read in the data from CSV files. After the diagram is built, the hierarchy can also be saved to CSV files again to preserve the diagram.

[1] http://blogs.office.com/b/crabby_office_lady/archive/2010/08/18/visions-amp-concepts-stickysorter.aspx

Besides the previously mentioned problems of desktop solutions, this solution has the drawback that only one level of groups can be built. A real structure of the data cannot be created in this way.

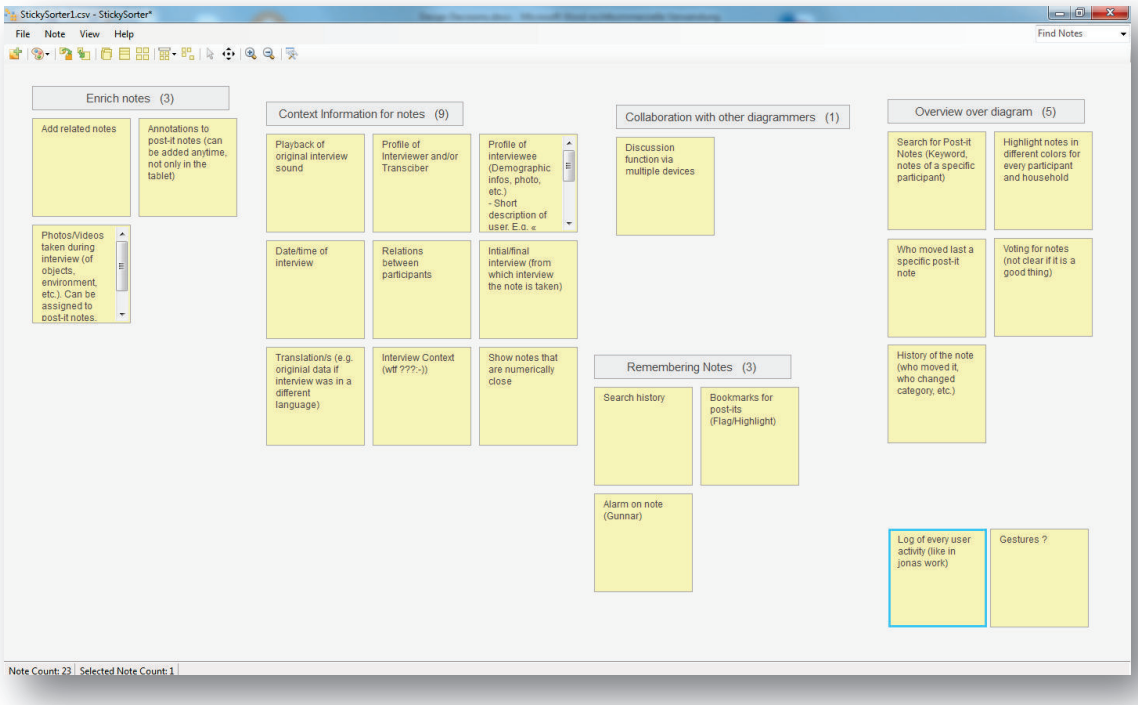


Figure 3: Screenshot of the StickySorter application with the affinity diagram used in the design process

2.2.1.2 CDtools

CDTools[2] from InContext is a commercial product. It supports the user with the creation of the sticky notes and can also be used to archive the affinity diagram to capture its hierarchy. The affinity diagram is nevertheless done by paper notes on a wall (Minke, 2011). This tool is currently discontinued and there is no more information about the product displayed on the web.

2.2.2 Touch Screen Solutions

Touch screen solutions let the user interact with the notes in a more natural way than desktop solutions. Users can interact with the notes directly on the screen and move them around similarly to moving around real notes on a wall. These solutions try to simulate the same feeling of notes on a wall that users have in the traditional approach. This combines the advantage of an electronic representation of the sticky notes with the possibility to collaborate on a shared space

[2] http://incontextdesign.com/process/cdtools/

(the wall or table). While a lot of the problems of desktop solutions are solved, other problems are not. The haptic feeling of real notes still cannot be simulated and the usability depends strongly on the quality of the implementation. In addition such systems are rather expensive because they use big displays and very new technology that is not available in the mass market (Harboe, Doksam, et al., 2012).

In the following, three touch screen solutions are presented.

2.2.2.1 Using Multiple Display Environments for Affinity Diagramming

This solution is a system of both personal and shared devices with PDA's for every researcher and a big shared display for the whole group. Researchers can work on their PDA's, but can also work together on the whole or on parts of the affinity diagram on the shared display (Judge et al., 2008).

2.2.2.2 The Designers' Outpost

The designers' outpost is a solution targeted to website designers with a big wall-sized display that detects traditional sticky notes with a camera that have been stuck onto the wall. Users can create links between notes or transform a paper note into an electronic representation of the note, so that the original note can be removed from the display. A projector behind the display projects the information for these interactions with the system. The system aims to make it easier to shift from paper notes to a digital representation, to enable distributed teams to work on the diagram and to enable the use of the diagram in other use cases where a digital representation is better suited (Klemmer et al., 2001).

2.2.2.3 AffinityTable

AffinityTable is a touch screen based solution developed at the University of Konstanz. The solution consists of a horizontal touch screen to manipulate the affinity diagram and a vertical high resolution display to discuss the affinity diagram together. The solution tries to implement the natural affinity diagramming techniques in a digital solution. The data is written on special paper with special pens that scan the written characters and send them to the system. The paper notes can then be placed on the horizontal touch screen and are recognized with the written text. An electronic duplicate of the note on the screen is then displayed and can be used for further manipulation (Geyer, Pfeil, Budzinski, Höchtl, & Reiterer, 2011).

2.2.3 Drawbacks of the Existing Solutions

While electronic-only solutions have numerous advantages, such as better archivability, less room and wall space requirements and better collaboration between geographically distributed

teams (Judge et al., 2008), the paper-pen-wall solution is still used more frequently in real-world scenarios (Harboe, Minke, et al., 2012; Judge et al., 2008). By fully switching the process to an electronic-only solution, users loose the advantages of the traditional approach, such as the better support for group interaction and collaboration, tangibility, spatial awareness and interaction richness (Harboe, Doksam, et al., 2012).

In this thesis a new approach is followed to use electronic devices only as support for the traditional paper based approach. Therefore the affinity diagram is still done with paper and a wall, but the user can use supporting electronic devices whenever he feels like using them.

2.3 Android

The tablet application developed in this thesis is built for the Android operating system. Android is an open source operating system that is primarily built for mobile devices like smart phones and tablets (“Android,” 2012). The operating system is developed and maintained by the Open Handset Alliance. The Open Handset Alliance is a group of 84 technology and mobile companies that came together to bring new innovations into the market of mobile devices and applications (“Open Handset Alliance,” 2012).

2.4 Scientific Context

This thesis is the continuation of previous work and is part of a bigger research project of a system to support the affinity diagramming, conducted at the University of Zurich. Minke (2011) describes an early prototype system of this project to support the affinity diagramming process with a search functionality for text on affinity notes. The solution is implemented on a tablet computer and highlights notes containing the search term in the display when the camera is moved over the affinity wall. This work led to the design of further concepts to support the affinity diagram, presented in Harboe, Doksam, et al. (2012). The affinity diagram should be augmented by a system with a fixed camera to capture the whole affinity wall, a beamer to overlay the affinity wall with color information and mobile devices, such as tablets and smartphones, to control the system and display additional information.

The proposed system is split up in two solutions. Doksam (2012) describes the design and implementation of the solution with the beamer and the fixed camera. The camera captures the whole affinity wall and reads in the positions of the affinity notes. The user can search with a smartphone client for text in affinity notes. Notes containing the search term are then highlighted on the wall by the beamer.

This thesis describes the design and implementation of the other part with the tablet application. While the two systems do not have a connection in the current implementation, they could work together in future versions. This would enable the tablet application to make use of information from the fixed camera and the now independent smartphone and tablet clients may be merged into one client.

The described bigger project already defined a certain frame for this thesis. One part of this frame is that the solution should be for affinity diagrams made on walls and the affinity notes should be made recognizable by QR codes. Another part is the initial idea to use a tablet application that can display context information for affinity notes or has other helpful functions that support the user doing the affinity diagram. This idea seemed to be an interesting approach to investigate. It has been evaluated and adapted before implementing the system, but has been followed in general as originally defined. Development of the system therefore started based on these ideas and respected, as far as reasonable, the given frame.

3 Design

As the tablet application should integrate as seamlessly as possible into the existing traditional process of working with affinity diagrams, the use of the application should be as easy as possible. Therefore a lot of effort has been put into the design of the user interface and the user interactions.

The design process consisted of three phases. In the first phase, the requirements for the tablet app were defined and possible use cases have been drawn in sketches and storyboards. In the second phase, interface designs have been developed based on the sketches and storyboards. For every design a paper prototype has been built and tested with test users. In the third phase, one of the designs has been implemented as a working Android prototype. The Android prototype was fully working on the tablet, but had no connection to the server. This prototype was also tested with test users to verify the planned design. The design was then again modified for the final implementation of the application, called AffinityScanner.

This section covers only the design of the tablet application as the rest of the solution was not designed in this way. Nevertheless, the whole system has a server, a database and the tablet app as client. The whole system is presented in more detail in the Implementation section. In this section the process to design the user interface of the client is in focus.

3.1 Design Process

The design process consisted of collecting requirements for the application from literature and team discussions, creating multiple designs that implement these requirements, testing and improving them with paper prototypes and user tests and creating a final design out of the gained insights and implementing this design in a real working prototype.

3.1.1 Methods

The following methods were used for developing interface designs and reviewing them.

3.1.1.1 Team brainstorming

Team brainstorming sessions were held during research team meetings. Everybody expressed ideas they had about possible functions of the app or the interface design and one person took notes.

3.1.1.2 Storyboarding and design critique session

Storyboarding is a technique described in the book Contextual Design (Beyer & Holtzblatt, 1997). Possible use cases of an application are drawn as short stories. In our case, use cases that occur during affinity diagramming, including possible interactions with the tablet app, were drawn as storyboards. These storyboards where reviewed in meetings of the research team. Possible problems and issues were discussed and new ideas captured.

3.1.2 Idea Finding

The idea finding was the first phase of the design process. The objective was to find possible ways how the application could fit into the affinity diagramming process and to find ideas what the user interface and the user interactions with the tablet application would look like.

3.1.2.1 Requirements finding

Requirements are definitions of what the final application should be able to do. For example which information the interface should show to the user. To collect possible requirements, interviews conducted in previous studies of the research group (see Minke (2011)) and reports and papers have been analyzed. Additionally the research group, whose members use affinity diagrams in their research, sat together multiple times and brainstormed on possible information and interaction features that would be useful to have at hand via a tablet device during an affinity diagram.

3.1.2.2 Conclusions of the requirements finding

The collected requirements have been reduced in another team discussion to a list of final requirements that are useful and also technically feasible with the planned equipment and infrastructure. The points on the list have been categorized (with the help of the affinity diagram in Figure 3) into the 5 following groups:

- **Enrich notes**: e.g. attach images/videos, attach annotations, attach related notes
- **Context information**: e.g. profile of participant (maps for hometown, etc.), details of affinity note (interview date, interviewer, etc.), play original interview sound
- **Remembering notes**: e.g. bookmarks, alarms on notes
- **Collaboration with other diagrammers**: e.g. chat to discuss with diagrammers not present
- **Overview over diagram**: e.g. browse affinity notes on tablet

These functions have been considered when developing the prototype designs. Some have been limited or even dropped entirely during the design process. Section Final Prototype Features explains which functions have been implemented in the end.

3.1.2.3 Design finding

Out of the findings from the research in the previous point, sketches and storyboards of possible interactions and work flows with the system has been drawn (see Figure 4). These drawings have again been discussed in a design critique session in the research team and positive and negative aspects have been made note of.

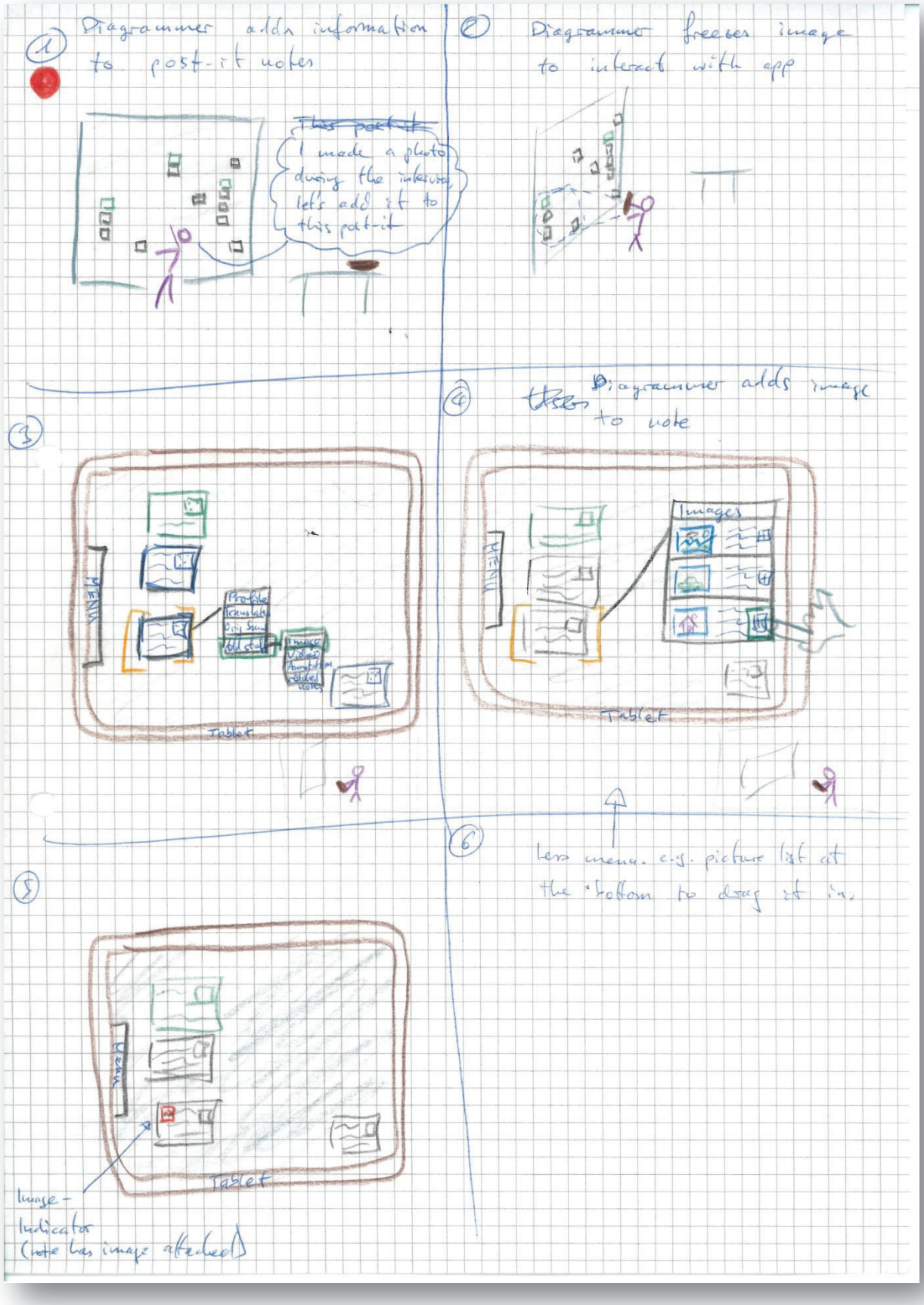


Figure 4: Early storyboard of the interactions and the interface of the tablet app

3.1.3 Final Prototype Features

During the design process of the prototypes, a lot of the previously defined features have been dropped. Reasons were that the design process and user tests revealed that they are hard to implement or that they are not useful for the user. In the following list are the functions that have survived and are implemented in all the prototype designs (paper and Android prototypes). From the initially defined features only the categories “enrich notes” and “context information” have survived the design finding process. At some of the features, the functionality has been limited during the design process, but the basic function is still implemented.

- **Annotating affinity notes:** The user can attach annotations in digital form to every affinity note. This should help storing thoughts that the user has in mind about a note and also to communicate ideas or remarks to other users that are working on the same affinity diagram.
- **Attaching media files:** The user can attach media files to affinity notes. Often pictures or videos are taken during interview sessions. During an affinity diagram, these pictures can be attached to affinity notes where a user thinks they help explain the affinity note. This is a way to communicate to the other users contextual information about a statement on an affinity note.
- **Information about data source:** For every note information about the data source is displayed. This can be information about the interview partner and the location of the interview if the data source is an interview.
- **Different language versions:** When interviews are held in different languages, the text on the affinity notes often needs to be translated into other languages to make it comparable or to enable all users to understand it. Sometimes the translation is unclear and consulting the original text can help.
- **Global media pool:** Media such as photos and videos that was recorded during data collection or that is part of the data should be accessible during the affinity diagramming process.

3.1.4 Paper Prototypes

After the idea finding was finished, designs for the user interface have been developed, based on the previous sketches and storyboards and reflecting the insights gained from the design critique and the brainstorming. This was a creative process which took up multiple days per design. Every design has been “implemented” with paper. All possible screens for a given set of tasks have been drawn. Some screen elements like popup windows have been made (re-)movable, so not

the whole screen had to be exchanged for small interface changes. The drawings were then stuck onto a real Android tablet, to simulate the feeling and the weight of the prototype.

After describing the conducting of the user tests, this part explains the functionality, the design decisions and the user test results for each prototype. Design decisions are only described in short paragraphs. The full details of the decisions are appended in section Appendix.

3.1.4.1 User tests

User tests of around one hour were made with all prototype designs to evaluate how users work with them and find ideas to improve them. The test users had not seen the designs before, so they were all unfamiliar with them. Most of the test users were also unfamiliar with the project and with the affinity diagram technique, so only the usability of the user interface was tested, but not how the application fits into the process of working with affinity diagrams.

The user had to execute 6 tasks which were written down on a sheet of paper. The covered tasks were unveiled after the user had finished the previous task. The tasks of the user tests are appended in section Appendix. They cover all the basic functions the prototypes had, so that each function was tested by the users.

For the user test scenario, only two paper notes were used with which the user interacted. As the prototypes were made of paper, a researcher played the role of the computer system to react to user actions. The test user sat in front of the paper prototype, the researcher right next to him. When the user made an action like clicking on the (paper) screen, the researcher either changed something on the screen or said what would happen in response to the users action (in case the user action did not cause a reaction on the screen).

The test was recorded with a voice recorder and notes of interesting or problematic incidents were taken. The tests were conducted similar to the paper prototype interview technique described in the book Rapid Contextual Design (Holtzblatt et al., 2005), but the prototype was not modified during the tests as suggested there.

To help understand how the user views the system, the Think Aloud technique was used during the tests. This means that test users had to speak out loud what they are thinking while executing the tasks (Nielsen, 1994). This technique made it easier to identify major problems of the user interface.

3.1.4.2 First design

The first design idea was inspired by the interaction categories defined in subsection Conclusions of the requirements finding. But only the categories “enrich notes”, “context information” and “remembering notes” have been implemented.

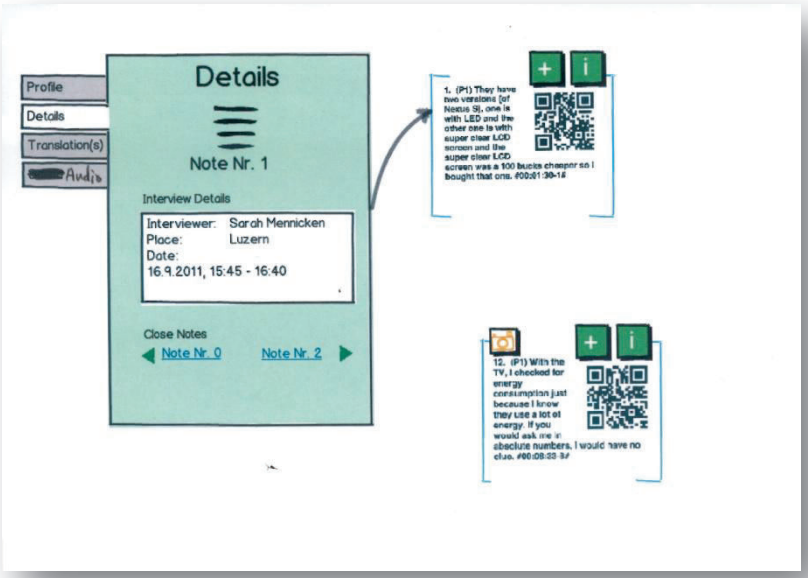


Figure 5: Display of the interview details of an affinity note

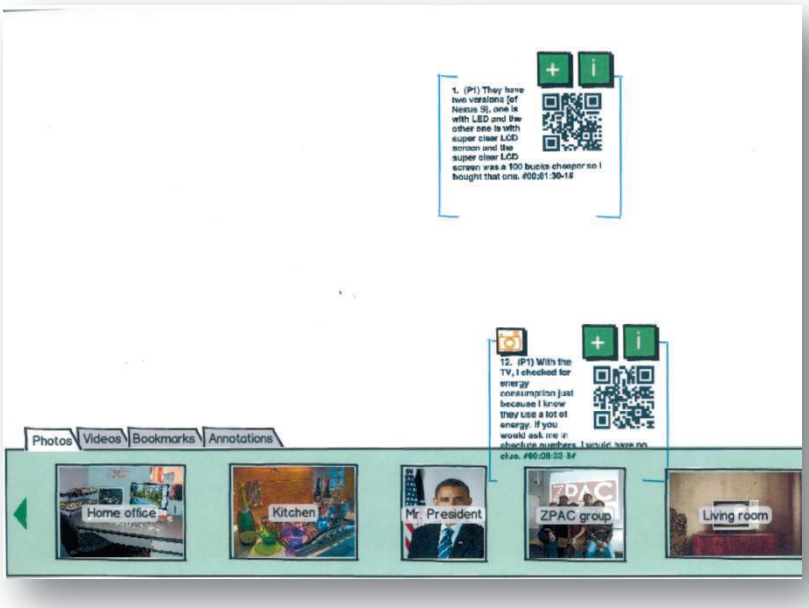


Figure 6: Browsing photos that have been made during the interviews the affinity diagram is based on

3.1.4.2.1 Functionality

When a note was recognized, the user had two buttons to interact with the note. An add button (with a plus sign) and an information button (with an information sign). With the information button, a window opened where the user could see different information about the note (see Figure 5). With the add button, a slider appeared at the bottom of the screen where the user could attach different items (photos, videos or annotations) to the note (see Figure 6). To bookmark a note, the user had to drag a bookmark icon from the slider to the note. If a note had something attached (e.g. a photo), a corresponding button appeared on the note (e.g. a photo button).In the case of the photo button, a photo viewer came up when the button was clicked. This was a simple window where the user could go through all attached photos.

3.1.4.2.2 Design decisions

In addition to the interaction categories, the design was based on the following findings from the previous idea finding process:

- **Menu depth**: Deep menu structures were omitted wherever possible. The solution is a concentration of two interaction categories “enrich notes” and “context information” that are directly tied to the note. They are represented in the UI with an ADD-button (+) and an INFO-button (i).
- **Constraining the interaction possibilities:** To make the interface easier to understand, the interaction possibilities are constrained. Only an add button and an information button can be used to interact with the note on the display.
- **Design Principles: T**he design principles **“**Consistency” and “Generalizability” (taken from the book Human-Computer Interaction (Dix, Finlay, Abowd, & Beale, 2003)) has been used as background when developing the user interface. The interaction in one interaction category (“enrich notes” and “context information”) is always the same.
- **Freeze Mode:** As the tablet is quite heavy, it is not very convenient to interact with it while pointing to the wall. So a freeze mode was introduced, where one can freeze the actual image of the wall and then lower the tablet to interact with the app.

3.1.4.2.3 User test results

The user test was conducted with 2 test users. During the tests, the following major problems were encountered and new ideas were found:

- **Global add button vs. local add button:** Users were confused when they clicked the add button on a note, because they did not know whether the displayed items in the item slider (e.g. photos) were only items attached to the clicked note (local) or all items in the database (global). A user suggested adding a global add button where you can access all the items that are in the database and local add buttons on the notes that show only the attached items.
- **Design changes:** During one test, the test user came up with the idea of a new interaction concept, where the buttons on the note are removed and the note as a whole is clickable. Instead of the buttons a menu would appear when the note is clicked with all functions accessible on the uppermost level. Annotations and bookmarks should be addable directly on the note.
- **General test outcome:** In general, the two test users had few major problems with the design, so the design appeared to be good. Nevertheless, the shortcomings were addressed in the second design.

3.1.4.2.4 Conclusion

Users in general had no major problems with the interface in the user test. But they had problems with the global and local add button. The second design should therefore not result in totally redesigning the interface, but address the criticized parts. The design idea of a test user will also be respected in the new second design.

3.1.4.3 Second design

The second prototype was an iteration of the first design and tried to resolve the problems found in the user tests with the first design and in addition included new ideas. The visibility of available functions has been tried to improve, e.g. the user was now able to see with an icon in the upper right corner if a translation of the text was available.

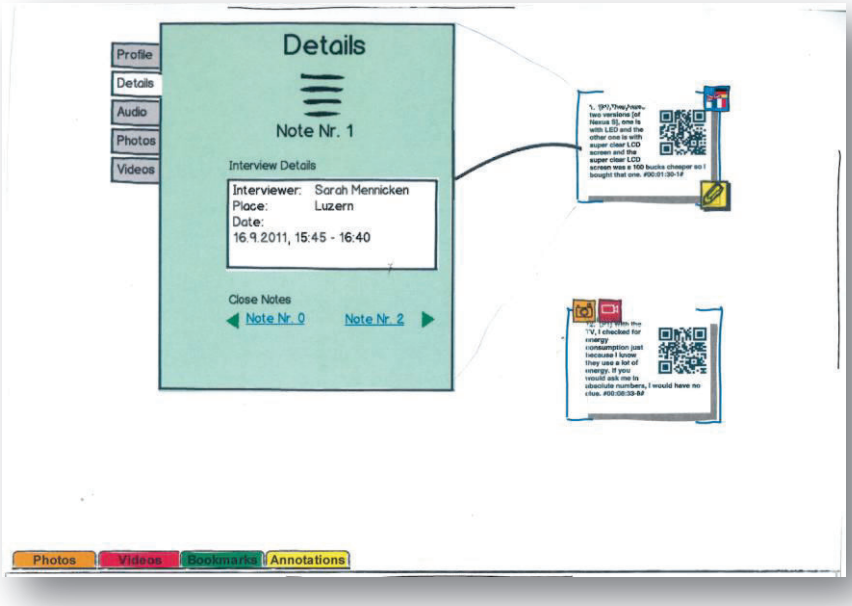


Figure 7: Interface after click on the affinity note

Figure 8: Item slider is activated and shows all photos in the database

3.1.4.3.1 Functionality

When an affinity note was recognized, buttons on the note were displayed, depending on information that was available or elements that were attached to the note (see Figure 7). If the note had photos attached, the photo button was displayed. If the note had a translated version, a translation button was displayed. These buttons were made to improve the visibility of information that was not present on every note. The reason they were made as buttons and not

just icons was to allow a direct access to these functions. The normal information about the note was accessed with a click on the note. This opened an information window similar to the first design.

At the bottom of the screen, the tabs of an item slider were visible. When they were tabbed, the slider moved up (see Figure 8). The user could drag and drop the items in the slider to the note, like in the first design.

3.1.4.3.2 Design decisions

In this design, insights from the previous user test were respected.

- **Making the note clickable:** Test users from the test run with the first prototype suggested making the whole note clickable. So the note was made clickable and a shadow was added to the note so that the user sees that it is clickable.
- **Menu:** The menu has been changed to a menu tree that opens when the note is clicked. This should make a less desktop-like user interface.
- **Item pool:** The item slider has been added statically to the Interface, but collapsed at the bottom of the screen. The content is now only global content that is not related to a note. It shows all bookmarks of the app and all annotations that are stored in the database.
- **Local/global:** A big issue with the first design was that the users had problems with content that is available to all notes like photos and movies in the database (global) and content that is related to specific notes like bookmarks and annotations of a note (local). This has been addressed by a global item pool at the bottom of the screen and a menu on the note that has only local functions and icons on the note.

3.1.4.3.3 User test results

The user test was conducted with one test user. During the test, it became obvious that the addressed issues have not been solved very well and further tests with this design have been canceled:

- **Global/local Problem:** The user expected the content of the item slider to be local to the note (photos that are attached to the activated note). Later he recognized that it is global and had no problem to change his behavior.
- **Data pane focused:** The user was totally focused on the data pane (the information window of the note). He only interacted with this window and did not interact with the note, nor with the symbols and buttons on the note.

- **Buttons not recognized:** Most of the buttons had not been recognized as buttons and were not used therefore.
- **Comparison of two language versions:** The user had the desire to have the two different language versions of the note side by side, so that he could compare the differences.
- **Translation button as language settings:** The button to show the translation of the note text was thought to be the language settings of the application by the user.
- **Drag and drop anywhere:** To attach elements to a note, drag and drop should also be possible on the data pane of the note, not only the note itself.

3.1.4.3.4 Conclusion

The changes to the design had made it worse. The interface was much more complicated than in the first design. The test user did not get the thoughts behind the interface system because it was not clear enough. The idea was to make the interface simpler by making the note clickable. But additional buttons on the note made it more complicated. The next design was a totally different approach from the previous ones, trying to get rid of the different issues of the previous approaches.

3.1.4.4 Third design

The third prototype was a total remake and was inspired by the idea to have a non-desktop computer feeling, i.e. exploiting the possibilities of the tablet (touch screen, gestures). Also the idea of previous test users of displaying the translated text at the same time as the original text to compare the versions has been taken into account. Following the same idea, all information was made visible at the same time, so that the user does not have to click through the information, but can see it on the display all the time he interacts with an affinity note. Multiple design ideas have been brainstormed and one has been chosen where the information is on cards that look similar to the note, but have different colors and are arranged in a stack behind the note.

Figure 9: Interface with two affinity notes. One note has the information cards expanded.

Figure 10: Interface with expanded information cards and photos displayed that are attached to the note

3.1.4.4.1 Functionality

When the user clicks on a recognized note, information cards are laid out around the note (see Figure 9). Information cards are elements on the screen that have the same shape as the affinity note (a rectangle), but contain information about the note instead of the note text. They are called cards because they can be stacked behind the note to use less space. They replace the information window of the previous designs. The user can move them around with the finger to

arrange them in different positions. There is an information card for details about the note (Detail Card), details about the interviewed person (Portrait Card), translated or original versions of the note text (Original Text Card), annotations on the note (Annotation Card) and media files attached to the note (Media Card). The Annotation Card is just a text field where the user can directly write text. The Media Card has two buttons that look like stacks of photos, one is for photos and the other is for videos. If the user clicks on the button, the attached photos or videos are put on the screen in random positions.

The media items like photos are no more displayed in a photo viewer, but they are laid out over the whole screen, exploiting the unused space of the display (see Figure 10). To access all media items that are in the database, the user could open the media gallery in the lower left corner. To bookmark a note, a button in the shape of a star was displayed in the QR code field of the note.

3.1.4.4.2 Design decisions

The central ideas behind the new interaction concept were the following.

- **Exploit the whole display, not only small parts of it to display information:**
  All information can be checked at the same time, because information cards and media items can be moved around over the whole display, allowing it to place them in parts of the display where no other elements are covered.
- **Make a new non-desktop computer interaction feeling with the app for the user, moving away from the classic click-and-display scheme of a desktop application:**
  In the last designs, the interactions were pretty similar to desktop applications. In the new design, everything can be moved around on the screen with a finger and photos can be zoomed in by pinch gestures.
- **Make multiple information types visible at the same time to facilitate comparison:**
  The new interface displays all information at the same time. Unnecessary information can be moved away from necessary information in order not to distract.
- **Reduce the needed clicks to display information:**
  Only one click on the note displays all available information.
- **Make a clear distinction between local content and global content:**
  To support the clear distinction, the new interface has now two similar media pools called galleries, one local for the note as an information card and one global for the whole affinity diagram in the lower left corner of the interface.

3.1.4.4.3 User test result

The user tests were conducted with 3 test users. The users did not have major problems using the interface and the feedback was very promising. In general, this prototype has been accepted by all users with very few problems. Most of the issues were about details of the design. They are mentioned in a short paragraph, outlining the points that were addressed in the final design.

- **Close button on the note:** Users would have preferred to have a stack button on the note to put all information cards back behind the affinity note.
- **Trash can:** To make photos and bookmarks removable from notes, a trash can has been proposed where the user can put in the element he wants to remove.
- **Info cards scalable:** To make content better readable, a user proposed to make the information cards scalable with a pinch gesture.
- **Information card titles not visible:** A special layout algorithm should be used to lay out the information cards around the note. Otherwise the titles of some cards are covered by the note and the user does not see what the content should be.

Figure 11: User test with the prototype of the 3rd design

3.1.4.4.4 Conclusion

As this design was accepted by the users with very few problems, it has been taken to implement the first Android prototype with only minor changes.

3.1.5 Android Prototypes

As the third design was accepted very well by the test users, this design has been taken to implement on a real tablet device. To test the real world implementation of the design before developing the final solution, a first implementation was made that runs only locally on the tablet and that has no connection to the server where the data is stored. To simulate the server connection, the data was mocked up on the device.

3.1.5.1 First implementation of final design

The first implementation was the design from the 3rd paper prototype with minor changes implemented as a fully functioning application, but without any server connection. Any displayed content was mocked up locally on the device in the first user test of the Android application. But the interface was already fully working, so it was testable if the planned design from the paper prototypes is feasible on a real touch display with real interaction.

3.1.5.1.1 Functionality

The functionality was the same as in the third paper prototype.

3.1.5.1.2 User test results

The user test was conducted with 4 users. The prototype had some known problems like reliably recognizing the notes on the wall and difficulties with the auto-exposure. These problems affected the test run quite a lot. Nevertheless, interesting and important observations and feedback from the user could be obtained. After the user test, the prototype and the findings from the test were discussed in a design critique session in the research team.

In the user tests and the design critique session the following major issues have been found and defined.

- **Keyboard comes up unintentionally: T**he annotation card was designed as a text input box. Unfortunately, users often clicked into the input box unintentionally and thereby triggered the soft keyboard to come up on the screen awaiting input.
- **Media Gallery:** The local-global problem still occurred. Two users were unsure whether the photos in the global media gallery are related to the affinity notes that are on the screen at the moment or to the whole affinity diagram.

- **Information card to front when touched:** In this design, it was not possible to move information cards on top of the affinity note. The affinity note was always on the upper most level of the screen. It is annoying when an information card is covered by the affinity note and has to be moved outside the note to be fully readable.
- **Multiple affinity notes should be able to be open:** In this implementation, an open affinity note was stacked again if another note had been opened. But this prevented from comparing information of different affinity notes.
- **Improve QR Code recognition rate:** The recognition rate of the QR Codes and therefore the recognition rate of the affinity notes on the wall was not very good. A new idea is to remember all found affinity notes of the last 2 seconds and display them even if they were no longer recognized.
- **Annotation and Media Card hardly movable:** Users had problems to move the annotation and the media card. On the annotation card, the big text input box was the problem while on the media card the two image stack buttons made problems when users tried to move the card while pressing on them.

3.1.5.1.3 Conclusion

The main problems that occurred during the user test were the recognition rate of the affinity notes and that the media and the annotation card were hardly movable. To address the affinity note recognition problem, the idea of analyzing the last two seconds of recognized affinity notes has been implemented in the final implementation. The card moving problems have also been addressed.

3.1.5.2 Final implementation

Based on the findings of the user tests with the first Android implementation, the Android app was modified to address the issues. Additionally, and equally important, the server has been implemented and a network communication part added to the Android app to communicate with the server. The server stores the whole content that is displayed when a note is clicked. Whenever the app recognizes the QR Code of an affinity note, it makes a lookup on the server for the data that is stored about the note. When a user attaches a photo or adds an annotation, this change is communicated back to the server and updated there.

The whole solution with client, server and database is called AffinityScanner and the features are explained later in a dedicated section.

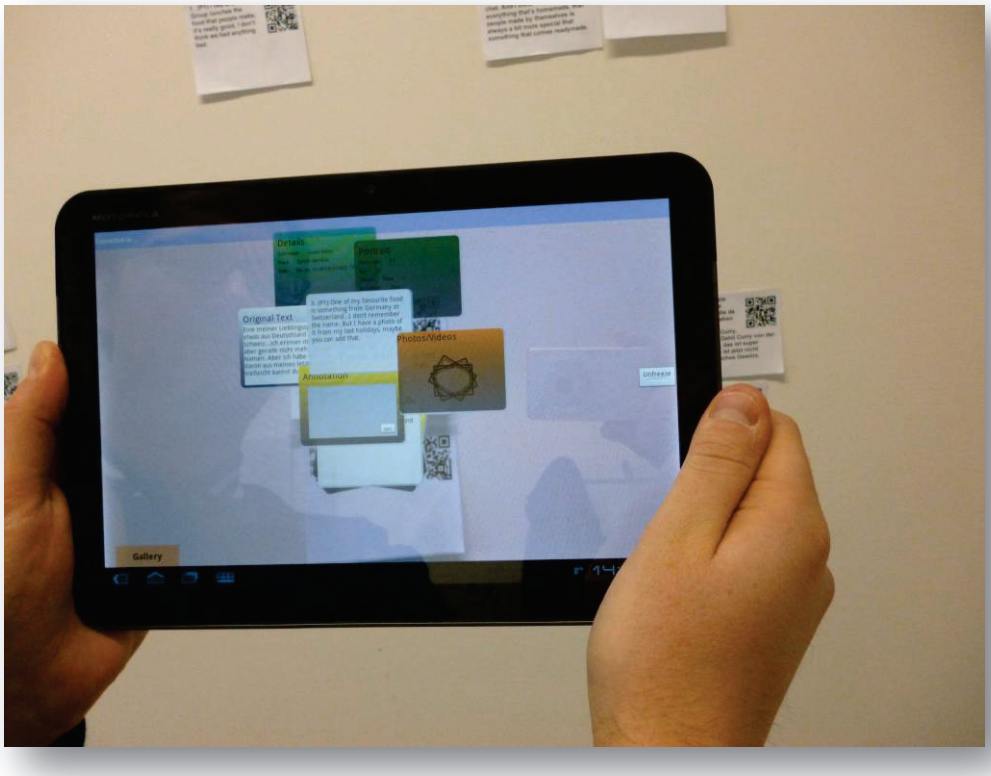


Figure 12: Final implementation of the Android prototype, called AffinityScanner

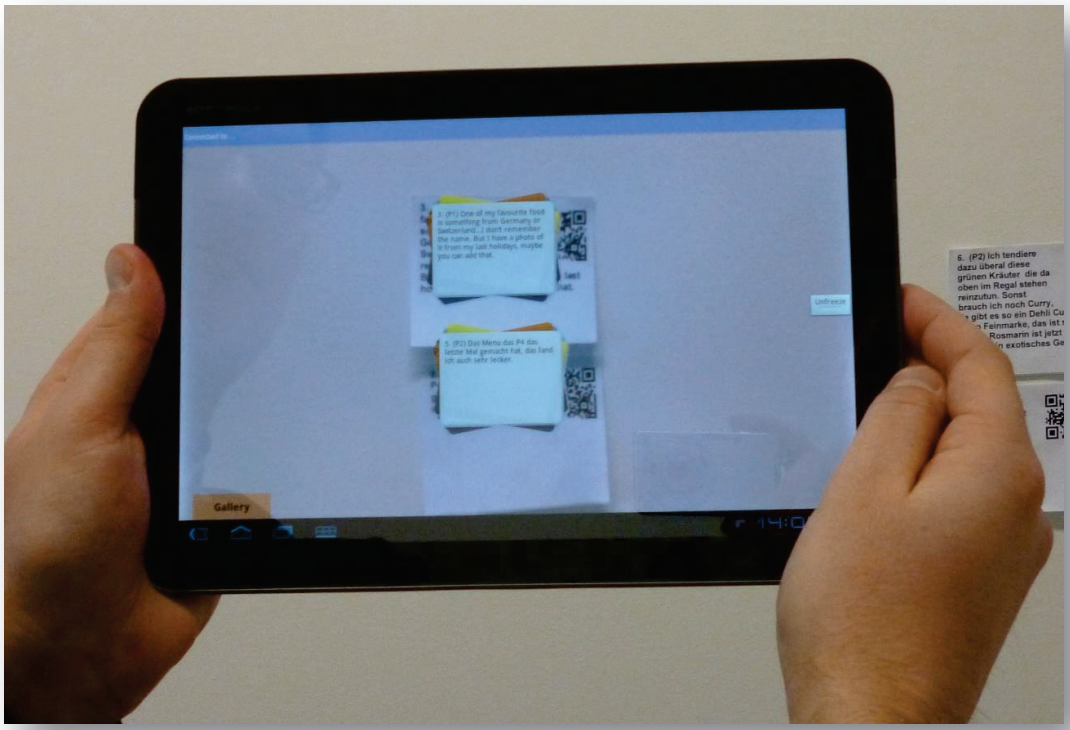


Figure 13: AffinityScanner with two recognized affinity notes. The information cards are stacked behind the note.

3.1.5.2.1 Functionality

The functionality is basically the same as in the first Android prototype. Minor changes have been made that are explained in the next paragraph. The whole functionality of the system is explained in section AffinityScanner.

3.1.5.2.2 Design decisions

The following changes have been made in the final design.

- **Annotation Card input box:** To address the input box problem, the input box was changed to a normal text box without any input functionality. The user had to press an edit button if he wanted to input text.
- **Information Cards come to front:** When an information card is clicked, it comes to front on the display, so it is the uppermost card on top of the affinity note.
- **Recognition rate improved:** To address the affinity note recognition problem, the last two seconds of recognized affinity notes are displayed together with currently recognized affinity notes.
- **Annotation and Media Card moving problem:** On the Annotation Card the input text box has been changed to a normal text box. So the user can now move it without triggering the keyboard. On the media card, the photo and video button have been merged into one media button. This leaves more room outside the button for the finger to move the card around.

3.1.5.2.3 User test

The user study that was made with the final implementation is described and discussed in the Evaluation section.

4 AffinityScanner

The final implementation has been named AffinityScanner. The system consists of specialized affinity notes, a server and a tablet application. The functionality of these elements is explained in this section.

4.1 Affinity Notes

In order to enable the tablet application to recognize the notes, the affinity notes are enriched with a QR code next to the text. With this code the application can access the additional information that is stored in a database on the server. Figure 14 shows an example of an affinity note that is used with the application. The affinity note contains a note number (4), a participant number (3), the text (2) and the QR code (1). In the QR code the note number is encoded. The code is decoded by the AffinityScanner client and the note number is used to look up the information about the note on the server.

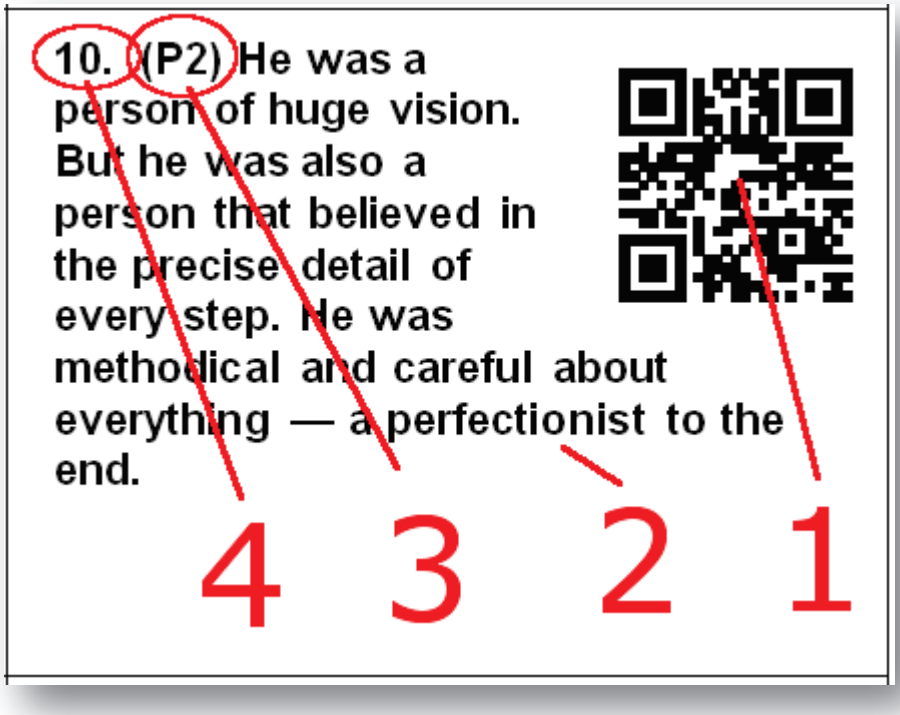


Figure 14: Affinity note how it is was used in the user tests

The special affinity notes are created with a word macro that takes the note text as input and formats the affinity notes with the QR code on a sheet of paper that can be printed out.

4.2 Server

The whole data that is displayed on the client is stored in a database on a server. Before one can start with the affinity diagram, the note text of all affinity notes has to be stored into the database. In this database also the additional (context) information about the affinity notes is stored. This means information about the data source of the text or who collected the data or photos which should be used during the affinity diagram.

4.3 Android Tablet App

To run the AffinityScanner client a tablet computer from Motorola is used that runs Android 3.1 and has a camera at the backside.

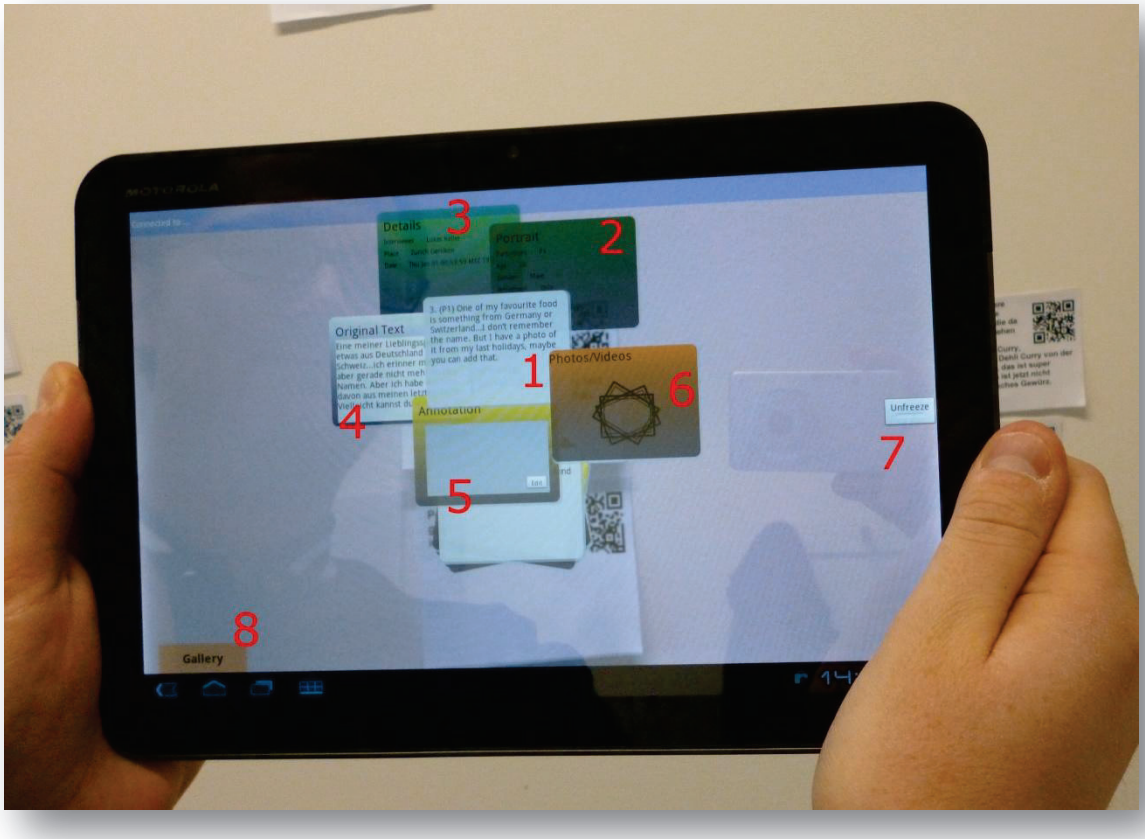


Figure 15: Affinity note with expanded information cards

When the application is launched, the user just sees the environment behind the tablet on the screen and the gallery slider and the freeze button (Figure 15 number 8 and 7). As soon as an affinity note is recognized, a duplicate of the note is placed over the physical note (Figure 15 number 1) on the screen. When the user clicks the digital note, the information cards are placed

around the affinity note (Figure 15 numbers 2,3,4,5 and 6). Five cards that are movable with the finger are displayed.

- **Portrait Card (2):** Displays information about the participant that this affinity note is based on.
- **Detail Card (3):** Displays information about the data source this affinity note is based on. Normally an interview.
- **Original Text Card (4):** Displays translations if available.
- **Annotation Card (5):** Displays attached annotations. They can be edited by the user.
- **Media Card (6):** This card has a button that looks like stacked images if media files are attached to the note. When clicked, the attached media files are placed on the screen. When clicked again, they disappear.

Figure 16: All media items from the media gallery are placed on the screen

The media gallery (8) on the lower left corner can be slid up. When slid up, it has a similar button as the media card that places all media files on the screen that are in the database. This lets the user access all the media from the whole affinity diagram. These media files can be attached to any affinity note by dragging and dropping them on the note when it is displayed on the screen.

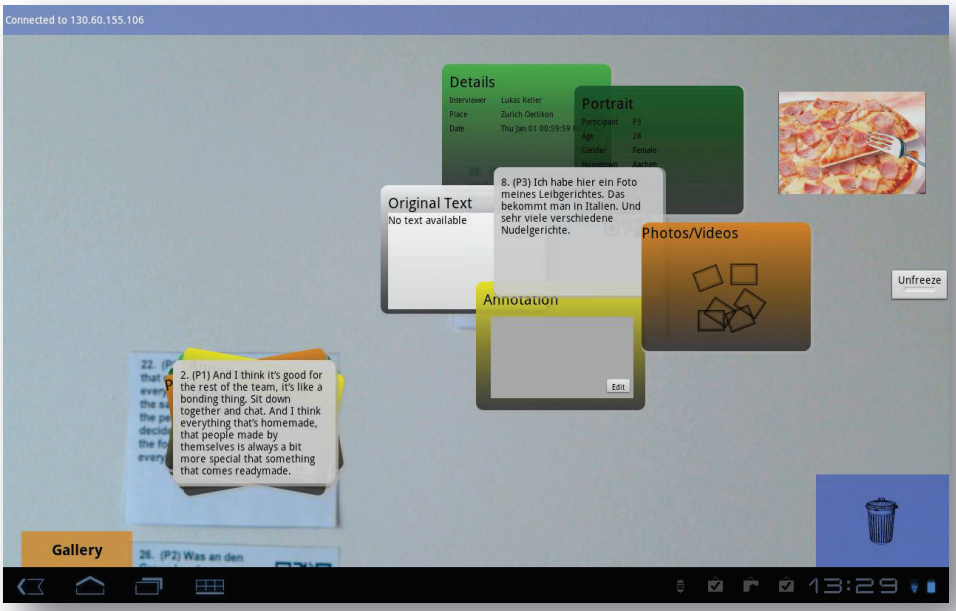


Figure 17: Attached photo of affinity note 8 with a trash can to remove it

The freeze button (7) is used to pause the image recognition routine. The affinity notes that are currently recognized and displayed on the screen are frozen and the camera is paused and does not try to recognize new affinity notes. This is helpful when interacting with the notes, because the tablet device does not need to be pointed to the wall anymore and can be held in more convenient positions.

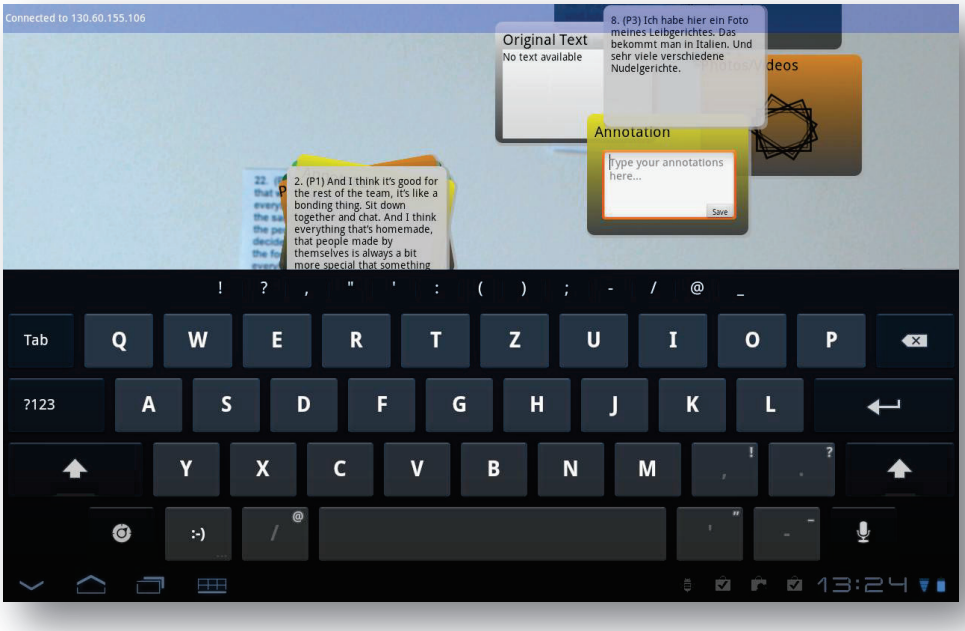


Figure 18: Keyboard slides up when editing an annotation

5 Implementation

The implementation process consisted of implementing a client, a server including a database and the network connection protocol. This section describes important aspects of the technical implementation.

5.1 Architecture

The basic architecture is a client-server model. The server manages all the information about the affinity diagram in a database. The client retrieves information from the server and sends back changes to the server, which updates the database accordingly.

The client server architecture makes it possible that multiple devices work on the same affinity diagram. This does not only mean multiple tablet clients, but also other solutions like cameras on a tripod that scan the whole diagram and communicate the position of every note to the server (such a solution belongs to the bigger idea this thesis is part of and is currently implemented in another master thesis at the same lab). The information gained by the camera may be used in the future to make keyword searches for notes on the tablet and guide the user with visual indicators on the screen to the corresponding note.

5.2 Server

The server is a JBoss Application Server 7[3]. As server and database was needed together, the bundle from OpenSCG[4] is used, which includes JBoss server and Postgres database.

The server is implemented in J2EE[5] and provides an API that can be used by clients. The API method calls are done on the server port 8443 as they use SSL connections and are in the format "https://<serveraddress>:8443/TabletServer/<apicallname>?param1=lorem&param2=ipsum".
The server by default only accepts calls that are made over SSL connections. Apart from the login method, all method calls need to include a security token that is assigned to the user when he authenticates with the login method (see Figure 19). This prevents access to the affinity data by unauthorized parties.

[3] http://www.jboss.org/as7
[4] http://openscg.com/se/jboss/
[5] http://java.sun.com/j2ee/overview.html

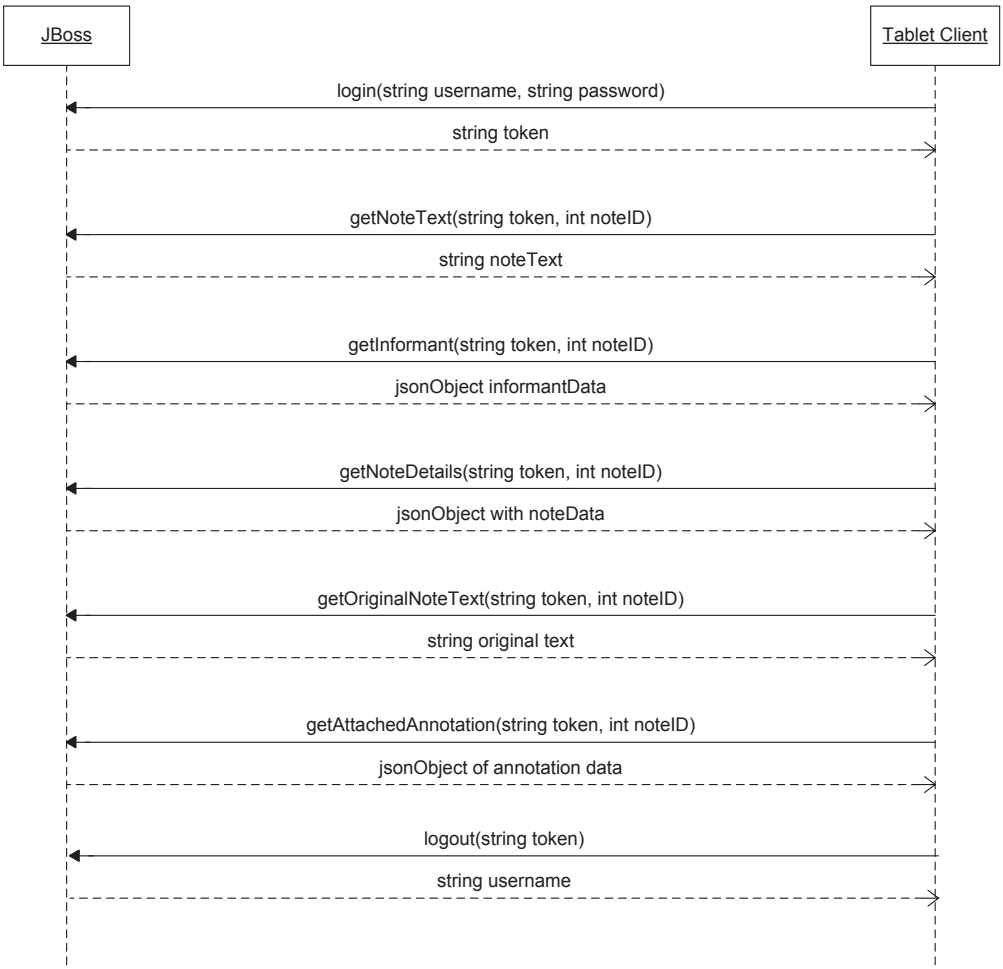


Figure 19: Sequence diagram of the communication between client and server. Note that the order of the calls between login and logout does not need to be the same.

The server has some configuration options which can be modified in a configuration file. Apart from general server settings, the validity of the security token can be changed and the size of the preview thumbnails of photos can be assessed. For debugging reasons, the enforcement of SSL connections can be disabled. But this option should be used with care.

5.3 Database

Postgresql version 9[6] is used as database to store the data. The full database schema is attached in the appendix. In Figure 20 the columns are shown which are important in order to see the relation between the tables. The "notes" table is the main table. It contains most of the information of the notes. The note text, the translations of the note text, a reference to the informant (e.g. the interviewee if the data source is an interview), a reference to the data source

[6] http://www.postgresql.org/

(the actual interview in this case) and a reference to the analyst who collected the information (i.e. the person who conducted the interview). The table “attached_media” records which media files are attached to which notes. The media files are stored as normal files in the servers file system. The database just records which filename is attached to which affinity note. The table “users” is used for accounting reasons, to let different password protected users access the diagram information with the client.

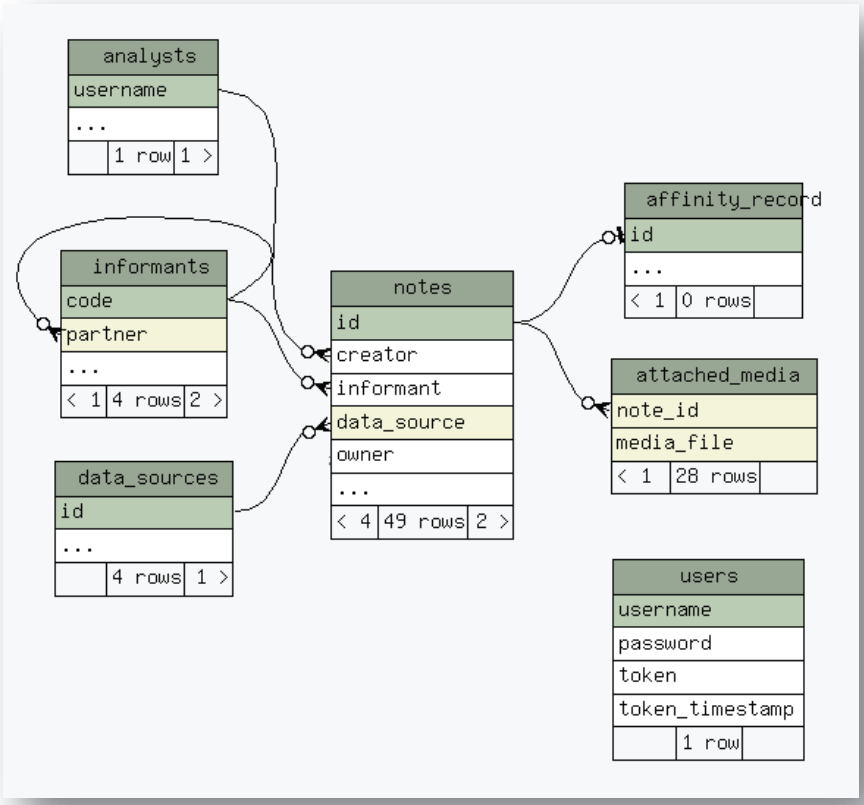


Figure 20: Shortened schema of the database with only the primary and foreign keys (created with SchemaSpy[7])

5.4 Client

The client is an Android application for tablets with at least Android version 3.0 Honeycomb[8]. It uses the rear-mounted camera to film the affinity wall. If it recognizes an affinity note with a QR code on the wall, it overlays the physical note with an image of the same note.

5.4.1 QR Codes

To enable the client to recognize the affinity notes on the wall, every affinity note has a QR code next to the note text. A QR code is a two-dimensional barcode that can contain text or internet

[7] http://schemaspy.sourceforge.net/
[8] http://www.android.com/

addresses encoded in it (see Figure 21). It can easily be read by computers and smartphones and is therefore often used in mobile applications ("What is a QR Code?," 2012). For the use on affinity notes, the QR code encodes the number of the affinity note in the database. With this identifier all information that is stored about the note can be looked up on the server.

The client application is built on top of the open source Android app ZXing[9], which is specialized for QR code recognition. The ZXing app handles the recognizing of the QR code, while the self-developed part of the client displays the additional affinity diagram information on an additional layer on the screen.

Figure 21: QR code containing the internet address http://www.ifi.uzh.ch/zpac.html

5.4.2 Information Lookup

When the client recognizes a QR code, it contacts the server with the decoded note number to retrieve all the information that is stored with the affinity note in the database. The client makes look-ups for the note text, the translated note text, details about the data source and participant, attached photos and attached annotations. As soon as the information is retrieved from the server, the client displays it on the screen.

Photos are retrieved in a two-phase process. First, small-sized grey-scale thumbnails of the images are downloaded and displayed. While the thumbnails are displayed, the full sized images are downloaded in the background. When the solution is used in slow networks, this approach provides the user a preview of the photos, while the long-lasting download of the full sized images is done in the background.

[9] http://code.google.com/p/zxing/

5.4.3 Security

As the data used in the affinity diagram is often not for public use, it is important to protect it from unauthorized access. The database server is considered as a secure environment and the data in the database is therefore not encrypted. But transmission of data over public networks requires special protection. For this reason the exchange of data between client and server is made over SSL connections and the client can only retrieve data from the server when providing a valid username and password.

5.4.3.1 Token-based access

The database has a table called “users”, where all users are stored who have access to the data. This enables different user access for different databases and makes it possible to have data of multiple affinity diagrams stored on the same database server in different databases. The “users” table contains the columns “username”, “password”, “token” and “token_timestamp” (see database schema in the Appendix). Username and password are the credentials the client has to provide when logging into the server. When a client logs into the server, the server generates a security token, stores it in the column “token” together with a timestamp of the actual time and sends the token to the client. For every consecutive call to the server, the client has to authenticate itself with the token. The token expires after a certain time, which can be changed in the server configuration, forcing the client to re-authenticate itself with username and password. The security token is a consecution of 16 lowercase characters, which is randomly generated by the server.

5.5 Limitations of the Implementation

This subsection discusses technical limitations of the current implementation which can be addressed in further work.

5.5.1 Password Encryption

As all the data in the database is not encrypted, the passwords in the table “users” are neither encrypted. An attacker who has access to the database can retrieve the usernames and passwords and is enabled to access the data with the client. But as the database server is considered as a secure environment, this aspect has been neglected.

5.5.2 Pinch Zoom

An intuitive way to zoom a photo is by a pinch gesture. It was planned to include this feature for the media files on the client. Due to a conflict between the move gesture and the pinch gesture, the pinch gesture has not been activated. In the current implementation, only one of the two

gestures on photos is possible. As moving the photos around is more important in the application, the pinch gesture has been disabled.

5.5.3 QR Code Recognition Rate

The recognition rate of the QR codes on the affinity notes is not very good in the current implementation. When the affinity wall is scanned from more than one meter, only few affinity notes are recognized. Ideas to improve the recognition rate are discussed in the Future Work section at the end of the thesis.

6 Evaluation

The final implementation of the Android prototype has been evaluated in a user test. The final implementation is called AffinityScanner and consists of a tablet client, a server and a database. The whole system has been evaluated, but as the test users only worked with the client, basically only the client was evaluated as the rest of the system worked in the background.

6.1 Focus of the Evaluation

The usability of the developed solution was a big focus during this thesis. Therefore one major focus of the evaluation was the usability of the Android tablet app. The other focus was how the app is used in the process of doing the affinity diagram. Therefore during the test and in the feedback round the following criteria were observed:

1. **Usability of the Android tablet app**
   What kind of problems do participants have with the app interface
2. **How is the Android tablet app used**
   Is the app used together, is the app a bottleneck because it can only be used by one person at a time, do users only use the app interface and not look at the physical diagram anymore, etc.

6.2 Method

The evaluation has been made with observations of the test users during the test and with a feedback round after the test. In the feedback round, special incidents which occurred during the test were discussed and prepared questions have been asked.

Prepared questions (translated from German):

- Were there problems handling the tablet application?
- When did you use the tablet application?
- Was there something that prevented you from using the tablet application?
- Was it a problem that there was only one tablet device available for all participants?
- Which functions of the tablet app did you primarily use?

6.3 User Test

The final user test was held in a room that is often used for affinity diagramming by the research group.

6.3.1 Participants

The test has been conducted with unknown test users recruited in online forums and job boards. Every participant received an incentive for the one hour test. Four tests were conducted in total, with three groups of three participants and one group of two participants (one participant did not show up). The participants were locals from Zurich, Switzerland and vicinity, age between 18 and 42 and the test was held in German.

6.3.2 Test Data

For the data used during the user test, an interview was conducted with members of the research group about the random topic of the weekly group lunch. Out of this interview data, about 50 affinity notes have been written and filled into the database, together with additional information about the interviewed persons and the interviews itself. Photos have also been added to the database. With the photos and attached annotations on notes, the affinity notes were digitally enriched, so that test users were able to make use of the features of the prototype.

6.3.3 Test Tasks

The test tasks of the users were to create an affinity diagram with these 50 affinity notes to answer the following questions:

1. What problems do the interviewed persons see in the weekly group lunches?
2. What do the interviewed persons like at the weekly group lunches?
3. What are the favorite meals of the interviewed persons?
4. Which ideas for improvement do you see for the weekly group lunch?

6.3.4 Activity

Prior to the test, a quick introduction on how to create an affinity diagram was shown to the participants. But the process on how to create the affinity diagram was free. The participants were also instructed in the use of the Android tablet app and were advised to use it whenever they needed more information about an affinity note.

On some notes, information was intentionally removed or changed, so that users were forced to use the app in some cases to get more information that was attached to the affinity note like photos or annotations. This was done because it was expected that test users might not use the Android tablet app if able to solve the tasks without it. Observing how users work with the app would then have not been possible.

6.3.5 Test Remarks

During the user test, some functions that were known to have a defect have been disabled, so that they do not influence the interaction with the app:

- **Adding photos with drag and drop:** Because the drag and drop function of photos on affinity notes had a bug and was not used during the test anyway, it was disabled.
- **Gallery update disabled:** The gallery of all photos of the affinity diagram had an out-of-memory bug and made the app crash under certain circumstances. The update was therefore disabled, as the gallery function was not used during the test. Participants only interacted with the photos that were attached to affinity notes.

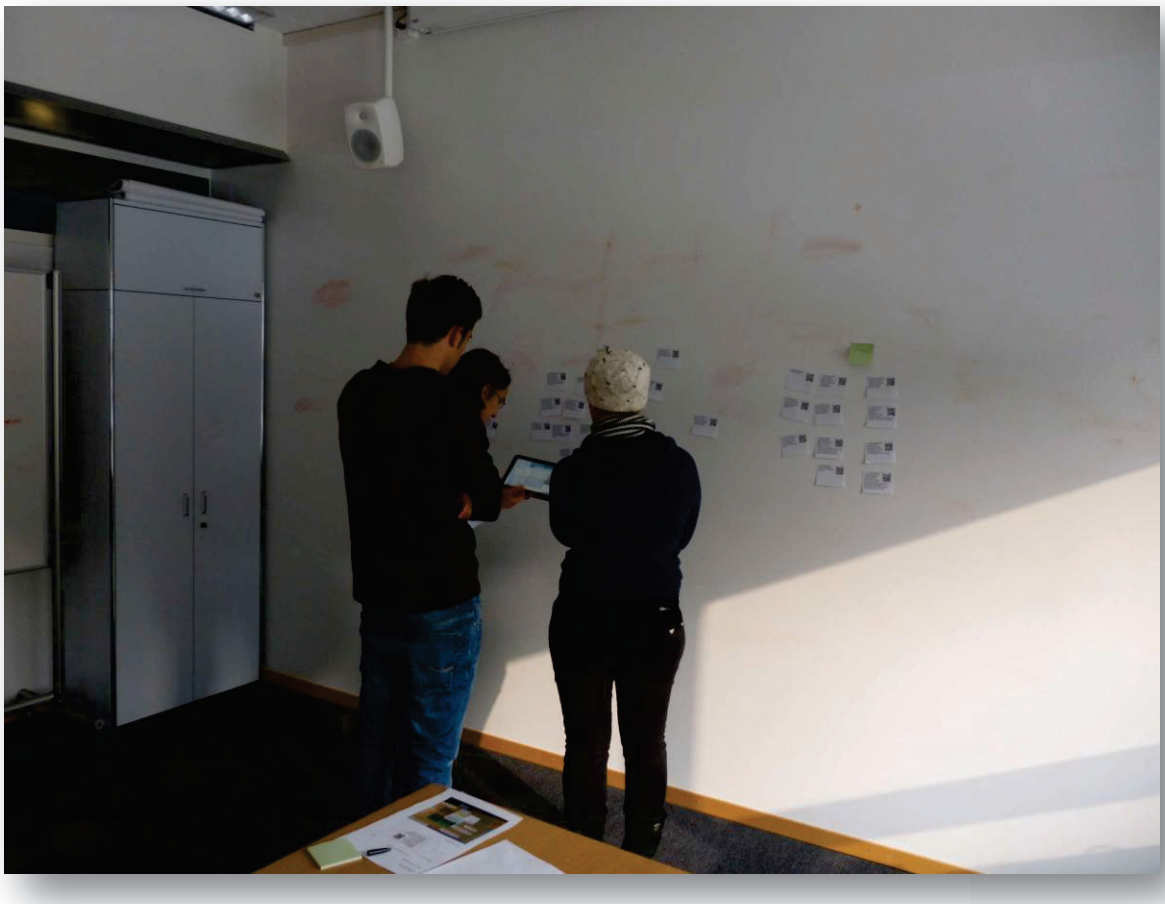


Figure 22: Test users consult the AffinityScanner

6.4 Observations and User Feedback

Data collected during the tests consists of observations by researchers and feedback from the test users. The feedback of the users is mainly about the usability of the Android tablet app

interface, as the users did not know the affinity diagramming method before and therefore did not comment on how the Android tablet app influenced their process of doing an affinity diagram. In the observations it has been observed how the app is used by the participants.

Participants per group:

- Group 1: 2 test users
- Groups 2, 3, 4: 3 test users

6.4.1 Observations

Noticeable observations:

- **Two phase process:**
  Almost all groups made the affinity diagram in two phases. First, they made group labels and put all the affinity notes on the wall, already grouping them. Then in the second phase they tried to answer the questions. In this phase they rearranged the affinity notes a lot and also used the app a lot more, because they had to find missing information that was not on the affinity notes to answer the questions given as test tasks.
- **Photo issue:**
  When users clicked on the photo stack of the media card, the photos were not displayed, but the card was brought to front. The users had to click a second time on the photo stack to make the photos appear.
- **Tablet use (who):**
  In two groups, the tablet app was mainly used by one person. The other persons also used the app, but much less frequently. In one group the app was used by all persons more or less with the same frequency and in one group the app was almost always used together by both test users and rarely individually. In this test group, the test users were mother and child, so this presumably influenced the way the app was used.
  In all groups it was observable that the tablet app was used differently in the two phases of the affinity diagram mentioned before. In the first phase, the Android tablet app was primarily used individually by different users, while in the second phase the Android tablet app was mainly used together by all users at the same time.
- **Tablet use (how):**
  In three of the four groups, the app was used right from the beginning by at least one user while in one group the app was ignored in the beginning. In all groups the app was used extensively in the second phase. In two groups that used the app right from the

beginning, the app was only used from time to time in the first phase. In the third group one test user used the app in the first phase almost all the time. In the second phase the other users also used the app.

- **Tablet use (what):**
  The app was mainly used to look up more information about a note. It was used to scan notes before they have been put on the wall, but mostly to scan the notes when they were already on the wall. Most of the times, only one note at a time was scanned. Sometimes multiple notes were scanned.

6.4.1.1 User Feedback

In the feedback round after the test, the users were asked about special events that occurred during the user test and also general questions about the Android tablet app and how it was used during the test from the users' perspective. The feedback round was held in the group, so the answers are normally group specific and not user specific. Only when a given user made an important statement is it mentioned. Aside from usability issues with the app handling found by the users and which are documented in the Usability Aspect Report in the Appendix, users answered multiple questions about the Android tablet app.

Confronted with the question in what moments they mainly used the app, all the groups said that they used the Android tablet app when they felt that some information was missing or that something was unclear. The first group used it often to check the German translation when the note text was in English. The third and the fourth group used it to check attached photos when they did not know what something, e.g. a meal, looks like.

Confronted with the question if there was something that prevented the users from using the tablet, no group had met any problems. But three of them mentioned that when it was clear to them what was written on the affinity note, they did not use the tablet.

In the fourth question, the groups were asked whether the fact that there was only one tablet device available for all participants was a problem. Group three mentioned that it would have sped up the process if they had two tablets. But two would have been enough. The other groups found that one tablet was enough for a group of three.

When asked about the primarily used functions of the Android tablet app, all groups mentioned the function to display attached photos. Group one used the function to display the translation a lot, while group three used it only once. The other groups did not use the translation at all. Group two and four mentioned that they used the annotations, but not very often.

6.5 Summary

The focus of the user test was the usability of the AffinityScanner client and to observe how the client is used in the affinity process. The findings from the user test are discussed in the following and interpreted at the end.

6.5.1 Usability Problems

The biggest usability problem that really affected the process of using the app in the affinity diagramming was the low QR code recognition rate. Apart from that several minor problems have been found. The following minor problems are taken both from observations and from user feedback. The user stated issues are taken from the Usability Aspect Report in the Appendix which has been created based on the user feedback.

- The screen has to be frozen to interact with the screen. The recognition rate of the QR codes does not allow the client to capture an affinity note reliably over a given period of time. This means that it often loses the affinity note moments after it has recognized and displayed it. This makes an interaction with the affinity note impossible without freezing the screen after the affinity note has been recognized.
- When an affinity note is scanned, all information cards are displayed, even if some of them do not have content (e.g. Original Text Card when there is no translation). It would be better not to show them and only show information cards that have content.
- The position of the information cards relative to the affinity note is not preserved when an affinity note is recognized again later on. Users would prefer to have them appear at the same position as before, so that they do not have to rearrange them again.
- Photos are put on the screen in random positions. Users would prefer to have them positioned near the button that triggered the photos.

While the minor problems are solvable with medium effort, the problem with the QR code recognition rate is an issue that probably needs to be addressed in a bigger scope.

One bottleneck is the camera of the tablet. The resolution of the camera in video recording mode with 1280 x 720 pixels is not very high (“MOTODEV,” 2012) and therefore has problems to capture the QR codes reliably from more than one meter. A camera with better resolution could be used or special preprocessing of the image before scanning it for the recognition code could be done.

Another bottleneck is the QR codes used. The QR codes are optimized to store a lot of content like entire internet addresses. That is far more than required to store the identifier of the affinity

note. A different type of recognition code on the affinity notes that has less storage capacity but is easier to detect than the QR codes could do a better job.

These problems should be solved in order to use the client as a serious support for affinity diagramming. The AffinityScanner solution is a good basis, but needs some improvement on the client side before it can be used in real affinity diagramming sessions.

6.5.2 Use of the AffinityScanner Client

As described in section Observations, the affinity diagramming method of the test groups could be divided into two phases. In phase one, the users put all the notes on the wall and in phase two, they analyzed and rearranged them to find the data that is needed to answer the questions. In all but one test group the client was not used very often in phase one. But in phase two, all test groups used the client intensely. It seems that, at least in this test setup, the tablet is not of good use when putting the affinity notes on the wall, because at this stage no exact information is needed about the content and the notes can be grouped after loose criteria. Later, when the affinity notes and affinity groups are needed to get insight into the data, more accurate information about the notes is desired and the data on the AffinityScanner client is consulted. The method to use the tablet only when special information is required fits well into the defined objective of the solution, which is not to force users to use it but to be of use when needed.

The app was mainly used to look up missing information and not to write annotations on notes. This is influenced by the fact that no task of the user test contained anything that would have forced users to do so and also that the diagram was small and one could easily keep track of the whole diagram any time. If the diagram were bigger, the need for annotations on notes could be more important.

Information was normally looked up by the group as a team. Normally the scanning was done just by one user but all other users stood around the tablet and analyzed the data on the screen. This may be due to the fact that the affinity diagram was very small (50 notes). Because it was small, the participants were able to check all affinity notes together without losing too much time. In the case of bigger diagrams however, this approach might be too time-consuming.

An event worth mentioning happened during the test run with group 3. The group had already arranged the affinity notes in their groups and was analyzing the data to answer the task questions. On one affinity note they checked the attached photo of a spoon and discovered thereby that the text on the note talked about a spoon. This discovery changed the meaning of

the note and they placed the note into another group. This is a nice example whereby the meaning of an affinity note can change when there is more context information available.

In general, the test users liked to use the app as the following statements of group two and three show.

"Sonst funktioniert es gut, ist recht schnell beim einlesen…Es ist recht intuitiv" - **user group 3**

> Translation: Apart from that [she mentioned multiple problems that are described in usability problems] the app works good, it is quite fast recognizing the affinity notes…it is quite intuitive

„*Die App ist ziemlich übersichtlich*" – **user group 2**

> Translation: The app is rather well-arranged

6.5.3 Interpretation

That the client was only used in the phase where more accurate information is needed about notes indicates that one of the objectives of the system is met. The objective is not to force the user to use it, but to support the user whenever he needs more information. That users liked the handling of the app in general also shows that the usability of the client application is good, apart from the discussed usability issues. This was another objective of the system and seems to be met.

The example from group three with the image of a spoon has not occurred in the other groups, but does nevertheless show the usefulness of context information. It shows that a photo can give needed information about note texts that are otherwise difficult to understand. This indicates that the concept of providing additional information about the affinity notes in an augmented reality style can be of use in the affinity diagramming process.

If the mentioned improvements on the client side are done, the AffinityScanner may become a valuable support for affinity diagramming teams.

7 Summary and Conclusions

The objective of this thesis was to develop a solution that can support researchers during the affinity diagramming process. It should be a solution that can be used by the researcher, but should not force him to use it. He should be able to decide at anytime in the affinity diagramming process to use it or not. The solution was built as an Android tablet application that can be pointed on affinity notes and display context information or enable the researcher to attach annotations. The interface of the app has been designed in an iterative design process including storyboards, team design critique sessions and user tests with three evolving paper prototypes. The final implementation on a Motorola Xoom tablet has been evaluated with eleven test users to observe the use of the application in the affinity diagramming process.

Before concluding this thesis in the last section, a short outlook on possible improvements for the developed solution is given.

7.1 Future Work

During the development, user tests and discussions with colleagues from the University of Zurich and Konstanz, various interesting points have been found that could improve the existing solution. The most promising of these features are described here.

7.1.1 Solutions for Known Issues

A possible solution for the problem of the low QR code recognition rate is to modify the QR codes. As the information encoded in the QR codes are only numbers, a QR code version with a lower storage capacity could be used. If the storage capacity is lower, the pattern in the QR code is less dense and could be easier decoded by the client (“QRcode.com,” 2010).

Test users put forward another idea for the QR code recognition problem. In case notes are not recognized by the scanner, the user should be able to click on the “physical note” on the screen and the scanner retries to find a QR code in this region.

7.1.2 New Ideas

An idea that came up, but has not yet been looked into, was to electronically detect group notes. When a user creates a new group he could take a blank sticky note with a QR code on it, write the group name on it and stick it to the wall. Then he scans the group note and since the system knows that the QR code is not yet in the database, the user can input the group name into the system. From now on the system recognizes the group note. With this approach it could be possible to search also for affinity groups.

Another idea which could be explored is a solution to the problem that detail and portrait card have static text fields to display information and are inflexible in adapting to different information. It would be better to implement them as small browsers and let the server send an HTML representation of the data. Every affinity diagram has different data to display and this would make it more flexible to adapt the solution to new affinity diagrams.

The last idea came up during a user test. A test user suggested a search function directly from an affinity note that is on the wall. In addition to a normal keyword search with a search textbox, it could be possible to click a word on a note and directly initiate a search for this word. The search results could either be combined with future solutions that provide positions of the notes or could be displayed as a list of notes found. While with future solutions the user could be directly navigated to the results on the wall, a simple list would also give a quick overview of other contexts in which the word has been used.

7.2 Conclusions

The tests have shown that the developed Android tablet application is usable to augment the affinity diagram and can support users with contextual information about the affinity notes, at the same time not forcing them to use the app or change the process in the way they make the affinity diagram.

It has been observed that decisions on where to put an affinity note on the wall have been changed after consulting additional information of this note with the Android tablet app. While the application is not yet a finished product and additional implementation work needs to be done, these observations give rise to the optimistic view that the app can be of use in future real affinity diagram sessions, giving the researchers a broader view on the data and support their decisions with more information on individual affinity notes.

Bibliography

Android. (2012). Retrieved February 27, 2012, from http://www.android.com/

Beyer, H., & Holtzblatt, K. (1997). *Contextual Design: Defining Customer-Centered Systems* (1st ed.). Morgan Kaufmann.

Beyer, H., & Holtzblatt, K. (1999). Contextual design. *interactions*, *6*(1), 32–42. doi:http://doi.acm.org/10.1145/291224.291229

Curtis, P., Heiserman, T., Jobusch, D., Notess, M., & Webb, J. (1999). Customer-focused design data in a large, multi-site organization. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99 (pp. 608–615). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/302979.303170

Dix, A., Finlay, J. E., Abowd, G. D., & Beale, R. (2003). *Human-Computer Interaction* (3rd ed.). Prentice Hall.

Doksam, G. (2012, January 31). *Where's the note? A Multimodal System to Augment Affinity Diagrams* (Master Thesis). University of Zurich, Zurich.

Geyer, F., Pfeil, U., Budzinski, J., Höchtl, A., & Reiterer, H. (2011). AffinityTable - A Hybrid Surface for Supporting Affinity Diagramming. *In INTERACT 2011: Proceedings of 13th IFIP TC13 Conference on Human-Computer Interaction, Lisbon, Portugal* (pp. 477–484). Springer. Retrieved from http://dl.acm.org/citation.cfm?id=2042182.2042225

Harboe, G., Doksam, G., Keller, L., & Huang, E. M. (2012). Two Thousand Points of Interaction: Augmenting paper notes for a distributed user experience. Presented at the CHI 2012, Austin, TX, USA.

Harboe, G., Minke, J., Ilea, I., & Huang, E. M. (2012). Computer Support for Collaborative Data Analysis: Augmenting Paper Affinity Diagrams. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12 (pp. 1179–1182). Presented at the CSCW 2012, New York, NY, USA: ACM. doi:10.1145/2145204.2145379

Holtzblatt, K., Wendell, J. B., & Wood, S. (2005). *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design* (illustrated ed.). Morgan Kaufmann.

Judge, T. K., Pyla, P. S., McCrickard, D. S., & Harrison, S. (2008). Using Multiple Display Environments for Affinity Diagramming. *Computing Systems* (pp. 9–12). ACM Press. Retrieved from http://workshops.fxpal.com/cscw2008/submissions/tmp6F.pdf

Klemmer, S. R., Newman, M. W., Farrell, R., Bilezikjian, M., & Landay, J. A. (2001). The designers' outpost: a tangible interface for collaborative web site. *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01 (pp. 1–10). New York, NY, USA: ACM. doi:10.1145/502348.502350

Minke, J. (2011, May 17). *Augmentation of the Affinity Diagram: Searching for Notes on the Affinity Wall* (Master Thesis). University of Zurich, Zurich.

MOTODEV. (2012). Retrieved March 5, 2012, from http://developer.motorola.com/products/xoom/

Nielsen, J. (1994). *Usability Engineering* (New ed.). Morgan Kaufmann.

Norman, D. (2002). *The Design of Everyday Things* (Reprint.). Perseus Books.

Open Handset Alliance. (2012). Retrieved March 2, 2012, from http://www.openhandsetalliance.com/

QRcode.com. (2010). Retrieved March 6, 2012, from http://www.qrcode.com/index-e.html

What is a QR Code? (2012). Retrieved March 6, 2012, from http://www.whatisaqrcode.co.uk/

Abbreviations

API	Application Programming Interface
CSV	Comma Separated Values
HTML	Hypertext Markup Language
SSL	Secure Sockets Layer

Glossary

Affinity Note:

This term means the paper artifact that is used in the affinity diagram as smallest unit. Normally it is a Post-it™ note type paper with some text on it. Sometimes the term refers also to the text itself.

CSV file:

This file type stores multiple values of any kind separated by commas. It is often used to exchange data when no special format for the data is given. For example passing contact books entries between different contact storage applications.

Diagrammer:

A diagrammer is a person that creates an affinity diagram. In special cases the person can also just be involved in the diagram, but is not actively creating the diagram itself.

Information Card:

The final implementation of the Android tablet app uses information cards to display information about the affinity note. An information card is a box on the screen that has the same shape as the affinity note (a rectangle), but contains other data instead of the note text. The information cards can be stacked under the affinity note, that is why they are called information cards. Types of information cards used in the interface are annotation card, details card, portrait card, media card and original text card.

QR code:

QR code is an abbreviation from Quick Response code and is a type of matrix barcode. A matrix barcode is a two-dimensional barcode that can easily be read by machines. The barcode can contain encoded information like internet addresses or normal text. QR codes are often used in the mobile environment to pass information between mobile devices.

Paper Prototype:

A prototype of the real system that is made out of paper or cardboard without any electronic elements. It is often used to verify early designs in user studies before they get implemented.

Secure Sockets Layer:

This is a protocol to communicate in a secure way over the internet. All communication between two stations that is done over this protocol is encrypted.

Tablet:

A tablet is an electronic device that has a touch screen that is used as display and as input element. The user interacts with the device by interacting with the touch screen. Tablets often have a built-in camera to capture images and videos. Popular tablet devices are the iPad from Apple or the Xoom from Motorola.

Appendix

A. User Manual

This manual gives instructions how to set up the AffinityScanner system and how to use it. To use the system, the database, the server and the client need to be running and connected. Set up the system in the order described here. Start the database server first, then the JBoss server and at the end, start up the client.

The AffinityScanner system consists of two binaries. The TabletServer is the code that is run on the JBoss application server and the AffinityScanner is the code that is run on the Android tablet.

This manual first describes the required system and gives hints to install it on a Windows machine. It then describes how to start up and operate the database, the server and the client.

Prerequisites

It is expected that a fully working JBoss Application Server with at least version 7 and a PostgreSQL database with at least version 9 is running on the server machine. The applications may also run with lower versions, but correct functionality cannot be guaranteed. Additionally, it is important that the PostgreSQL database is configured as the standard data source for the JBoss server.

This manual does not provide instructions how to install these systems. Nevertheless, hints are described in the following part to prevent the reader from making the same mistakes the author made installing the system on a Windows machine.

NOTE: All configurations on the JBoss server described here are made in the file “standalone.xml” in the directory <JBOSS_HOME>/standalone/configuration/. To make your life easier an already correctly configured standalone.xml file is on the accompanying CD of this thesis. The original file can be replaced with this one and the configurations described in this section can be omitted.

OpenSCG

As development system the JBOSS/PostgreSQL bundle from OpenSCG[10] has been used, because it includes all needed software.

[10] http://openscg.com/se/jboss/

Set PostgreSQL as standard data source in JBOSS

The PostgreSQL database must be configured as the standard data source for the JBoss server. There is a description how to set the PostgreSQL database as standard data source for the JBoss server on the OpenSCG website. Unfortunately, at the time of writing this thesis the description was still for JBoss Application Server 6, whose configuration files are different from JBoss Application Server 7. To help out, there is an instruction how to configure JBoss Application Server 7 with PostgreSQL here.

In the file <JBOSS_HOME>/standalone/configuration/standalone.xml put the following xml code into the node <subsystem xmlns="urn:jboss:domain:datasources:1.0"> … </subsystem>. Normally, another database is preconfigured as standard database in this node. Delete this database declaration by removing all xml code between the tags <subsystem xmlns="urn:jboss:domain:datasources:1.0"> and </subsystem> and put in the following snippet, so that only the snippet code is between the start and end tag of the subsystem node.

```
<datasources>
        <datasource jndi-name="java:jboss/DefaultDS" pool-name="postgresDS" enabled="true" jta="true" use-java-context="true" use-ccm="true">
          <connection-url>
                jdbc:postgresql://localhost:5432/jboss?charSet=UTF-8
          </connection-url>
          <driver>
                postgresql-9.1-901.jdbc4.jar
          </driver>
          <transaction-isolation>
                TRANSACTION_READ_COMMITTED
          </transaction-isolation>
          <pool>
                <min-pool-size>
                      10
                </min-pool-size>
                <max-pool-size>
                      100
                </max-pool-size>
                <prefill>
                       true
                </prefill>
                <use-strict-min>
                      false
                </use-strict-min>
                <flush-strategy>
                      FailingConnectionOnly
                </flush-strategy>
          </pool>
          <security>
                <user-name>
                      jboss
                </user-name>
                <password>
                      hci
                </password>
          </security>
          <statement>
                <prepared-statement-cache-size>
                      32
                </prepared-statement-cache-size>
          </statement>
        </datasource>
        <drivers>
                <driver name="org.postgresql" module="org.postgresql"/>
        </drivers>
</datasources>
```

The user name and the password in the security sub node of the snippet need to be modified if a different ‘database user’ is used.

After inserting the snippet into the standalone.xml file, copy the PostgreSQL JDBC jar file (normally named something similar to “postgresql-9.1-901.jdbc4.jar”) from “<JBOSS_HOME>/standalone/lib” to “<JBOSS_HOME>/standalone/deployment”.

Correct language setting issue in PostgreSQL

A problem with the language settings has been discovered. This may only be a problem on non-US machines. In order to correct the problem, you need to change the values of the following parameters in the file <POSTGRESQL_HOME>\data\postgresql.conf to 'German_Switzerland'.

-lc_messages

-lc_monetary

-lc_numeric

-lc_time

Correct Linux related bug on Windows

Linux specific network settings may cause problems when PostgreSQL is run under Windows. Therefore you have to comment out line 66 in the file <POSTGRESQL_HOME>\data\pg_hba.conf with a '#'-sign at the beginning of the line (the line after: '# "local" is for Unix domain socket connections only').

Open ports in JBoss and PostgreSQL

By default, PostgreSQL database and JBoss allow only access from the same machine. While this may work for the PostgreSQL database as long as it is on the same machine as the JBoss server, this will be a problem when the client tries to access the server.

JBoss

Add the following declarations in the file <JBOSS_HOME>/standalone/configuration/standalone.xml. Under the node <interfaces> add the following interface:

```
<interface name="global">
        <any-address/>
</interface>
```

Under the node <socket-binding-group … > change the following two lines

```
<socket-binding name="http" port="8080"/>
<socket-binding name="https" port="8443"/>
```

to:

```
<socket-binding name="http" port="8080" interface="global"/>
<socket-binding name="https" port="8443" interface="global"/>
```

Database

Hopefully the database has been installed successfully by now. In order to use the database, it has to be configured with a special database schema and filled with data. How to do this is described in this part.

Create User

The server uses the username jboss to access the database. As this is not the standard user of PostgreSQL it has to be created. The easiest way to create a new user is with the software PgAdmin III that comes with PostgreSQL. You have to create the user "jboss" with password "hci" in order to make the system work. The username and password are hardcoded in the TabletServer code and cannot be configured.

Create Database

To create the database, copy the database creation script called "affinityserver.sql" on the accompanying CD of this thesis to the server and execute the following command in a shell window (this command assumes that the script is under C:\ and the database user is jboss).

<POSTGRESQL_HOME>\bin\psql.exe -f C:\affinityserver.sql --username jboss

This creates an empty database called "Affinity" that contains all required tables. Bear in mind that you can rename the database, but have to change the default database setting then in the TabletServer's configuration file (described later).

Import Data

To import the data of the affinity notes into the database table "notes" of the database, a Python script has been created. This script is on the accompanying CD of this thesis and can be used after Python has been installed (at least Python version 3[11]). The script takes a CSV file in the same folder called dataimport.csv as input and creates an SQL import script called dataimport.sql. The SQL import script can then be executed in PgAdmin III to directly import the data into the table.

The CSV file must have at least these columns that are named after the columns in the database table (they are required by the "notes"-table):

- id
- qr_code
- note_type
- body

[11] http://python.org/

The first line in the CSV file looks for example like this (the words in parentheses are the data types of the columns in the database):

id(integer);qr_code(integer);note_type(smallint);body(text)

It is important that after each column name, the data type of the column is provided in parentheses. The data type of the columns can be checked in PgAdmin III.

Other columns like “translations” can be added if other data is available (which normally is the case).

CSV files made with Microsoft Excel are compatible with the script. Hence one can import data from Excel when saving the data as a CSV file with the column names and data types as headers of the columns (header = first line of a column in Excel).

The Python script needs to be in the same directory as the dataimport.csv file. When run, the file dataimport.sql is created in the same directory.

IMPORTANT: The generated script can only be used to import the data into the database and not to update existing values in the database. So when one of the datasets in the CSV file is already in the database, PostgreSQL will throw an error.

Server

At this point, the JBoss server should be successfully installed and running.

Most of the description on the server configuration in the standalone.xml file has been done in the “Prerequisites” part. To make your life easier, you find an already correctly configured standalone.xml file on the CD accompanying this thesis that already contains the configuration described there. This file can be taken to replace the standalone.xml that comes with the distribution.

This part describes how to start the server binary, how to do the required steps to setup the SSL certificate and how to create the repository for the media files. It is important that you restart the server after executing all steps in order to activate the new settings.

It is important to not confuse the JBoss server and the actual server code that has been developed in this thesis. The JBoss server is a web server that runs java code in WAR packages. The developed code is such a WAR package and is called TabletServer. The TabletServer package

needs to be put in the deployment folder of the JBoss server and is then executed by the web server.

Start the TabletServer

As JBoss is the web server, the actual code that is run is in the folder TabletServer.war on the CD. When placed in the deployments folder of JBoss, it is executed as WAR-binary. Therefore this folder needs to be put into the JBoss deployment folder (<JBOSS_HOME>\standalone\deployment) and a file called “TabletServer.war.dodeploy” in the same directory needs to be created. JBoss then automatically detects the WAR-folder and launches it. From then on, the TabletServer can be used.

Media Files

While the server already works now, another step has to be taken if images should be included in the affinity diagram. The media files are not stored in the same folder than the TabletServer code. They are stored in a separate folder called “affinitymedia.war”. In order to use media files with the server, a folder with the name “affinitymedia.war” must be created in the <JBOSS_HOME>/standalone/deployment directory (same directory as the TabletServer). The “affinitymedia.war” folder must contain a sub folder called WEB-INF, containing the file web.xml. The file should have the content of the following xml snippet:

```
<?xml version="1.0"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
    <description>Tablet Scanner media files</description>
    <display-name>Tablet Scanner media files</display-name>
    <filter>
        <filter-name>ExpiresFilter</filter-name>
        <filter-class>org.apache.catalina.filters.ExpiresFilter</filter-class>
        <init-param>
            <param-name>ExpiresByType image</param-name>
            <param-value>access plus 1 month</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>ExpiresFilter</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
</web-app>
```

The directory structure of the “affinitymedia.war” folder should look like the example below (in this view already containing three image files).

```
affinitymedia.war
|   Backofen.jpg
|   Bratwurst.jpg
|   bread-sandwich.jpg
└──WEB-INF
     web.xml
```

All media files (in the current version of the TabletServer only image files are supported) are read in and are made available by the TabletServer to the client. The TabletServer creates a sub folder “small”, where it stores the thumbnails of the images. This folder is created automatically if it does not exist already.

When the folder structure is created, a file called “affinitymedia.war.dodeploy” needs to be created in the same directory as the affinitymedia.war folder is located. This triggers JBoss server to make the content of the affinitymedia.war available online (under the address https://serveraddress/affinitymedia/...). The client can then download the files from there.

SSL

Communication is done via SSL connections. The certificates for the communication between server and client are self-signed. The client has a self-signed root certificate stored, while the server uses a certificate that is signed by this root certificate. This small certificate chain is needed, because the client only communicates with servers who have a certificate named with the server’s current IP address. Therefore every time the server changes its IP address, the server needs a new certificate. As the new certificate is signed by the root certificate, the client accepts the new certificate even if he does not trust the certificate itself.

The same root certificate as on the client is also stored in a keystore called zpacrootkeystore.jks on the CD. This is the root certificate used to sign the certificate that the server uses for the SSL connections. To create a new certificate on the server, do the following:

Copy the zpackeystore.jks and the zpacrootkeystore.jks from the CD to the directory <JBOSS_HOME>\standalone\configuration. In these keystores there are the custom certificates that are used for SSL connections. The file zpackeystore.jks holds the certificate that is used by the server for the communications. The file zpacrootkeystore.jks holds the root certificate that is required to sign new certificates.

To configure the JBoss server to use the certificate for SSL connections, add the following connector definition

```
<connector name="https" protocol="HTTP/1.1" socket-binding="https" scheme="https" secure="true">
        <ssl name="https" password="bookmark" certificate-key-file="../standalone/configuration/zpackeystore.jks"/>
 </connector>
```

to this part in the <JBOSS_HOME>/standalone/configuration/standalone.xml file (under the first connector node):

```
<subsystem xmlns="urn:jboss:domain:web:1.0" default-virtual-server="default-host">
        <connector name="http" scheme="http" protocol="HTTP/1.1" socket-binding="http"/>
        <virtual-server name="default-host" enable-welcome-root="true">
                <alias name="localhost" />
                <alias name="example.com" />
        </virtual-server>
</subsystem>
```

Then copy the Python script create_trusted_cert.py from the CD into the same directory as the keystores and start it. This script creates a certificate for the actual IP address and stores it into the server's keystore. At the start the script asks the user if it should delete existing certificates. This question should be answered with yes, because the server has problems if there is more than one certificate in his keystore.

If the script like in this case is in the same directory as the two keystores, it creates the certificate without prompting the user for the keystore locations. If the script cannot find the keystores in the same directory, it asks the user for the location.

If you like to verify if the certificates are created correctly, the application "Portecle"[12] is of good use. It can open the keystores and display the content. To find out the password of the keystores, search in the Python script (create_trusted_cert.py) for it.

The script uses the "keytool" program that comes with Java. It is important that the Java commands are directly executable from the command line from any location in order to run the script.

Another requirement for the script to run properly is that the "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" are installed. They are downloadable from the Java

[12] http://portecle.sourceforge.net/

website together with instructions on how to install them. The files are required because Java cannot handle long key length for the SSL connections otherwise.

Configuration File

The TablerServer has its own configuration file (do not confuse with the standalone.xml-configuration file from the JBoss server). This configuration file is in the “config”-folder of the TabletServer project (TabletServer.war) in the <JBOSS_HOME>\standalone\deployment directory and is called “configuration.xml”. The possible settings are described in the file itself. Do not forget that changes to the file only take effect after restarting the server.

Client

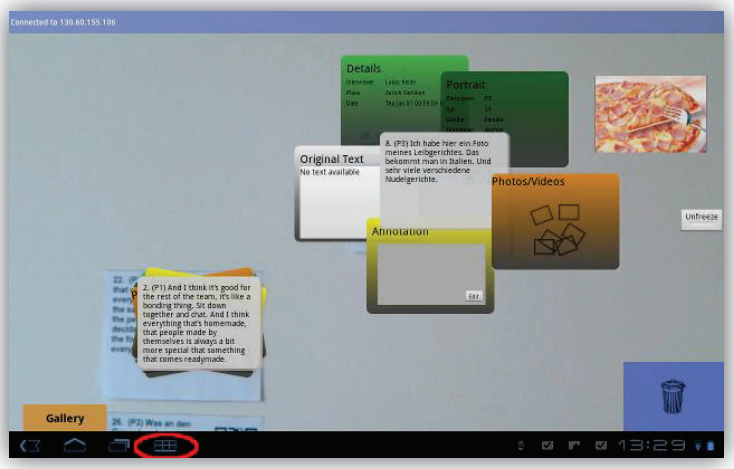
The client does not have a lot of options. It automatically tries to connect to the IP address that is set in the preferences. As login name it takes “lukas” and as password “zpac”. Unfortunately this is hard coded and cannot be changed in the interface.

Install application

To install the application, take the AffinityScanner.apk file on the CD and copy it to the USB-connected tablet. Then open the location on the tablet with a file browser and install the application by opening the file.

Server IP

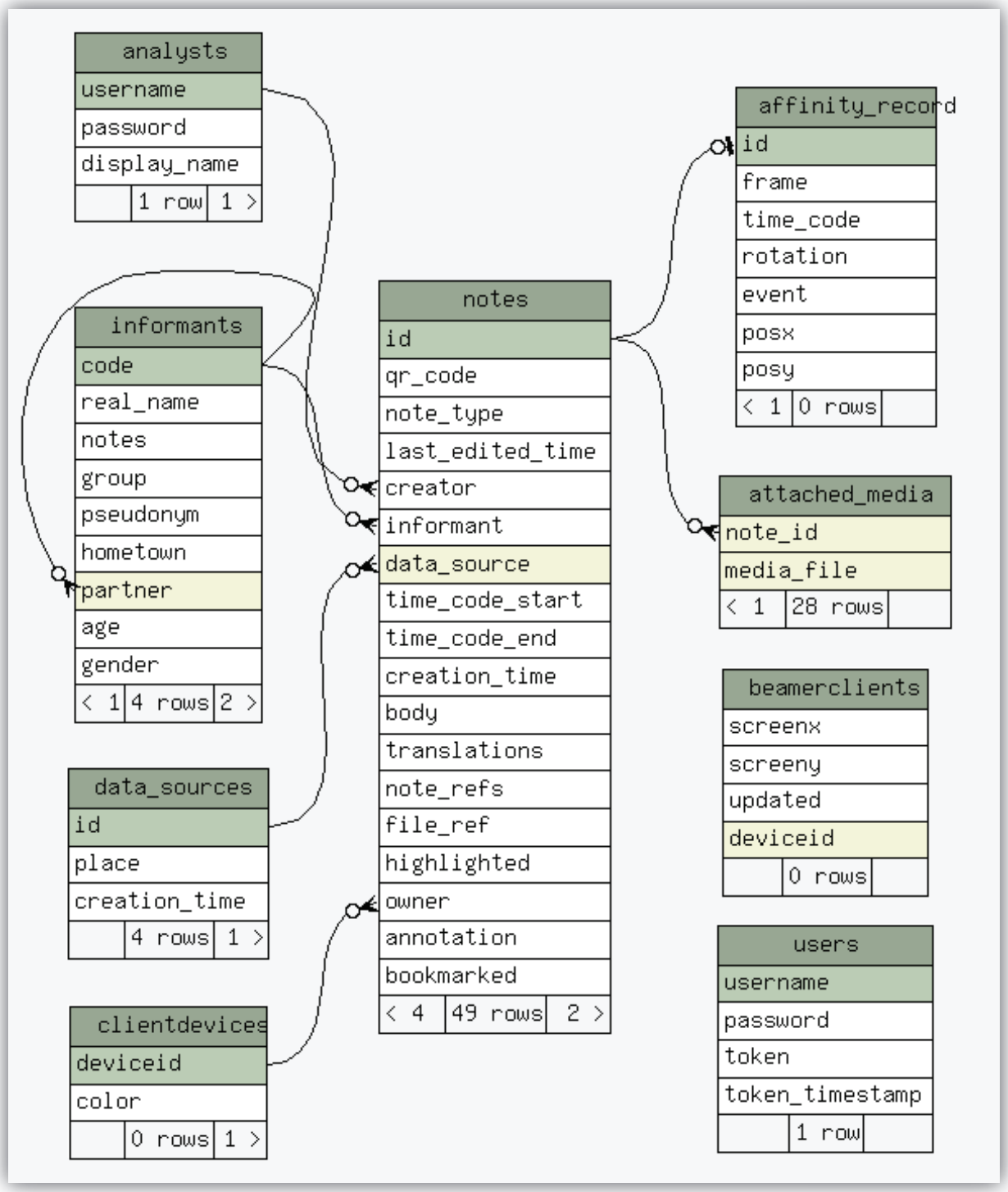
The server IP can be changed in the preferences of the client. Access the preference panel by clicking the icon that has a red circle in the image.



Known bugs

- Too many images in media gallery leads to out of memory crash sometimes

B. Database Schema



C. Sketches

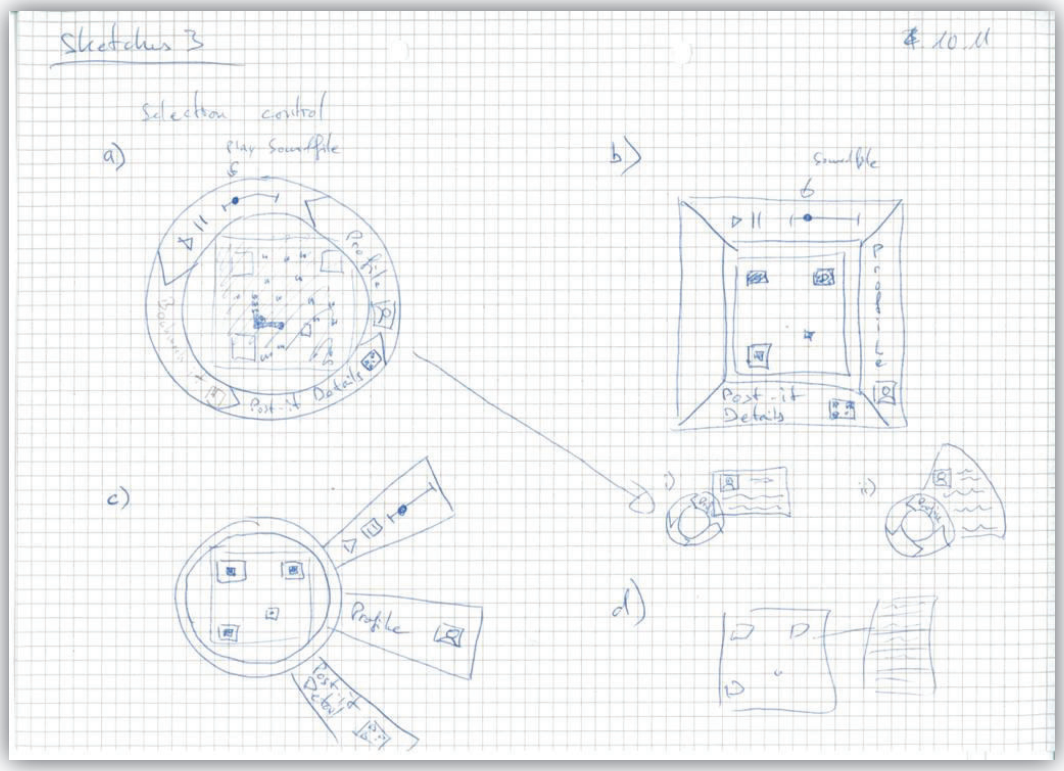


Figure 23: Early ideas of control elements of the digital affinity notes

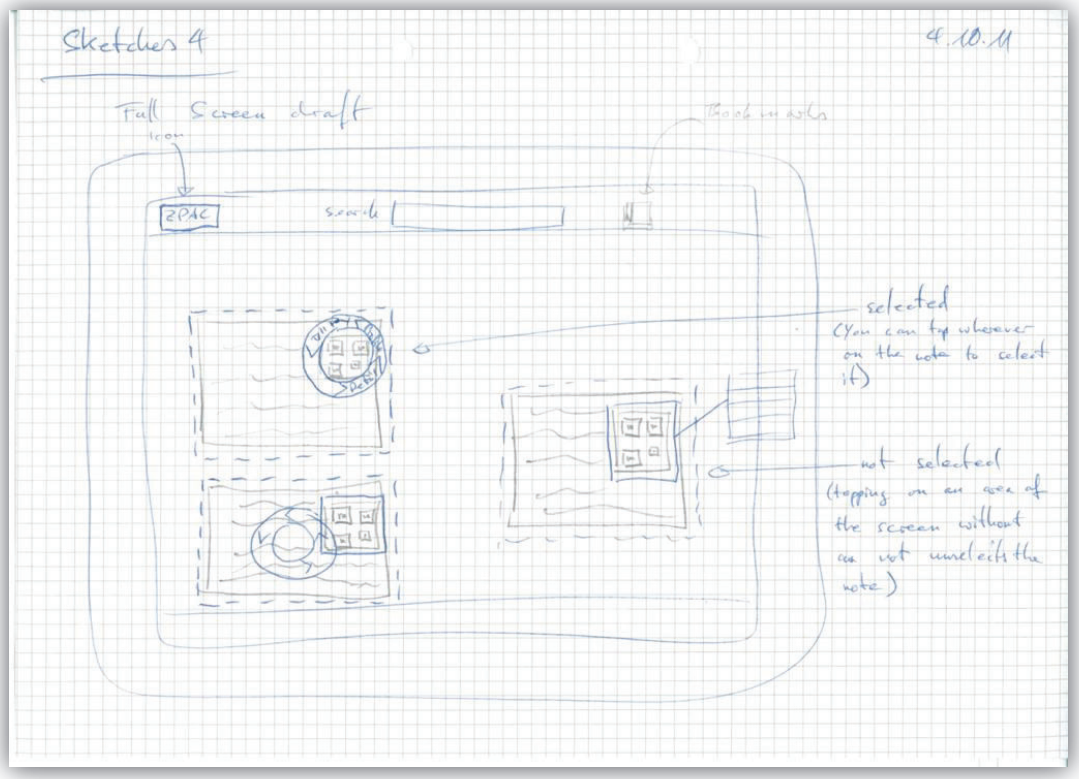


Figure 24: Sketch of the interface of the AffinityScanner client

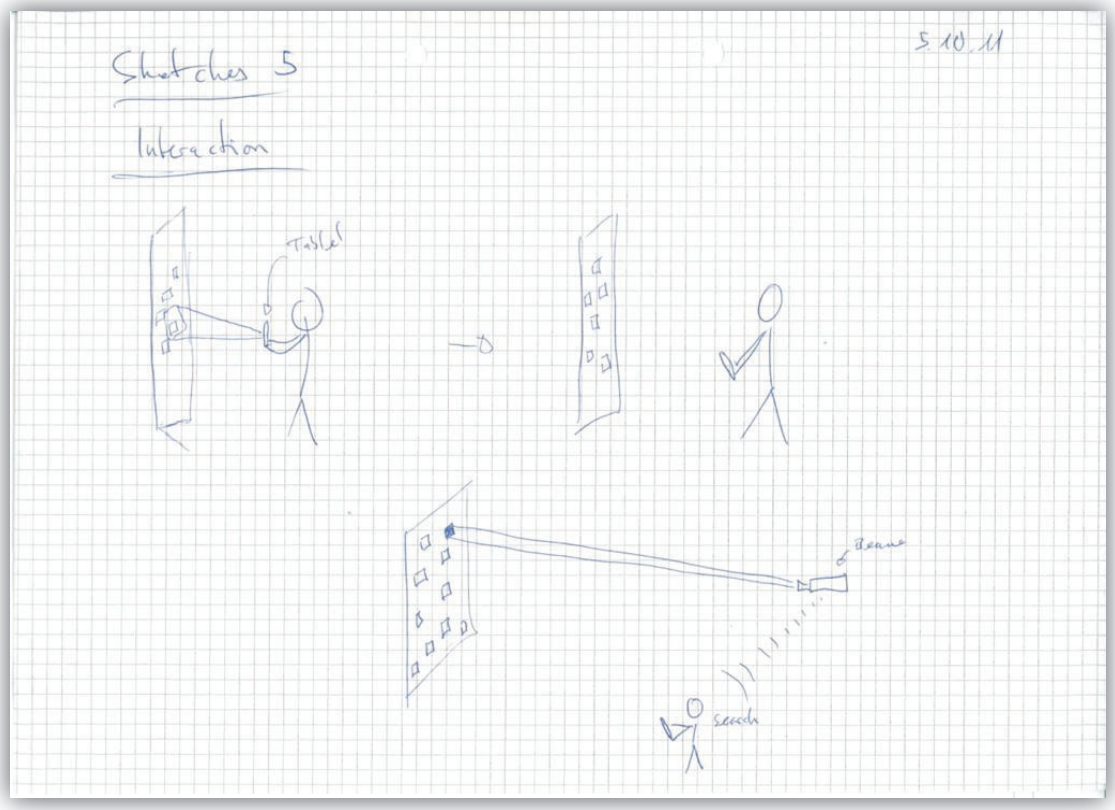


Figure 25: Not implemented search function that would have worked together with the solution from Doksam (2012)

D. First Storyboard Iteration

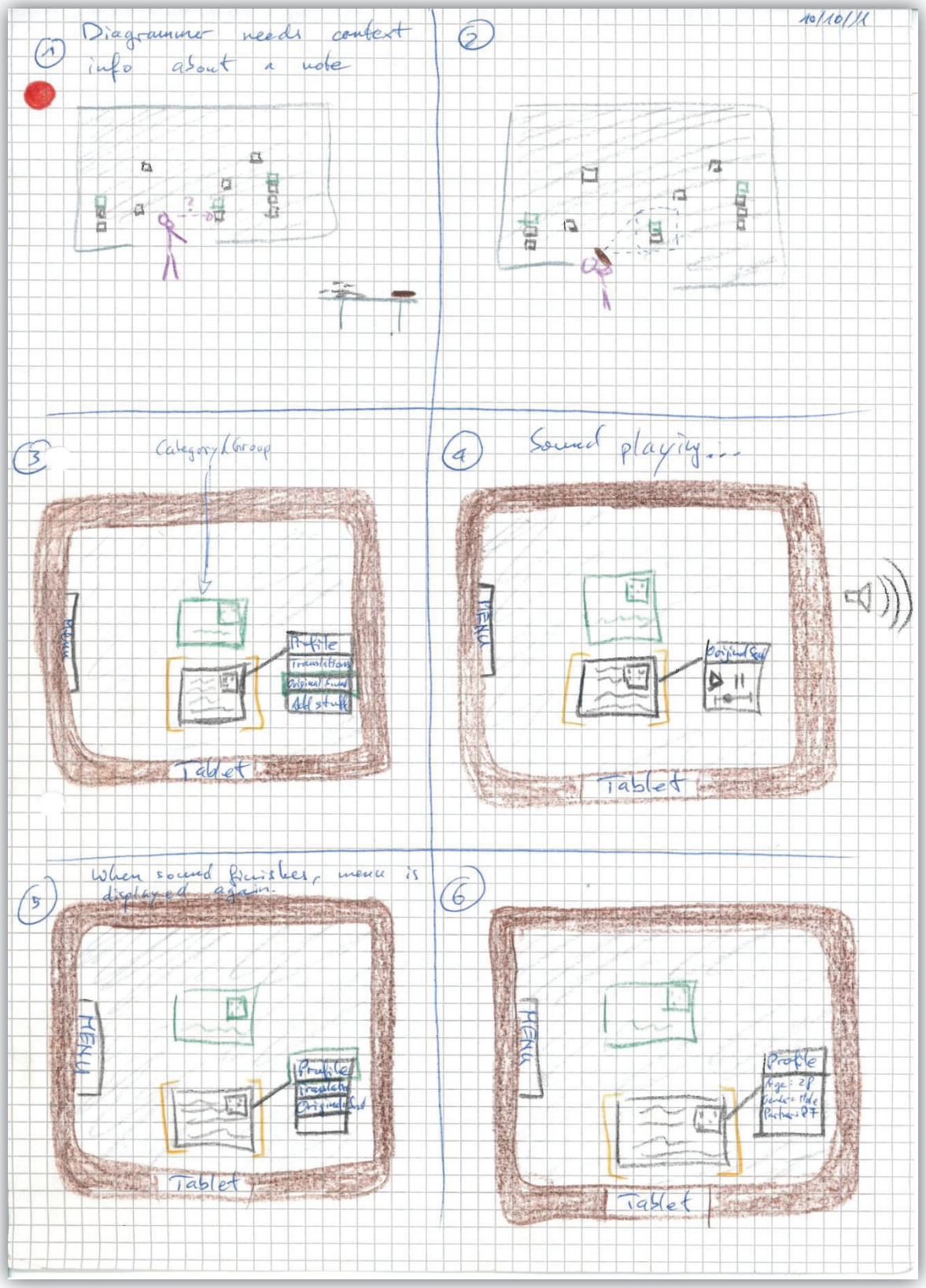


Figure 26: Storyboard of information lookup with the AffinityScanner

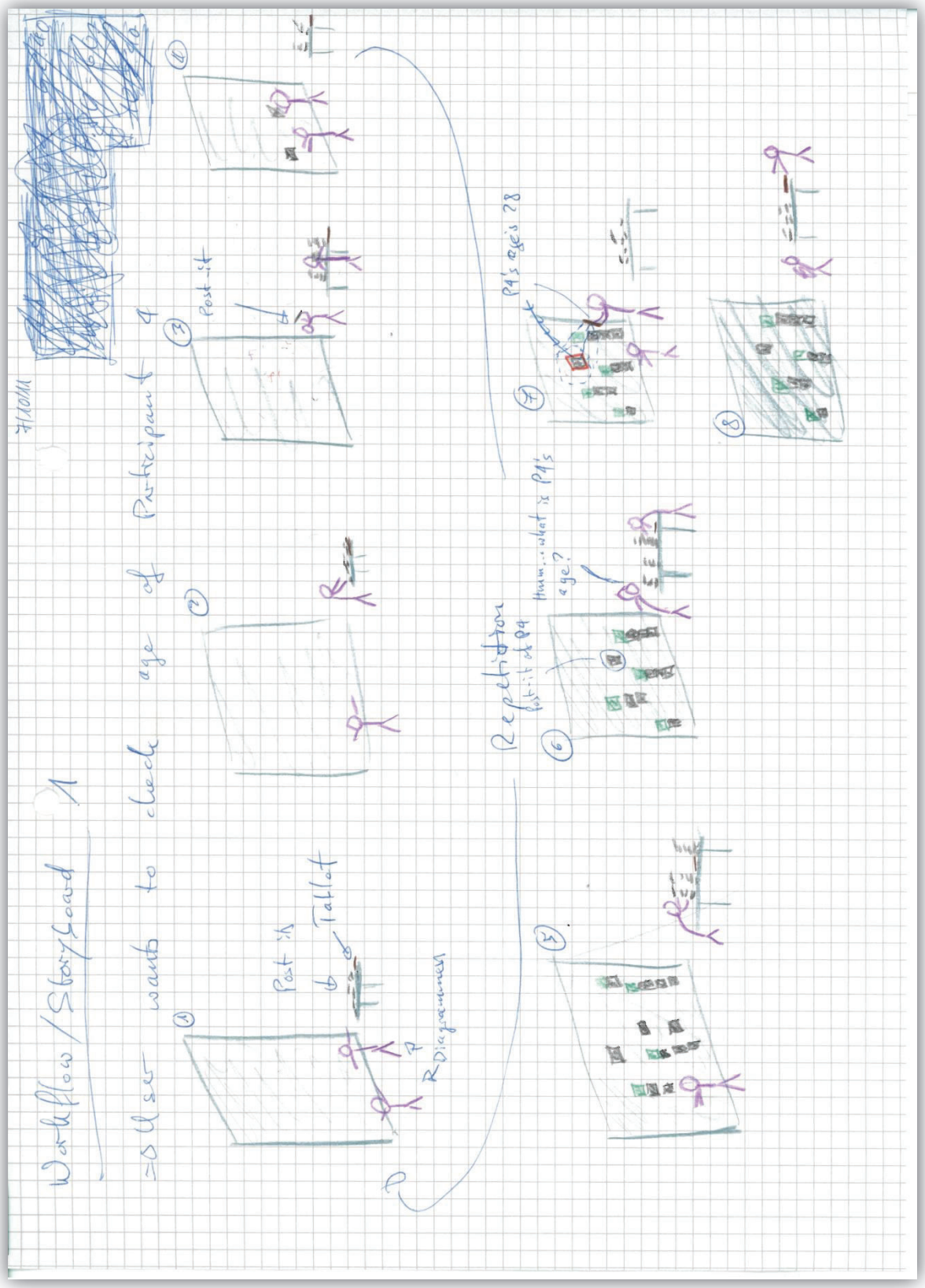


Figure 27: Workflow with the AffinityScanner

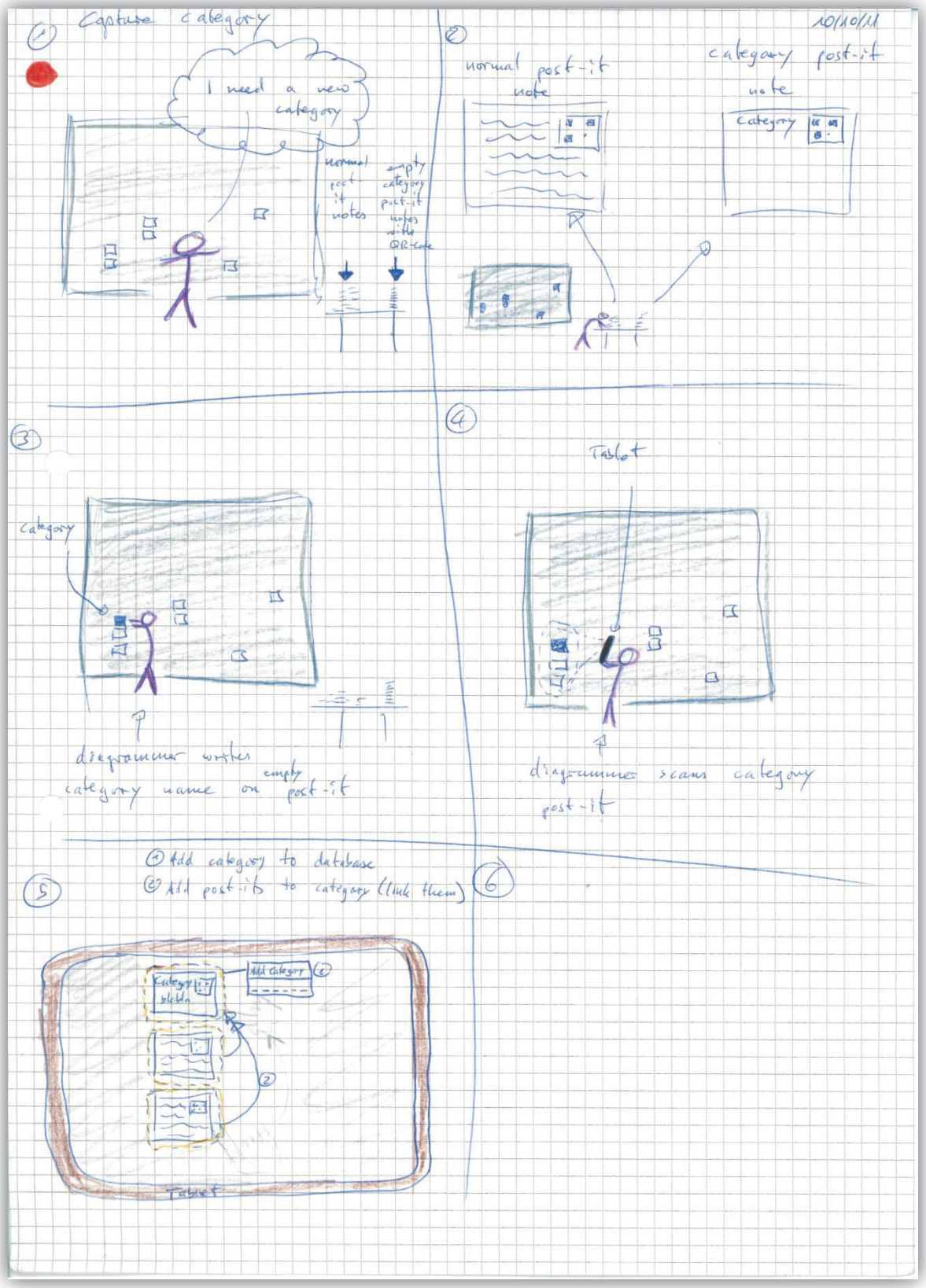


Figure 28: Storyboard about a not implemented feature described in the Future Work section to capture new categories in the affinity diagram.

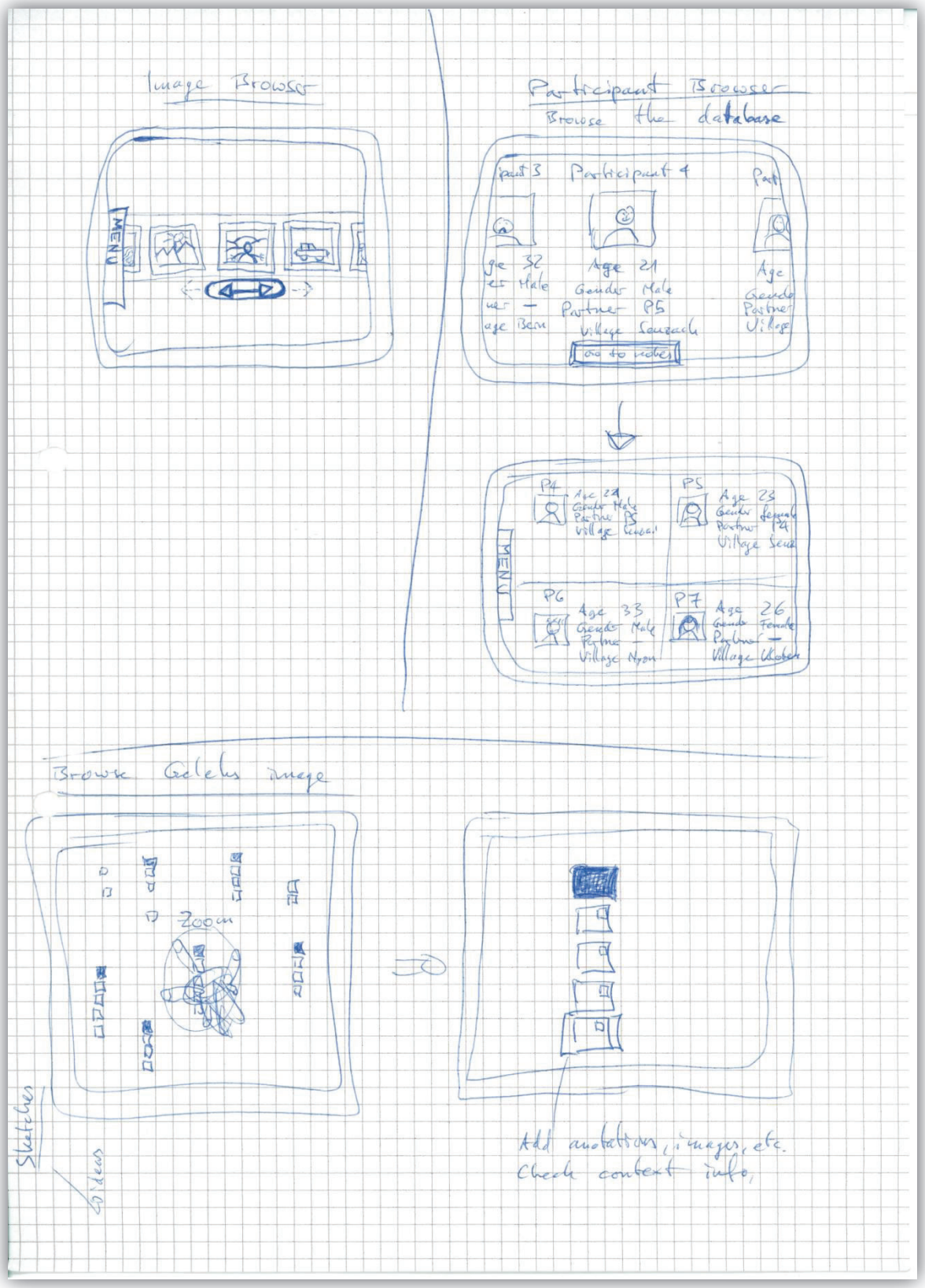


Figure 29: Upper part: Early idea of a browser for participants and images
Lower part: Interaction of the AffinityScanner with the solution from Doksam (2012)

E. Second Storyboard Iteration

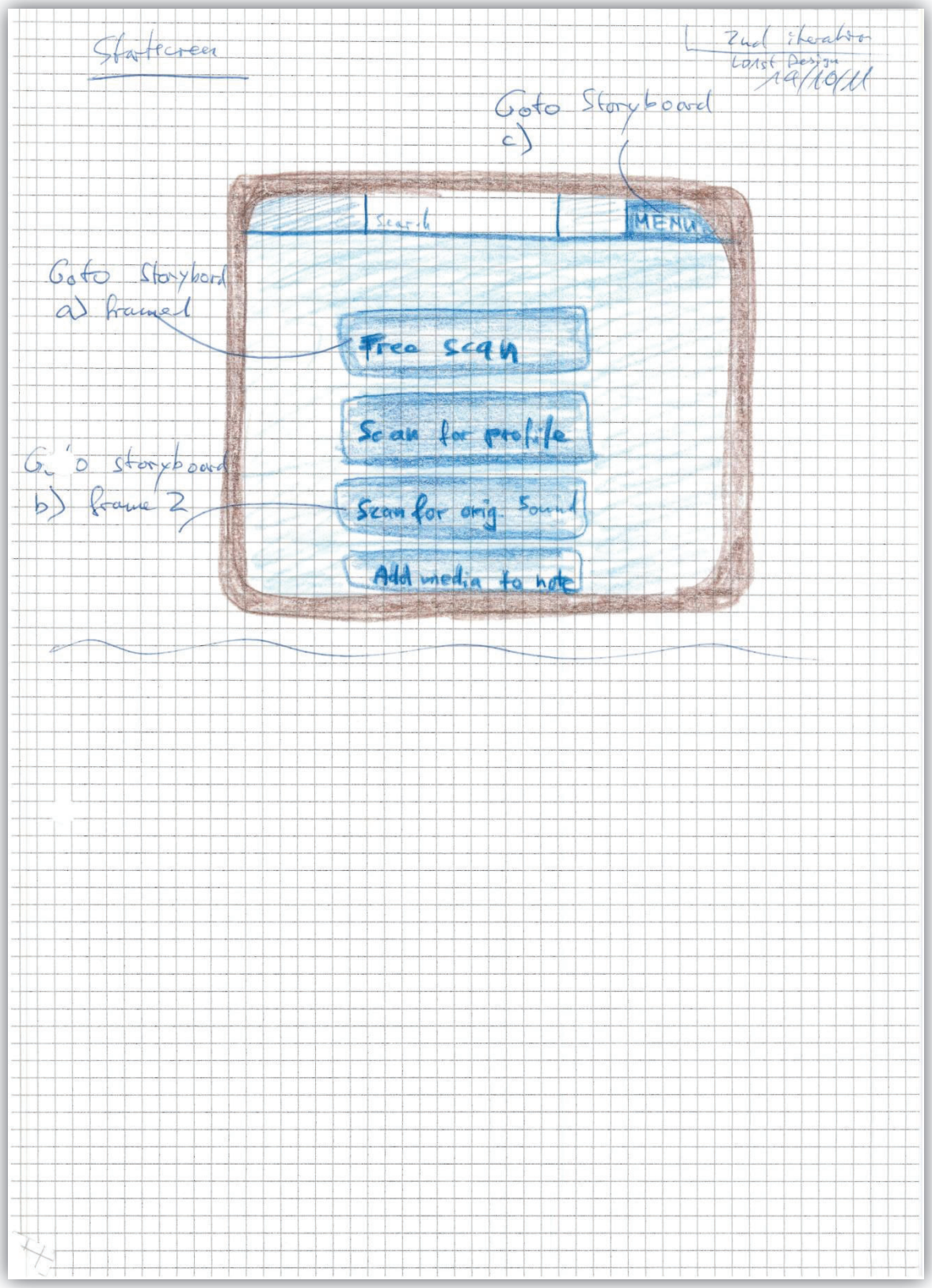


Figure 30: Interface design pointing to more specific storyboards

Figure 31: Storyboard a: Scanning the affinity wall with the AffinityScanner client

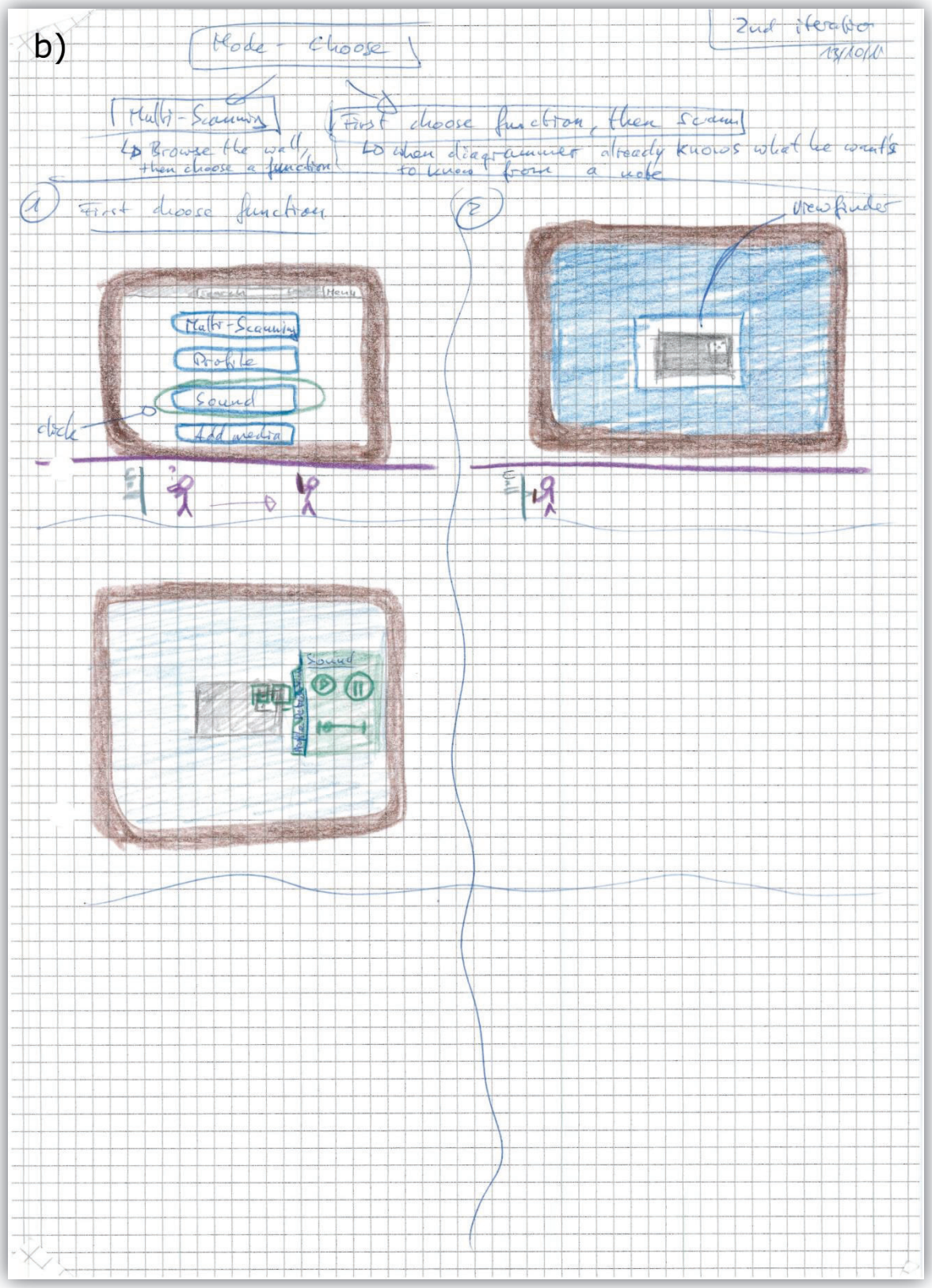


Figure 32: Storyboard b: A not implemented idea for a single note scan mode

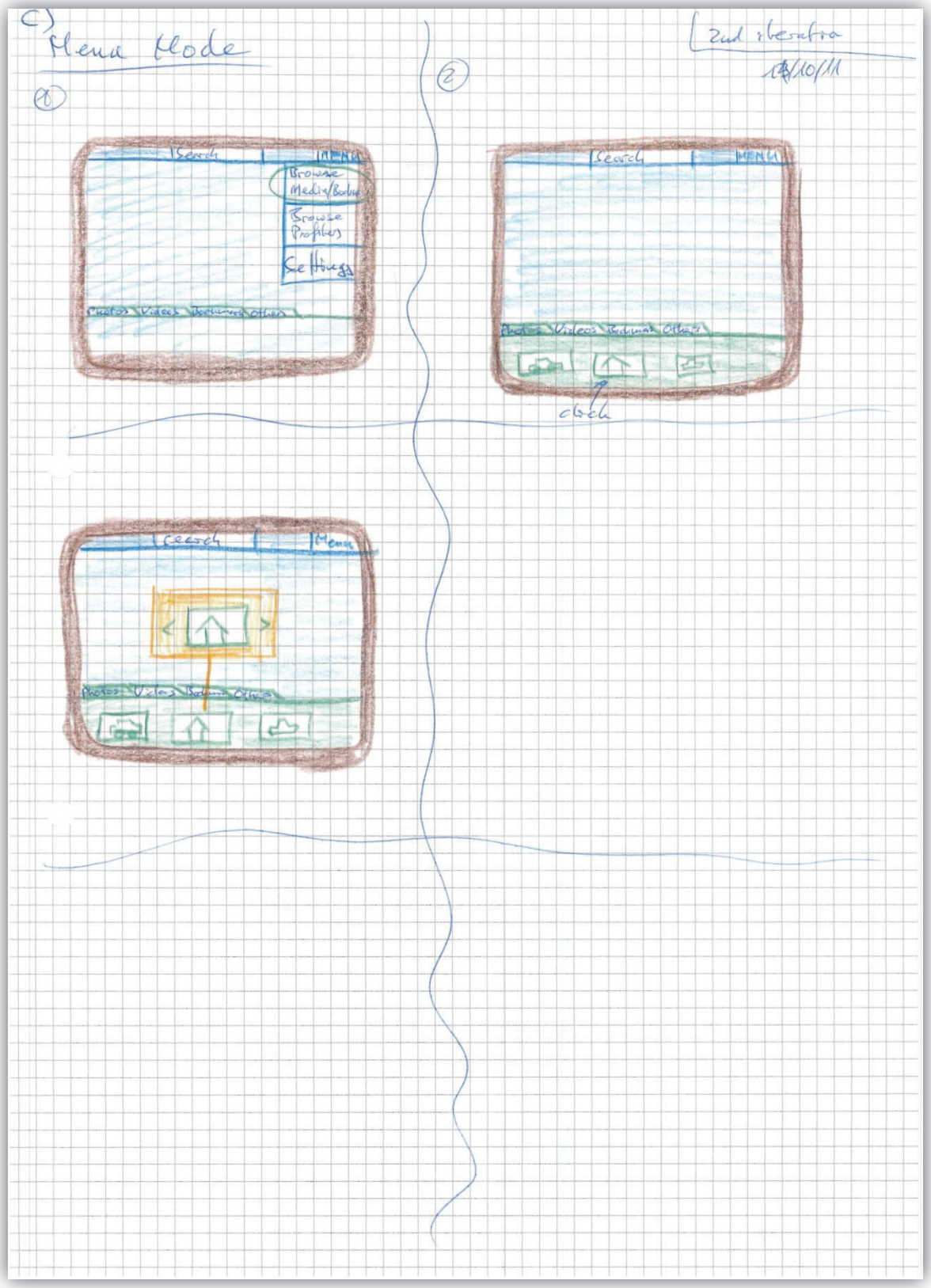


Figure 33: Storyboard c: Menu of the AffinityScanner client

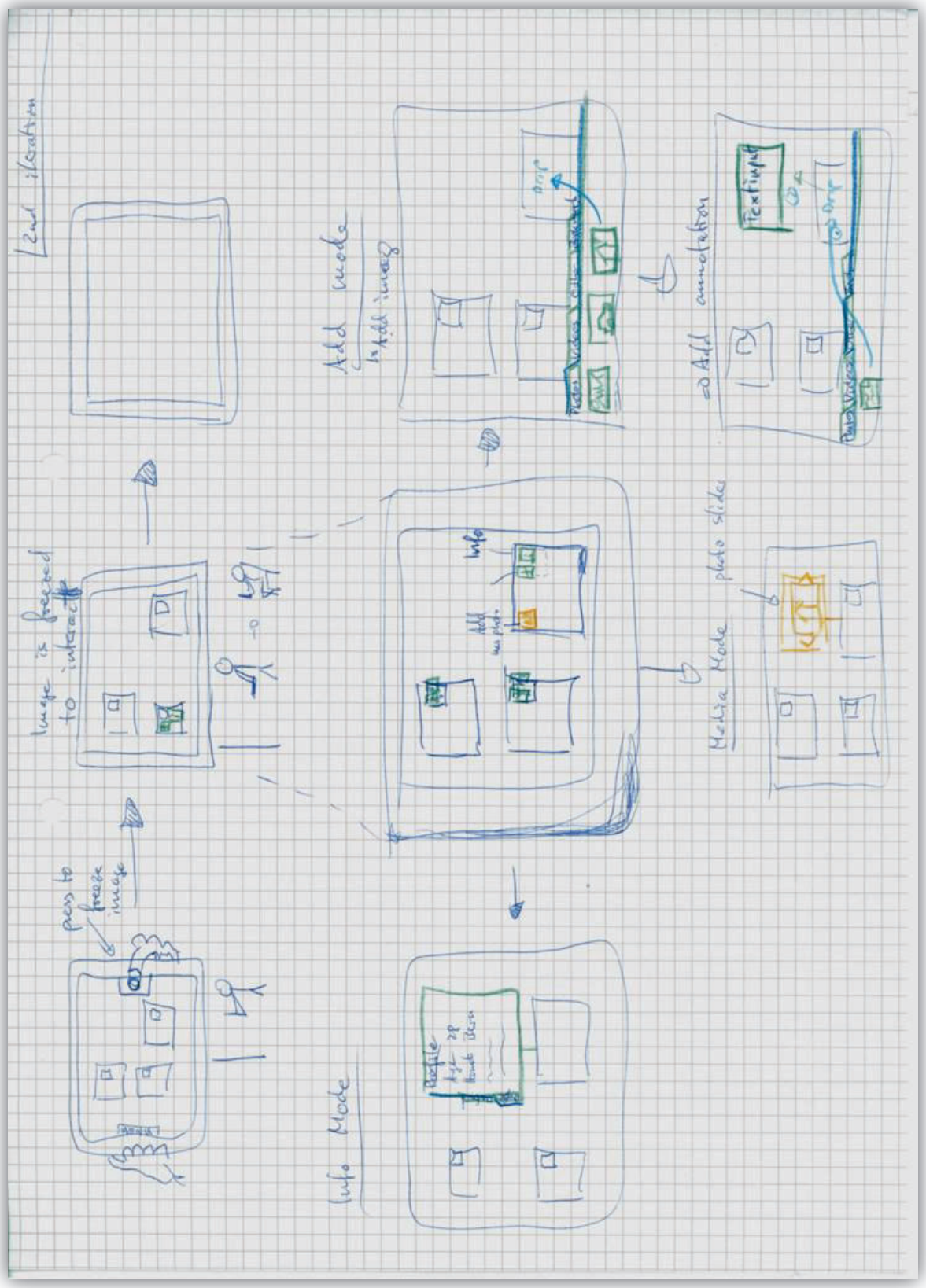


Figure 34: Workflow when using the AffinityScanner client

F. Paper Prototype User Test Task Sheet

Paper Prototype Testing Tasks
3rd round, V 1.4

Testing

- You are free to try out any possible way to fulfill a task. If an interaction/function doesn't work, I say it. Tell us when you think that you are finished with a task.
- Think aloud. Tell everything what you think.
- During the interview that was made prior to the affinity, photos have been taken of the environment to show how the location looks like to others.
- The notes you interact with are translated from German.
- Don't worry if you get stuck or do not know how to do a task. It's not you who is tested in this test, it's the system who is under test.

Tasks

You are doing an affinity diagram. During the process, you'd like to have some information about specific notes:

1. To get a better understanding of the environment the participant lives in, you want to know the age and the hometown of the participant from note number 1.
2. Look at notes that have photos attached to. Check the photos and if you find a photo of a person you know the name, add a new annotation to the note and write the name in this annotation.
3. You expect that some words in the English version are wrong in the text of note 1. You want to check the original german text.
4. You still don't get the meaning of the note as it just doesn't make sense to you. So you want to hear the original voice when the participant was interviewed.
5. Now you know what he means. You remember a photo that was taken during the interview that is related to the statement of this note (note nr. 1). It is a photo of the home office of the participant (like the picture below). Attach it to the note.

6. You want to remember note number 12 to use it later. Set it on the bookmarks list.

G. Android Prototype User Test Task Sheet

Android Prototype Testing Tasks
4th round, V 2.2

Testing

- You are free to try out any possible way to fulfill a task. Tell us when you think that you are finished with a task.
- Think aloud. Tell everything what you think.
- During the interview that was made prior to the affinity, photos have been taken of the environment to show how the location looks like to others.
- The notes you interact with are translated from German.
- Don't worry if you get stuck or do not know how to do a task. It's not you who is tested, it's the system.

Tasks

1. Find a note that's from participant number 2. Does he has a partner and when yes, who? Write the number of his partner in the annotation field of the note.

2. Check all the photos that were taken during interviews for this affinity diagram. If you find one that looks similar to the following one, add it to Note number 1.

3. Find a note with an attached picture. If you need a hint how to find one, check the annotation of note 7. If you find the note, have a look at the picture. Do you know the person? Write down the name of the person in the annotation field of the note. If you don't know the person, write your name.

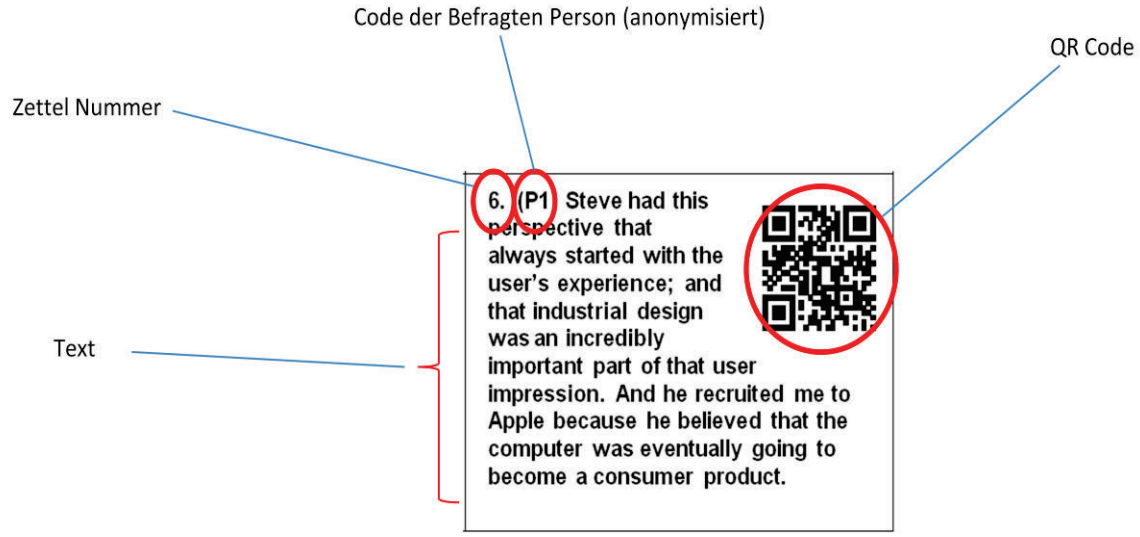
4. Check two notes that are close to another. Compare the age of the participants (interviewed persons). Now you should add a photo from the note with the older participant to the note with the younger participant.
   Hint: if the note does not have a photo attached, add one from the global picture pool.

H. Final Evaluation Task Sheet

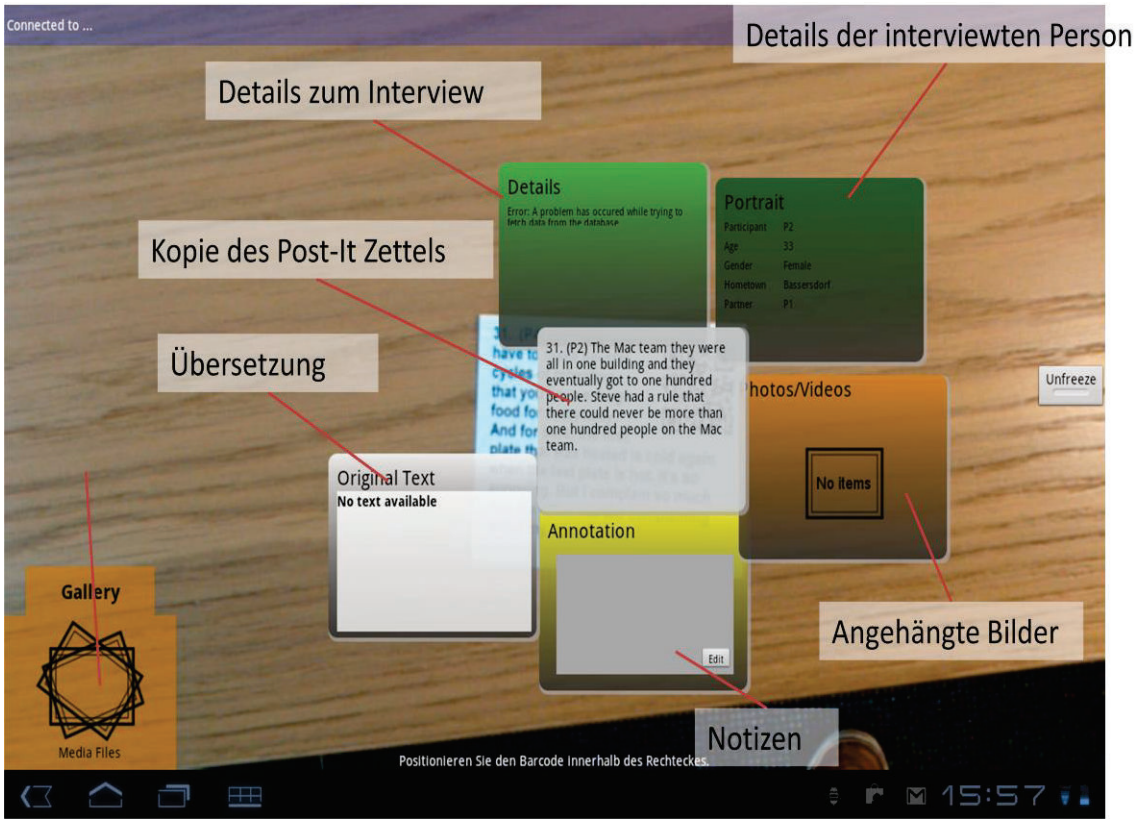
This information sheet was available to the final evaluation test users. It explains the most important elements of the test and lists the test tasks.

Übersicht

Post-It Zettel Legende



Screenshot der App



Vorgehen beim erstellen

1. Post-It Zettel sind unsortiert auf dem Tisch
2. Alle Zettel nacheinander an die Wand kleben und dabei ähnliche Aussagen beieinander positionieren
3. Zettel die bereits an der Wand sind können auch umgeklebt werden um andere Gruppen zu bilden
4. Gruppen sollten so gebildet werden, dass damit die Fragestellungen beantwortet werden können.
5. Wenn Informationen auf den Post-It Zetteln fehlen, können mit dem Tablet weitere Informationen abgerufen werden (z.B. Fotos, Notizen, Übersetzung)
6. Zeit: 40 Minuten um alle 4 Fragen zu beantworten

Fragestellung

1. Welche Probleme sehen die Befragten beim Mittagessen?

2. Was finden die Befragten positiv am gemeinsamen Mittagessen?

3. Was sind die Lieblingsspeisen der einzelnen Befragten?

4. Verbesserungsvorschläge für das gemeinsame Mittagessen?

Antworten

1.

2.

3.

4.

I. Consent Form Final Evaluation

Einverständniserklärung

Datenerfassung

Während des Tests werden Daten gesammelt in Form von handgeschriebenen Notizen. Die Daten sind hauptsächlich Beobachtungen über die Interaktionen mit der getesteten App.

Während der abschliessenden Feedbackrunde werden Daten mittels Sprachaufzeichnung gesammelt. Die Sprachaufzeichnung dient zur Analyse des gesprochenen Feedbacks.

Vertraulichkeit

Die während dem Test gesammelten Daten werden von Forschern der ZPAC Forschungsgruppe der Universität Zürich analysiert. Falls die Daten aussenständigen Personen gezeigt werden, werden sie immer anonymisiert sein.

Teilnahme

Die Teilnahme an diesem Test ist freiwillig und der Test kann jederzeit abgebrochen werden.

Ich habe die Einverständniserklärung gelesen und verstanden:

Name _____

Unterschrift _____

Ort und Datum _____

J. Detailed Design Decisions

The design decisions for the paper prototypes and the test results of the following user tests are described in detail here:

First design

Design decisions

Menu depth:	The design critique of the storyboards has shown that deep menu structures should be omitted where possible. Clicking through 2 or 3 menu levels makes the process slow and unhandy for the user. The solution is a concentration of two interaction categories that are directly tied to the affinity note to interact with the note and leaving away the note-menu. Through this step, the functions are accessible more directly, with fewer steps to do. Under these two interaction categories, the more specialized functions can be accessed. The two interaction categories are « Enrich notes » and « Context information on notes ». They are represented in the UI with an ADD button (+) and an INFO button (i).
Constraining the interaction possibilities:	As discussed in (Norman, 2002), constraining the interaction possibilities of a system makes it easier to understand it for users because it reduces the cognitive load. The user interface respects that by giving only two interaction possibilities with an affinity note. An add-button and an information-button. This helps the user find needed functions, as he only has to search for the function in these two categories.
Design Principles:	Taken from the book Human-Computer Interaction (Dix et al., 2003), the user interface has been tried to make operable as simple as possible. The design emphasized on design principles that help using the application right from the beginning even for new users (in the book defined as the category “Learnability”). That means the design principles **‘**Consistency’ and ‘Generalizability’ were used as background when developing the design of the user interface. The interaction in one interaction-category is always the same. • Enrich notes: The content is always added with drag and drop from an item container at the bottom of the screen, also the

	annotations and bookmarks. This makes it always similar to use and is also less effort to implement, as you only have to implement one window control element (reusability of code). • Context information: All information is displayed in the same window, changeable with tabs.
Freeze Mode:	As the tablet is quite heavy, it is not very convenient to interact with it while pointing to the wall. So a freeze mode was introduced, where one can freeze the actual image of the wall and then lower the tablet to interact with the app.

User test results

Global Add Button vs. Local Add Button:	In this prototype whenever the user pressed the ADD-button, a vertical slider at the bottom of the interface came up that displayed different types of elements. Users were confused when they clicked the ADD-button on a note, because they did not know if the displayed items in the item slider (photos, videos, bookmarks, annotations) were only items related to the clicked note (local) or all items in the database (global). This is indeed inconsistent, as the photos and videos displayed are all that are in the database, while the bookmarks and annotations are only the ones that are related to the note where the add button was clicked. Also when you drag and drop bookmarks or annotations to another note than the note that initiated the interaction (the note where you clicked the ADD button), the result is not defined. A user suggested adding a global ADD button where you can access all the items that are in the database and local ADD buttons on the notes that show only the related items.
Design changes:	During one test, the test user had the idea of a new interaction concept, where the buttons on the note are removed and the note as a whole is clickable (like in the internet, where the hole abstract of an article can be clicked to access the article, and not only the title). Interaction possibilities then would also include showing the translated text directly on the note over the original text. Instead of the buttons a menu would appear when the note is clicked with all functions

	accessible on the uppermost level. Or alternatively the window where the information was displayed now would show also the other stuff that was accessible via the add button. But only the related items. Other items would then be addable by clicking on the global add button (or dragging up the item slider from the bottom of the screen), that brings up the item slider, like it was used now, which has all the database content in it. Annotations (and maybe bookmarks) should be addable directly on the note, as the dragging does not make sense for the user when input has to be provided afterwards. Maybe an editable text input field could be open and accessible all the time when a note is selected, so that the user can enter text whenever he likes.
General test outcome:	In general, the two test users had no major problems with the design. So the design seemed to be good. Nevertheless, the flaws found were addressed in the second design.

Second design

Design Decisions

Making the note clickable:	Test users from the test run with the first prototype had the idea, to make the whole note clickable, so that we can remove the two buttons on the note. This would avoid the problem that the buttons are too small to be clicked or do not fit on the note when the user is far away of the wall and the notes are therefore small. **Appearance:** We had a look at how buttons are painted, as buttons are always perceived (when they are well designed) as clickable. Buttons almost always have a shadow that suggests to the user that it is hovering over the underling surface and is therefore clickable. So we added also a shadow to the note, so that the user sees that it is clickable and clicks it.
Menu:	The menu is still without needless hierarchy depth. But it has been changed to a menu tree that opens when the note is clicked. This should make a less desktop like user interface, what has been argued in the first prototype.

Item pool:	The item pool/slider has been added statically to the Interface, but collapsed at the bottom of the screen, so that only the tabs can be seen. It can be activated by sliding up the tabs or clicking on a tab (it then slides up automatically). The content is now only global stuff and nothing related to a note. It has been found as a flaw in the first design that in this slider global and local content is mixed (global photos and videos, local bookmarks and annotations). It shows now all bookmarks of the app and all annotations that are stored in the database.
Local/Global:	A big issue with the first prototype was that the users had problems with content that is available to all notes like photos and movies in the database (global) and content that is related to specific notes like bookmarks and annotations of a note (local). Users found it, for example, strange that they can add photos from the photo pool (all photos in the database) with the local add-button on the note. They expected something like a global add function (a global add button was denied by the users, but it should be something similar to the global add button). We try to address this in this new design with the global item pool at the bottom of the screen and a menu on the note that has only local functions and icons on the note that show locally attached items like photos and videos (when clicked, they show only the related items, not the global icons).

User test results

Global/Local Problem:	The user expected the item pool (photo/video/bookmark/annotations) to be local to the note. Later he realized that it is global (because he saw a lot of photos in the photo tab and concluded that these are all photos available). After that, he had no problem to change his behavior.
Data pane focused:	The user was totally focused on the data pane (the window that comes up after clicking on a note where all information of the note is displayed). He only interacted with this pane and did not interact with the note, not even with the symbols and buttons on the note. For him, the data pane was the (digital) representation of the note, and the note

	was of no importance anymore when the data pane was shown.
Buttons not recognized:	This problem is probably because of the bad button design in the prototype. Most of the buttons had not been recognized as buttons and were therefore not used.
Comparison of two language versions:	The user had the desire to have the two different language versions of the note side by side, so that he can compare the differences.
Translation button as language settings:	The button to show the original German text was thought to have the function to change the language of the application by the user. Like a settings button to change the overall language.
Drag and Drop anywhere:	A strong desire, also observed with the previous prototype, is to make drag and drop of photos (and probably other elements too) possible to all areas that are connected to the note, instead of only letting them drop on the note directly. So for example if the data pane in the audio tab is open, one should be able to drop a photo on this pane and the photo should get attached to the note.

Third design

Design decisions

Exploit the whole display - not only small parts of it - to display information:	Information cards are movable over the whole display, allowing placing them wherever it suits best on the display. Photos are also movable over the whole display, allowing positioning them in parts of the display where no information cards are. So photos and other information can be checked and compared at the same time.
Make a new non-desktop computer interaction feeling with the app for the user, moving away from the classic click-and-display scheme of a desktop application:	In the last designs, the interactions were pretty similar to desktop applications, where the information is displayed in a frame or window and the user has to click on a tab to display other information. The possibilities of the touch screen of the tablet have not been used. In the new design, information cards and photos can be moved around on the screen with a finger and photos can be zoomed in by pinch gestures.
Make multiple information types visible at the same time to facilitate	In prior tests, users complained that it is difficult to compare information, because the information was not visible at the same time and users had to switch between views that showed the different

comparison:	information. To address this issue, in the new interface all information is displayed at the same time. Unneeded information can be moved away from needed information to avoid distraction.
Reduce the needed clicks to display information (one click on the note displays all available information):	To make the control of the application more convenient, the user only needs one click on the note to display most of the information.
Make a clear distinction between local content and global content:	To support the clear distinction, the new interface has now two similar media pools called 'Galleries'. One is local for the note as an information card and one is global for the whole affinity diagram in the lower left corner of the interface. We hoped that this would make clear that the content in the note gallery is attached to the note and the content in the global gallery is the whole content of the database, because they look similar, but are at different positions in the interface.

7.2.1.1.1 User test results

Close button on the note:	Users would have preferred to have a stack button on the note to put all information cards back behind the affinity note into the card stack instead of clicking anywhere on a uncovered part of the screen.
Trash can:	To make photos and bookmarks removable from notes, a trash can, where the user can put the element he wants to remove, have been proposed.
Info cards scalable:	To make content better readable, a user had the idea to make the information cards scalable with a pinch gesture.
Information card titles not visible:	A special layout algorithm should be used to lay out the information cards around the note. Otherwise the titles of some cards are covered by the note and the user does not see what the content is.

Android Prototype

Design decisions

The Android prototype used the design from the 3rd prototype with minor changes.

User test results

Keyboard comes up unintentionally:	The Android operating system uses a soft keyboard on the screen to input text. This keyboard is hidden when not used, but slides up whenever a user clicks into a text input box. We designed the annotation field as an input box to enable the user to type in remarks at any time without clicking an edit button. Unfortunately, users often clicked into the input field unintentionally and triggered the keyboard thereby. This was an annoying problem that was addressed in the final implementation.
Media Gallery:	The local-global problem still occurred. Two users were unsure if the photos in the media gallery are related to the affinity notes that are on the screen at a given moment or to the whole affinity diagram. A solution would be to gray out the screen when the photos from the media gallery are displayed to make clear that they do not belong to any other element on the screen. This issue has not been addressed in the final implementation, because the proposed solution would have conflicted with the attaching function of media items via drag and drop, as the affinity note to attach it would be behind the gray layer.
Information card to front when touched:	In this design, it was not possible to move information cards over the affinity note. The affinity note was always on the upper most level of the screen. This is annoying if an information card is covered by the affinity note and has to be moved outside the note to be fully readable. Users therefore suggested bringing an information card to front when it is clicked. This issue has been addressed in the final implementation.
Multiple affinity notes should be able to be open:	In this implementation, an open affinity note was stacked again if another note was opened. The idea behind it was to prevent a mess of information cards on the screen. But it also makes it harder to compare information of different affinity notes, because only the information of one note can be read at a time. So the idea was, to let multiple notes be open, but to limit the distance the information cards have from their

	affinity note. With this limitation, a mess should be avoidable. Unfortunately, this issue was not addressed due to lack of time.
Improve QR Code recognition rate:	As already known before the user test, the recognition rate of the QR Codes and therefore the recognition rate of the affinity notes on the wall was not very good. During the design critique session, a new idea came up how to improve it. In the current prototype, the app only analyzed the last picture from the camera and displayed the found notes on the screen. The idea was now to also store found affinity notes of the last two seconds and display them, even if they were not recognized anymore. This issue was successfully addressed in the final implementation.
Annotation and media card hardly movable:	Users had problems to move the annotation and the media card. The annotation card was only movable if the user pressed the finger on the title of the card and moved it then, because the rest of the card was occupied by the text input field, which triggered the keyboard to slide up when pressed. The media card had two big buttons to show the photos and the videos. When a user pressed the finger there to move the card around, nothing happened because the system thought he wanted to press the button. As the buttons were so big, most users did not find out where to press the finger to move the card around. These issues were addressed in the final implementation.

K. Final Evaluation Data

This part shows the observations and the transcribed user feedback from the final evaluation. The user feedback is a summary of what all users of a group said about a topic.

General Observations

- All groups were able to solve the tasks and give the right answers to the questions given as tasks.

4th Test on 9.2.12 10:00

Test users

- User 9, 24, Student
- User 10, 25, Student
- User 11, 30, Worker
- User 10 and User 11 knew each other already

Observations

- Photo issue: When a user clicks on the photo stack of the Media card, the photos are not displayed, but the card is brought to front. The user has to click a second time on the photo stack to display the photos. Most users did not click a second time, but thought that there are no photos. After a while some users found the feature coincidentally.
- In the beginning the tablet was grabbed by User 10 and extensively used by him. The rest did not use the tablet in the beginning. After a while, the others also tried out the tablet. But User 10 was still the primary user.
- The tablet was often used by all three users to check information. One user scanned the note and all three checked the resulting information on the screen.
- Users wanted to maximize the photos (with double-click and pinching gesture), but feature was not implemented.
- The tablet was used right from the beginning by User 10. This is unlike other test runs, where user neglected to use the tablet in the beginning but first put all notes on the wall before checking them with the tablet. Some only checked them with the tablet when they had to answer the questions and therefore had to find out missing information.
- The users (especially User 10) used the app extensively. But unlike in the other tests, in the introduction to the test it has been stressed to use the app whenever they feel that they need more information.
- When User 10 scanned a part of the wall with a lot of notes and not all notes were recognized, he clicked on the unrecognized notes to make them been recognized. He later explained in the feedback round, that he thinks it would be a nice feature if he could click on an unrecognized note and the software would scan this part of the image again and tries to find a QR code there.
- A problem found is, that QR Codes in the left part of the screen are better recognized than in the right part.
- Unlike other groups, this group was really eager to use the tablet.

Feedback round

1. Were there problems handling the tablet application?
    a. Users would like to be able to maximize the photos. With double-click or pinch.
    b. Font of affinity notes was too small to read. Would be nice to be able to zoom the text.
    c. Users put similar notes very close to each other on the wall so that they overlapped. As a consequence, the notes on the tablet screen overlapped also and when the information cards were expanded, the other notes were no more visible. The users complained about that. A suggestion was to make the digital notes movable on the display to move overlapping notes away.
    d. When notes are at the border of the screen, not all information cards are visible. They should be arranged so that they are all visible
    e. It's not easy visible if there are images in the image-stack-button of the media card, because it's not clear to the user that there are images (it's only clear that there aren't any images when the button is labeled "no items", but not the other way round). The user suggestion was to either make a preview of the photos in the image-stack (make the real images part of the button) or to throw out the images on the display as soon as the information cards are expanded, making it needless to click on the media card image-stack-button.
    f. Photos are laid out randomly on the screen. Better lay them out near the media card and close to each other (maybe even in a vertical line)
2. When did you use the tablet application?
    a. Used the tablet app when there were missing information
    b. Also used the app when they didn't know how something looks like (like crostini)
3. Was there something that prevented you from using the tablet application?
    a. Sometimes they did not use the tablet because all information was on the note
4. Was it a problem that there was only one tablet device available for all participants?
    a. The users think that one tablet for 3 persons is enough (under these circumstances in the test where not all notes were unusable without the tablet)
5. Which functions of the tablet app did you primarily use?
    a. Photos
    b. Annotations were only used once
6. Additional comments:
    a. User 9 didn't like the layout of the application. The user thinks it looks like school. The users thinks the space is not used very well, because if you scan only one note from some distance, the note is very small on the screen and around it there is a lot of unused space. And the information cards are very close to the note and you have to click on it to see everything (because the rest is behind the note in the beginning). So the user suggests that the info cards are farer away from the note, to use the space better. And instead of only titles on the information cards, nice icons would be more appealing.
    b. The position of the information cards in respect to the note should be preserved, so that they can be laid out again in the same position the next time the same note is displayed.

Design Suggestions

- What to do with notes at the border. When notes are at the border and not totally visible, users complained that they cannot read them. A possibility would be to shift the digital note automatically into the screen.
- Use information card layout algorithm that respects screen borders.
- User suggestion: either make a preview of the photos in the image stack (make the real images part of the button) or throw out the images on the display as soon as the info cards are expanded, making it needless to click on the media card's image-stack-button.
- Photos are laid out randomly on the screen. Better lay them out near the media card and close to each other (maybe even in a vertical line).
- Don't display information cards that have no data. E.g. when there are no photos or no translation, don't display these cards.
- Place information cards farer away from the note so that everything is readable directly without moving the cards around.
- The position of the information cards in respect to the note should be preserved, so that they can be laid out again in the same position the next time the same note is displayed.

3th Test on 8.2.12 14:00

Test users

- User 6, 19, Student
- User 7, 21, Student
- User 8, 20, High-school graduate

Observations

- In the beginning, nobody used the tablet. They first put all notes on the wall without using the tablet. Later, when all notes were on the wall, they used the tablet.
- When they later used the tablet, User 6 was afraid of the tablet and only tried to use it once. After a short try-out, the user gave the tablet to someone else, frustrated.
- The photo issue (users have to click twice on the image-stack-button on the media card to display the photos) was not a problem in this test run, but the issue was explained in the introduction to the test.
- After the try-out of User 6, only User 7 used the tablet. User 8 never used the tablet.
- All participants were quite shy. So there was no leader.
- When they tried to answer the questions, the app was used extensively. But still only by User 7.
- Interesting: When they found the photo of a spoon, they rearranged the corresponding note because it got a different meaning. This shows how context information provided by the tablet app can change the ordering of the notes.

Feedback round

1. Were there problems handling the tablet application?
    a. App recognizes notes in the left side of the image better.
    b. Information cards are sometimes outside of the screen.

   c. Images are laid out randomly on the screen. That is not good.
   d. App works good, is fast and intuitive, according to User 7.
   e. User 6 explains why she was afraid of the app. She wanted to unfreeze the image and searched for a back button like on the iPhone. As she didn't find one, she didn't know what to do.
   f. User 8 didn't use the app, so he couldn't tell about problems.
2. When did you use the tablet application?
   a. App was used when they didn't know how the favorite meals look like (they watched at photos) or when they were unsure to what a note belongs to, e.g. if the text on the note is meant positive or negative, they needed more information
3. Was there something that prevented you from using the tablet application?
   a. No comment
4. Was it a problem that there was only one tablet device available for all participants?
   a. More tablets would have sped up the process.
   b. Users think 2 tablets would be enough, because one user has to do the writing stuff.
5. Which functions of the tablet app did you primarily use?
   a. Mainly the photos were used.
   b. Translation was also used once (but English was easy to understand in general).

Design Suggestions

- Make layout algorithm that respects screen borders for info cards.
- Make photos not laid out randomly on the screen.

2th Test on 6.2.12 14:00

Test users

- User 3, 43, Student
- User 4, 27, Student
- User 5, 23, Student

Observations

- When user clicks on the image-stack-button of the media card, the photos are not displayed, but the card is brought to front. The user has to click a second time on the photo stack to display the photos. Most users did not click a second time, but thought that there are no photos. After a while some users found the feature coincidentally.
- Use of the app was very good in general. The test users had no problem using it. That means the design is good!
- In the first phase, when the test users put the notes on the wall, the app was used individually. Later, when the users tried to answer the questions, the app was heavily used by all three together.

Feedback round

1. Were there problems handling the tablet application?
    a. Test users reported problems with the tablet handling, because the border is so thin that you easily touch the screen unnoticed. Clicks are not recognized when the screen is already touched at another position (that's a general multitouch screen problem).
    b. Note recognizing takes very much time sometimes.
    c. Would be better if it would be possible to work without freezing the screen.
2. When did you use the tablet application?
    a. When something was unclear.
3. Was there something that prevented you from using the tablet application?
    a. No. But when things were more or less clear, the tablet was not used. So it was only used when there was real missing information.
4. Was it a problem that there was only one tablet device available for all participants?
    a. No.
5. Which functions of the tablet app did you primarily use?
    a. Photos.
    b. Annotations were practically not discovered that there are some.
6. Additional comments
    a. App is well-arranged (übersichtlich).

Design Suggestions

- Future Work: Make freezing needless. The recognizing should be so good that the notes on the tablet screen can be used without freezing. Freezing seems to be annoying for quick lookups of information.

1th Test on 6.2.12 10:00

Test users

- User 1, 18, scholar
- User 2, 42, worker
- Mother-daughter

Observations

- App is not used in the beginning.
- App is used for German translation (e.g. dishwasher).
- App is used to scan notes on the table and to scan notes on the wall.
- App was often used collaboratively.
- App is mainly used for one note at a time.
- Normally the note was scanned on the table before put on the wall.

Feedback round

1. Were there problems handling the tablet application?
    a. User comment: "I once clicked at the wrong position and then I didn't know what happens. That was a bit difficult. But the rest is nice."

2. When did you use the tablet application?
    a. User comment: “Translations. And also the images. When we were not able to imagine what they meant by for example “Schweizer Küche” (Swiss food) we could check the photos of the Fondue.”
3. Was there something that prevented you from using the tablet application?
    a. User comment: “When the note text was clear, I did not go for the tablet.”
4. Was it a problem that there was only one tablet device available for all participants?
    a. Was not asked!
5. Which functions of the tablet app did you primarily use?
    a. Translation (used most frequently).
    b. Photos.
6. Additional comments
    a. User comment: “What I like at the app is that you have more information like who has said what, and that you can make an annotation to the note.”
    b. User comment: “What I like is that you can see the translation of the English text, and that you also have the additional information like the photos, who said it and annotations.”
    c. User comment: “The good thing is that you have all the stuff directly at hand.”

L. Usability Aspect Report

This report lists the issues found in the final user test with a rating for the severity. The Usability Aspect Report method is taken from http://vedantmehta.com/Presentations/HE_Bugzilla.pdf.

Legend

Severity Criteria
The **frequency** with which the problem occurs: Is it common or rare?
The **impact** of the problem if it occurs: Will it be easy or difficult for the users to overcome?
The **persistence** of the problem: Is it a one-time problem that users can overcome once they know about it or will users repeatedly be bothered by the problem?

Severity Levels
0 = I don't agree that this is a usability problem at all
1 = Cosmetic problem only: need not be fixed unless extra time is available on project
2 = Minor usability problem: fixing this should be given low priority
3 = Major usability problem: important to fix, so should be given high priority
4 = Usability catastrophe: imperative to fix this before a new version can be released

Issues

Number:	1
Description:	Users would like to be able to maximize the photos. With doubleclick or pinch.
Explanation:	Photos can only be seen as thumbnails even if they are bigger in original. That's an inconvenience for the user. They should be zoomable.
Severity:	**Rating=2**; Frequency=(3) always; Impact=(1) low, user can continue work; Persistence=(4) No way to overcome

Number:	2
Description:	Font of affinity notes was too small to read. Would be nice to be able to zoom in.
Explanation:	The size of the digital affinity notes is dynamic and depend on the distance to the wall. If the digital affinity note is small, it is sometimes hard to read the text. It would be nice to have a zoom function for the text.
Severity:	**Rating=2**; Frequency=(2) often; Impact=(1) low, user can continue work; Persistence=(2) User can move closer to the wall to make affinity note bigger

Number:	3
Description:	Overlapping notes on the wall make a mess on the tablet screen
Explanation:	Users put similar notes very close to each other on the wall so that they overlapped. As a consequence, the notes on the tablet screen overlapped also and when the information cards were expanded, the other notes were no more visible. The user complained about that. A suggestion was to make the notes movable on the display to, move those notes away.
Severity:	**Rating=1**; Frequency=(1) seldom; Impact=(1) low, user can continue work; Persistence=(1) User can move physical notes on the wall to prevent mess

Number:	4
Description:	When notes are at the border of the screen, the information cards are not all visible.
Explanation:	If an affinity note is at the border of the screen, they information cards that are layed out around the note are not all visible, because some of them are off the screen. It would be better to have an algorithm that detects the problem and lays out the information cards better
Severity:	**Rating=3**; Frequency=(3) often Impact=(3) medium, usability is degraded; Persistence=(2) User can position affinity note in the middle of the screen before scanning

Number:	5
Description:	Some users did not see that an affinity note has images attached.
Explanation:	It's not good visible if there are images in the stack of the media card, because it's not clear to the user that there are images (it's only clear that there aren't any images when the button is labeled "no items", but not the other way round). The user suggestion was to either make a preview of the photos in the image stack (make the real images part of the button) or to throw out the images on the display as soon as the info cards are expanded, making it needless to click on the media card image stack button.
Severity:	**Rating=2**; Frequency=(3) often Impact=(3) medium; Persistence=(1) Only a problem for new users

Number:	6
Description:	Photos are layed out randomly on screen.
Explanation:	When a user wants to display photos that are attached to an affinity note, the photos are thrown on the screen in random positions. It would be better to lay them out near the media card and close to each other (maybe even in a vertical line)
Severity:	**Rating=3**; Frequency=(4) all the time Impact=(2) user can work with it, but usability is degraded Persistence=(4) no way to avoid problem

Number:	7
Description:	Position of information cards in respect to the affinity note are not preserved
Explanation:	The position of the infocards in respect to the note should be preserved, so that they can be layed out again in the same position the next time this note is displayed.
Severity:	**Rating=3**; Frequency=(4) all the time Impact=(2) user can work with it, but usability is degraded Persistence=(4) no way to avoid problem

Number:	8
Description:	To interact with the digital affinity note on the screen, the user has to freeze the scanning image. Users would prefer direct interaction.
Explanation:	Would be better if it would be possible to work without freezing the screen.
Severity:	**Rating=3**; Frequency=(4) all the time Impact=(2) user can work with it, but usability is degraded Persistence=(4) no way to avoid problem

Number:	9
Description:	Do not display information that is not available
Explanation:	Information cards that show no information should be hidden (e.g. original text card when no translation is available, or media card with no media)
Severity:	**Rating=3**; Frequency=(3)often; Impact=(2) user can work with it, but usability is degraded; Persistence=(4) no way to avoid problem

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used any other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

This thesis has never been used for graduation at any academic institution.

Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Die vorliegende Arbeit wurde an keiner Hochschule zur Erlangung eines akademischen Grades verwendet.

Date Signature