Where's the note?

A Multimodal System to Augment Affinity Diagrams

Masterthesis

supervised by

Prof Elaine Huang Ph. D.

and Gunnar Harboe

Department of Informatics

University of Zurich

to obtain the degree of

"Master of Science in Informatics"

Author:	Gelek Doksam
Course of Studies:	Wirtschaftsinformatik
Student ID:	05-708-094
Address:	Neunbrunnenstrasse 86
	8050 Zurich
E-Mail:	gelek@doksam.ch
Closing date:	31.01.2012

Abstract

Qualitative data analysis is an important tool to get high level feedback from contextual data gathering methods. One popular method is Affinity Diagramming. While this method relies on a paper approach where data is written on notes there are disadvantages in ordering and identifying notes once their quantity rises. In this thesis we describe a prototype that will help identify these notes on an Affinity wall through highlighting with a beamer and a smartphone. We will show that this method works as an additional layer that can be switched on or off rather than a mandatory addition. Finally we test the usability of our protoype to see that there are strong tendencies to neglect the interaction either on the wall or the Smartphone Client without a proper introduction.

Zusammenfassung

Die qualitative Analyse von Daten ist ein wichtiges Werkzeug um bei Datenerfassungen die kontextabhängig sind Rückschlüsse zu ziehen und diese zu bewerten. Eine populäre Methode dazu ist das sogenannte “Affinity Diagramming” bei der Methode als Hauptwerkzeug Papier zum Einsatz kommt um Notizzettel mit Daten darauf zu erstellen. Wenn aber die Anzahl dieser Zettel ansteigt wird es schwerer eine Übersicht zu behalten und einzelne Notizzettel zu identifizieren. In dieser Arbeit wird daher ein Prototyp beschrieben der das identifizieren von Notizzettel auf der Affinity Wand vereinfacht durch Anleuchten des entsprechenden Notitzettels mithilfe eines Beamers und eines Smartphones. Wir werden zeigen dass diese Methode als zusätzliche Schicht welche jeweils ein- oder ausgeschaltet werden kann realisiert wird und nicht eine obligatorische Anpassung des Prozesses erfordert. Abschliessend werden wir die Benutzbarkeit unseres Prototypen testen un erkennen, dass ohne ordentliche Einführung eine starke Tendenz besteht die Interaktion mit der Wand zu vernachlässigen und sich nur mit dem Smartphone zu beschäftigen.

Contents

1. Introduction

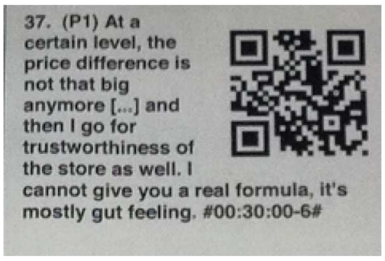
While todays digitally driven world provides a plethora of methods to quantitatively analyze data there is a lack of accountability for the analysis. Using "Stats" wrong is something that has plagued discplines such as sports and it still does today. There are so many easy ways to "quantify" data to make it look impressive that the qualitative judgement behind it is often forgotten. Or as one of the authors first statistics teachers used to say: "The value of quantitive analysis is dependent on the quality of the one doing it" Especially in a field such as HCI or Contextual Design, qualitative data analysis is important since we want to understand the user in order to assist him. Quantifiyng personal preferences and ones own judgement is something that psychology is spending a great deal of effort about (think Leikert scale) but the qualitativ analysis is still an established form to gain insights and understanding. Affinity Diagramming is a way to get data and get a shared understanding of it. It assists interpreting qualitative data and derriving design ideas from it. It uses a method relying on paper notes and the fact that they are easy to move, duplicate, edit and remove. There are however disadvantages to using paper. Affinity Diagramming is a group task, when a group of people are handling large amounts of paper notes, there are bound to be situations where person A is not aware where note 01 is. Since an Affinity wall can contain a lot of notes (see figure 2 in section 2.1) locating a note can be troublesome. Those impracticalities can be addressed with the use of present day technologies. The balancing act of addressing problems while preserving the advantages of paper is something that presents a current challenge to extending the Affinity Diagramming process.
In this thesis we will look at one way to tackle this challenge and try to address the problems. We will use a layered-approach for this where we will leave the basic Affinity Diagramming process as is. Additionally we will create a system that will not only host our solution for the presented problems but also be ready for further challenges. Section 2 will provide an short introduction into various topics that are of relevance for this thesis whereas section 3 will establish the requirements of the system and present the system. Section 4 will display the implementation process for our system and Section 5 will describe usability tests for our system. Sections 6 and 7 finally will be a discussion about our work, further opportunities and a conclusion for our thesis project.

2. Related Works

In this section we will shortly present some of the topics that influenced the design of the system described in this thesis. Affinity Diagramming is a way to analyze qualitative data while barcode recognition helps code artifacts of the data in a way that they can be machine read. Augmented Reality finally is a concept that lies in the roots of our system and the "layer"-approach of it.

2.1. Affinity Diagramming

An Affinity analysis is a method to analyze, organize and interpret qualitative data or ideas based on similarities and distinctions. It is a collaborative process and will help facilitate a shared understanding of the data. The data is usually gathered from methods that deliver unstructured qualitative data, e.g. interviews. An interview is transcribed and then partitioned into singular statements. Affinity diagramming is the task of putting up sticky notes with data artifacts on a wall and try to structure them. Those notes can be moved on the wall and should be placed in groups that share some relevance to the data. Figure 1 shows an example for a note.



37. (P1) At a certain level, the price difference is not that big anymore [...] and then I go for trustworthiness of the store as well. I cannot give you a real formula, it's mostly gut feeling. #00:30:00-6#

Figure 1: A sticky note for an Affinity

Participant profiles are created alongside the notes and serve to give an additional perspective to the statements of a participant. To identify the participants and their profile, a code is used for both the notes and the profiles. In figure 1 the participants ID would be "P1" while the number 37 is the ID of the note and consequently the statement on that note.
The Affinity method follows an bottoms up approach where unstructured data is first clustered into groups. Those groups then are clustered into higher level groups. A procedural list for an Affinity analysis would therefore look as such:

1. Stick notes on the wall

2. Cluster

3. Group

4. Second Level

5. Additional Level

6. Walk the walls

7. Design from data

As we will see in this thesis, our system in its current state will assist during step 2-6, the phase where actual interaction with the notes on the wall happens.

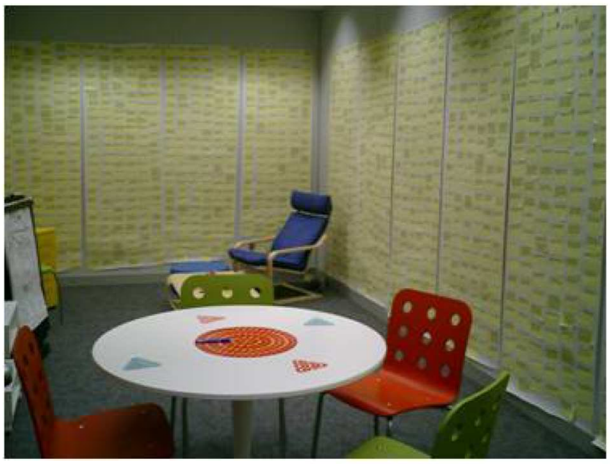


Figure 2: An example of a filled Affinity wall

2.2. Augmented Reality

Azuma (1997) has defined that augmented reality technologie has to have the following characteristics:

- Combines real and virtual
- Interactive in real time
- Registered in 3-D

During the last couple of years many applications, that support the second point have been introduced. While as Azuma (1997) contends a movie with virtual objects overlayed over film such as Jurrasic Park didn't count as augmented reality technology, video games such as Sony's `Invizimals`[1] are truly interactive. Augmented Reality also allows for different layers of feature sets that can be projected upon a core landscape freely without a need for time consuming reconstruction. This enables us to introduce additional information in a process without taking up permanent real estate when that resource is scarce. The system introduced in this thesis makes use of that fact.

2.3. Barcode recognition

Barcodes enable the storing of information a format that is easily understandable for computers. As such they are used when large quantities of items are used and have to be recognized such as in a retail store, ticketing etc. QR-Codes and other advanced barcodes allow for the storage of higher-level information and can be used to program a cellphone to use a certain wireless network amongst other applications. For our system they can store an identification key on each note. Libraries like ZXing [2] allow easy access to QR-Code recognition. Computer power has reached such a degree of sophistication that even cell phones can decode a QR-Code in almost real time.

2.3.1. QR-Code

A QR-Code is a 2-dimensional barcode meaning the information is coded in a NxN matrix rather than just a list of bits. It was developed by japanese Corporation Denso in 1994 and was standardized in 1997 (Denso Corporation 1997). Figure 3 shows an example of a QR-Code that contains the text: "ZPAC". The QR-Code contains a matrix of bits that usually are represented with black and white blocks of pixels. There are multiple "patterns" that serve to define the orientation of the barcode. Within the coded content besides the pure content are also versioning information, the data format (there are variants for japanese typesets amongst others), error correction and built in levels of redundancy depending on the version of QR-Code chosen.

[1]http://www.invizimals.com
[2]http://code.google.com/p/zxing/

Figure 3: A QR-Code that contains the text: ZPAC

3. Experimental Setup

There was one main requirement for the system at hand: It should be augmented unto the preexisting Affinity analysis process and enrich and facilitate the experience. At the same time it should be designed in a way where it could possibly be omitted with a still functioning analysis process in place. In essence we were looking to add an additional layer of tools. The use of those tools would not be required. For that approach to work we took a look at the existing Affinity process and looked for problems and issues that could be solved with it. One issue was locating a note on a wall, especially once there are a large quantitiy of notes on the Affinity wall. If those notes could be marked it would be easier to pinpoint others towards them. Ideas such as using a smart board with digital avatars as notes could not satisfy our requirement of leaving the core process as is. Consequently, since our idea was to project layers of services to the Affinity analysis we decided to project the markings unto the wall. For that we used a beamer that could highlight areas on a wall. If we introduced a way to identify the notes and the position of the notes on the projected wall space we could highlight the notes. To identify notes we had to use some kind of ID, conveniently in a way that it could be recognized by a machine. We used QR-Codes for that. These barcodes should contain an ID and could also be used to store further information if possible and desired. Decoding the barcode on an image also gives us the positional data of the barcode on that image. So we decided to use a camera that takes a picture of the Affinity wall. Each note on the wall contains a QR-Code that has its unique ID stored in it. At the corners of the wall we also display larger QR-Codes that just contain the information UP_LEFT, UP_RIGHT, DOWN_LEFT, DOWN_RIGHT. With these we can identify the corners of the displayer area in a photo taken from the wall.
A second issue that arises is the movement restrictions on an Affinity. During the Affinity

analysis it is very inconvenient to just read through the notes. If we could make the information on those notes available it would be much easier to just browse through them. Setting up such a database would also open up the system-to-be to further opportunities of data transformation, enrichment etc. without touching the original wall.
With these thoughts in mind we created a system as described in Section 4

3.1. Setup

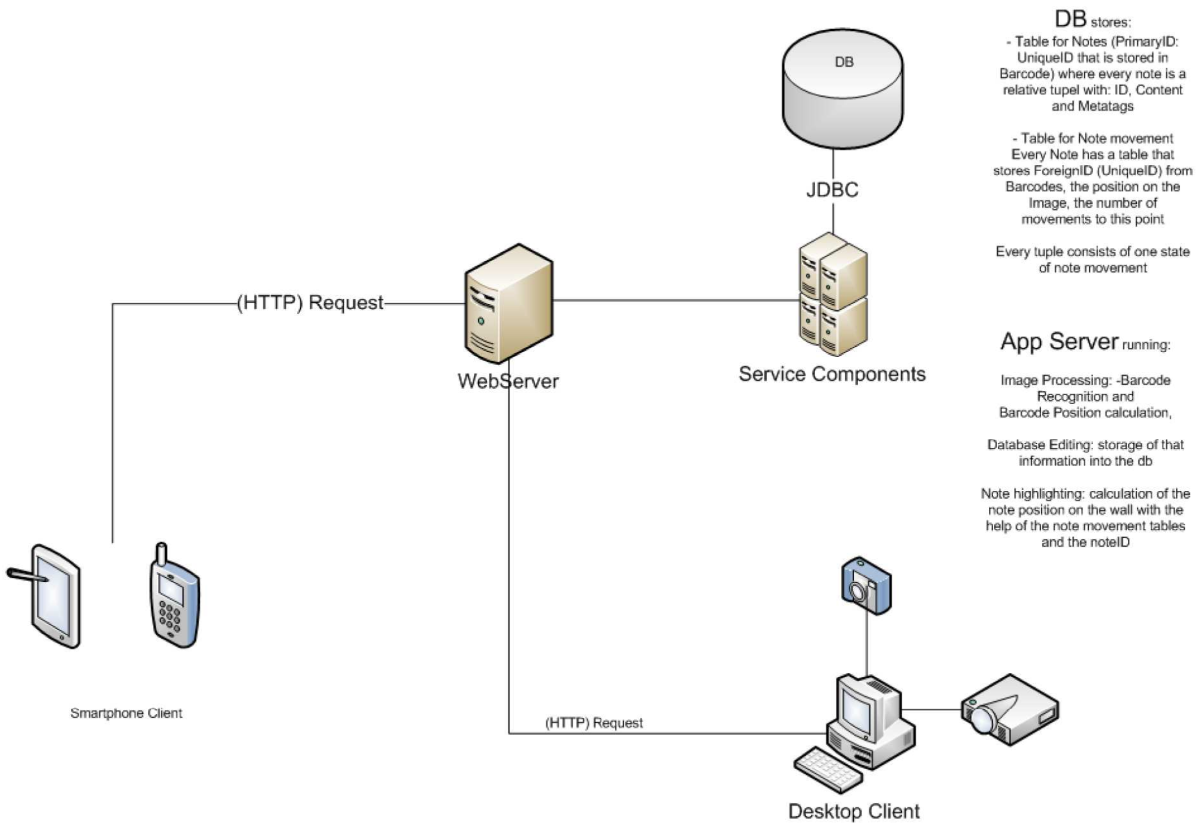

DB
DB stores:
- Table for Notes (PrimaryID: UniqueID that is stored in Barcode) where every note is a relative tupel with: ID, Content and Metatags
- Table for Note movement Every Note has a table that stores ForeignID (UniqueID) from Barcodes, the position on the Image, the number of movements to this point
Every tuple consists of one state of note movement
JDBC
(HTTP) Request
WebServer
Service Components
App Server running:
Image Processing: -Barcode Recognition and Barcode Position calculation,
Database Editing: storage of that information into the db
Note highlighting: calculation of the note position on the wall with the help of the note movement tables and the noteID
Smartphone Client
(HTTP) Request
Desktop Client

Figure 4: The system setup

One important facet of our system was that it should keep the cost of implementation and use as low as possible both in material value and in time consumed. Most of the required parts should therefore be already available in offices and labs. Figure 4 shows the system setup. We have an Android Smartphone, a Desktop Client and a Server running an application server and a database server. In the ZPAC-Lab, the Desktop Client is connected to a beamer which displays a computer screen on a wall. The wall is also used for an Affinity diagram.

3.1.1. Workflow

There are two different flows. The first flow is how the system will be used during an Affinity while the second flow is the setup of the system that has to be done before notes can be highlighted correctly. The setup-flow has to be performed before the system can be used and it should then be repeated in reasonable cycles i.e. the usability of the system will decrease over time, because notes will no longer be at their stored position, if this cycle is not repeated. Ideally this could be automated and be performed in short intervalls. The action-flow will be performed repeatedly during an Affinity session. A user can search the database for information, if he wants to he can highlight the note to show it to others and/or locate it on the wall. This cycle can be repeated Figure 5 shows the workflow(s) of our system.

3.1.2. Server

The application server is a JBOSS-Server [3] and the database is a Postgresql-Server [4]. Currently they both run on a laptop which is connected to the internal network of the University of Zurich. All communications to the clients are over HTTP-Requests.

3.1.3. Desktop Client

The Desktop Client is currently implemented as a Java application. It is designed as a thin-app, only able to upload images and to display images on a screen. That screen is a connected beamer. The camera that is connected to the Desktop Computer and takes pictures of the wall has to be calibrated to the beamer. We assume a flat wall for the beamer and that the camera is in an upright position to the wall. Figure 6 shows our highlighted notes on a wall with the use of our Desktop Client.

3.1.4. Smartphone Client

The Smartphone Client is an android app that can be run on both smartphones or tablets. It allows users to search the notes for keywords, IDs or participants. The displayed results can then be selected to be highlighted on the wall. The selected notes can also be read in their entirety as a digital representation and would then be highlighted with a different color on the wall. Figure 7 shows a screenshot of the android app.

[3]http://www.jboss.org/
[4]http://www.postgresql.org/

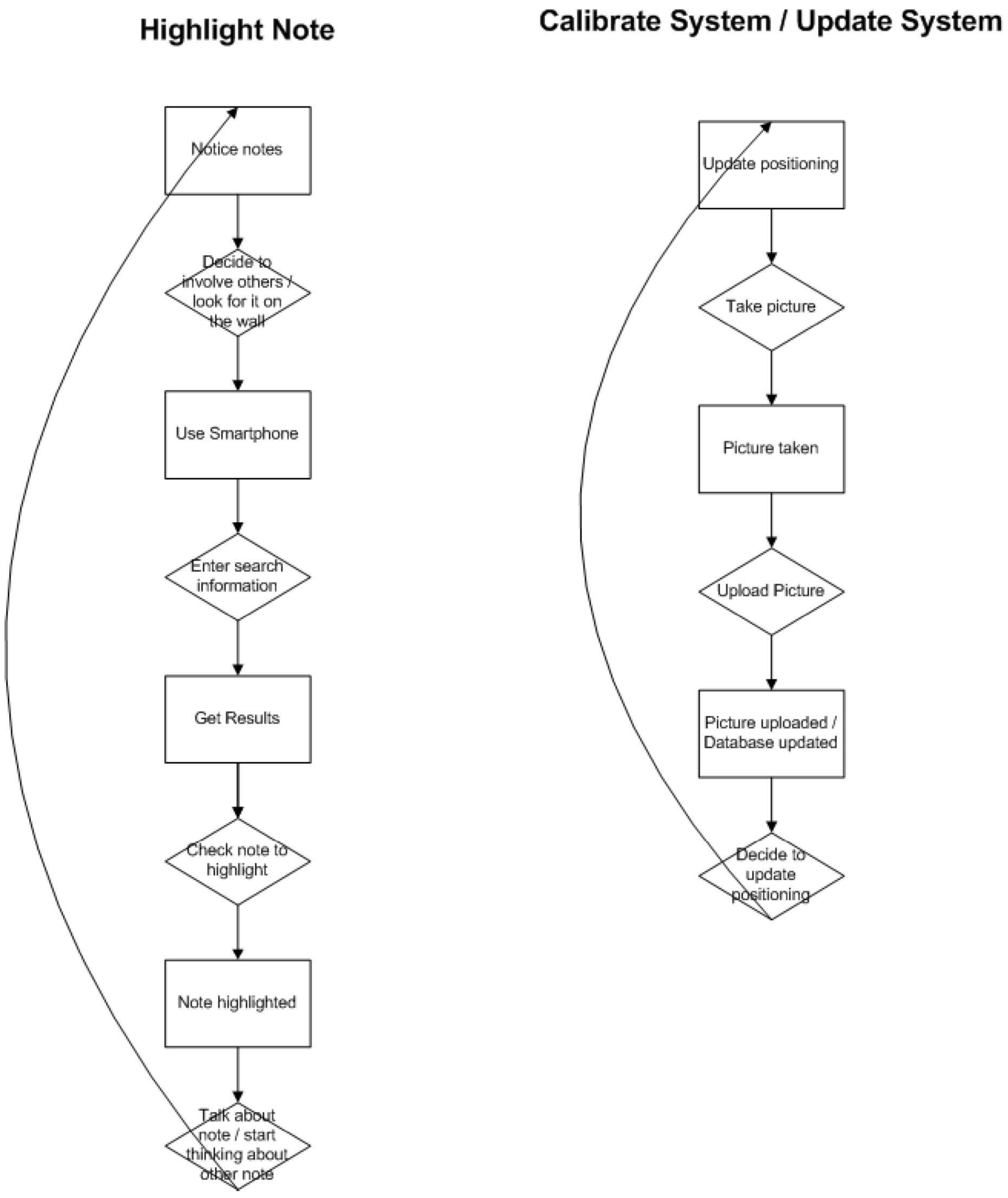

Highlight Note
Notice notes
Decide to involve others / look for it on the wall
Use Smartphone
Enter search information
Get Results
Check note to highlight
Note highlighted
Talk about note / start thinking about other note
Calibrate System / Update System
Update positioning
Take picture
Picture taken
Upload Picture
Picture uploaded / Database updated
Decide to update positioning

Figure 5: The workflows of our system

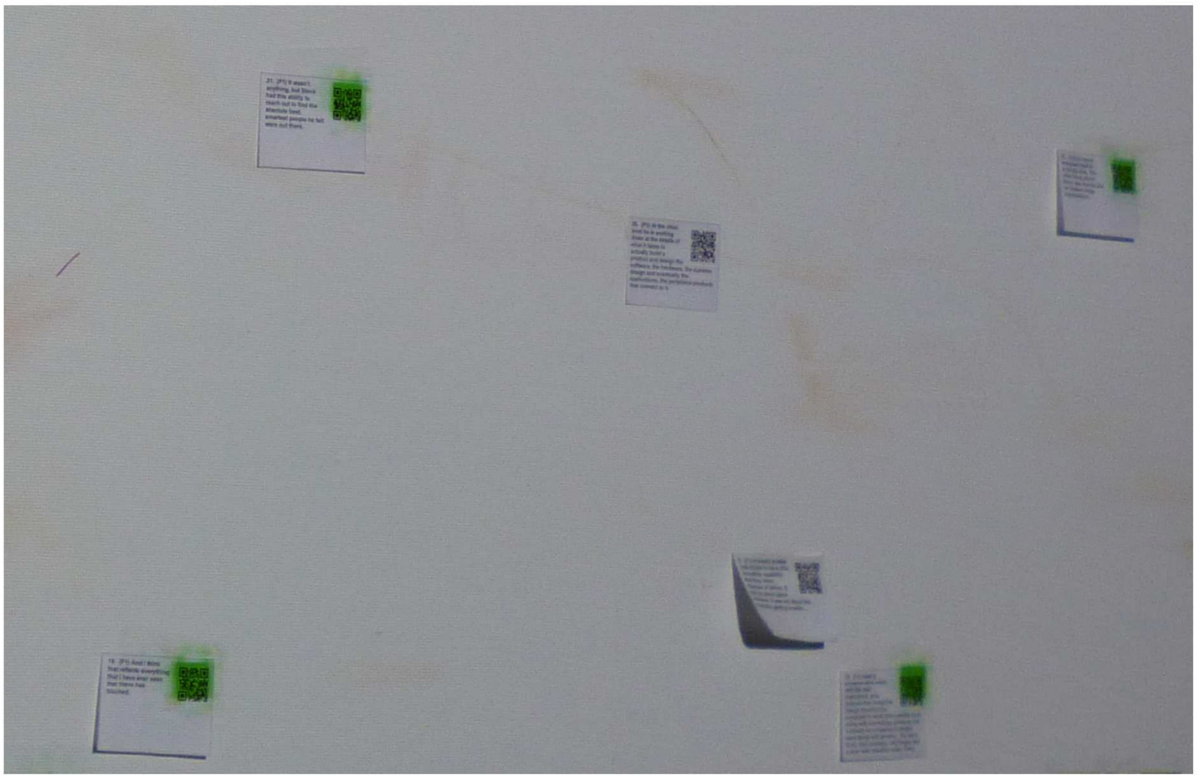


Figure 6: Some highlighted notes on a wall

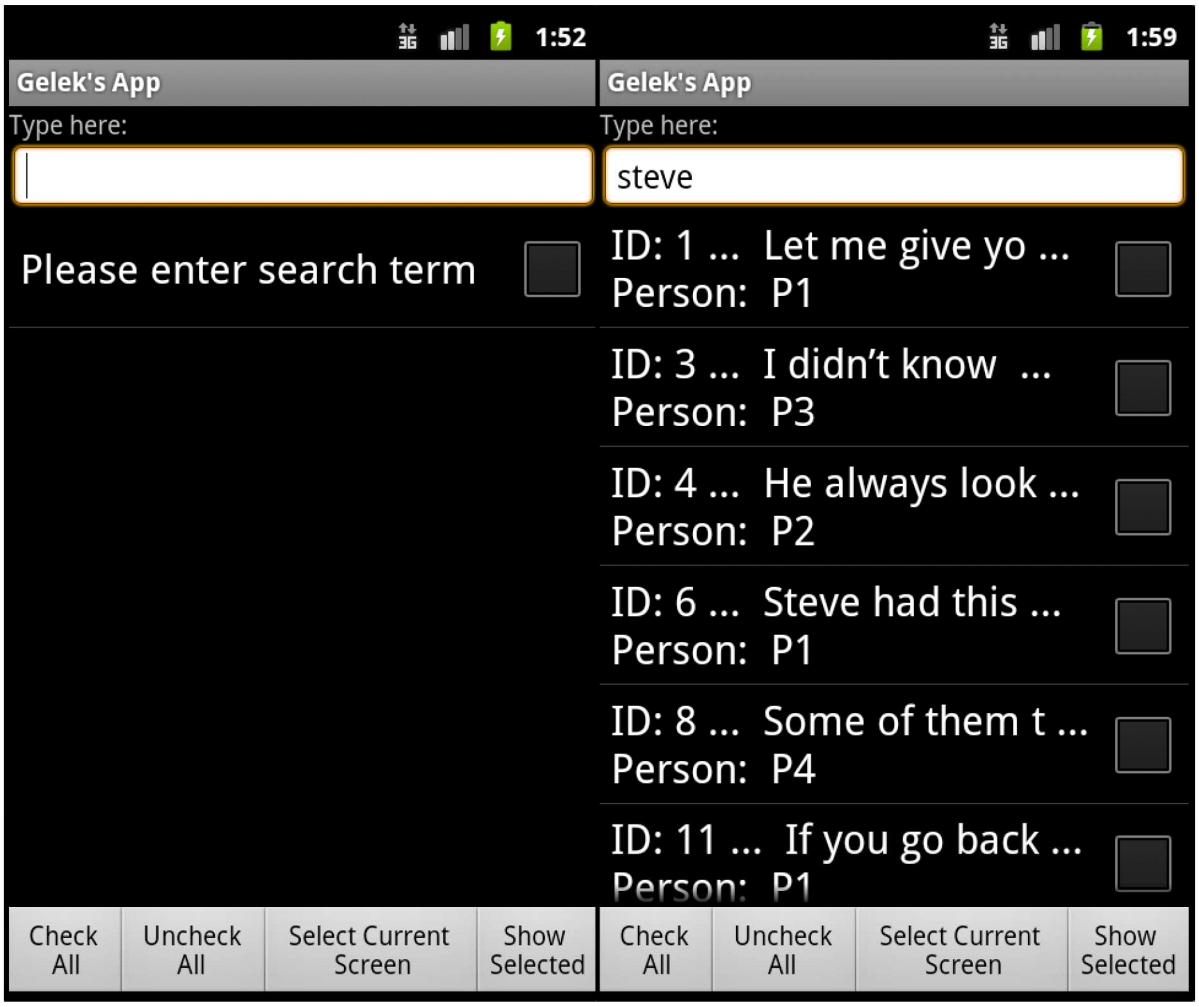

1:52
Gelek's App
Type here:
Please enter search term
Check All
Uncheck All
Select Current Screen
Show Selected
1:59
Gelek's App
Type here:
steve
ID: 1 ... Let me give yo ... Person: P1
ID: 3 ... I didn't know ... Person: P3
ID: 4 ... He always look ... Person: P2
ID: 6 ... Steve had this ... Person: P1
ID: 8 ... Some of them t ... Person: P4
ID: 11 ... If you go back ... Person: P1
Check All
Uncheck All
Select Current Screen
Show Selected

Figure 7: Two screencaps of our Smartphone Client: left side the search mask, right side with results for a search for the term "steve"

3.2. Issues

In regards to the barcode detection, the limiting factor for our system was the information density given to the decoder. While the camera resolution was sufficient in our setup (see section 3.1), noise like the brightness of the room (or lack thereof), the brightness of the ‘base’ display or the pixel value issues on the border regions of a stitched image could lead barcodes to not get recognized.
The Desktop Client in its current implementation is a thin client. To reach this goal we send bytestreams of images and leave all image handling operations on the server. This can have some disadvantages. First the amount of data sent can be considered high, especially since each highlighting on the wall leads to a new image being sent to the Desktop Client. At the same time once multiple instances are run on the server, the server capacity could be a factor in a potential increase of waiting time.

4. Implementation

This section will try to give an understanding of how the system was built. An overview of the system and its components is given and the components then discusssed seperately. It aims to facilitate an understanding of the different system layers and their cooperation.

4.1. Initial Requirements

After determing the desired use cases, our system requirements were defined. The specific choices mentioned in section 3.1 were a mixture of preferences and convenience with one facet being that the system could preferably become open-source. The system to be implemented should also try to lay the groundwork for future additions and enhancements so the system architecture design should account for that. Some actors of the system would be smartphones and tablets. To account for their comparatively weak computation power we decided to use a Client-Server architecture. So as to not restrict future development we agreed on an architecture that should be platform independent if possible. To achieve this, communications between the different actors in the system would be handled with HTTP-Requests with Java-Servlets running on the server. In our current iteration of the system, we decided to use an Android smartphone as hardware basis of the Smartphone Client. This allowed us to use one IDE, Eclipse, for the development of all our system components.

4.2. Design

Following the requirements analysis, we started developing first paper prototypes which we discussed with potential users in the ZPAC-Lab. Within the scope of this thesis, only the UI of the Smartphone Client was prototyped since the Desktop Client and the server-solution were designed to be used with minimal user input or had no UI to speak of. Figure 8 show one draft of a flow within the Smartphone Client. From these we started development on our Smartphone Client while at the same time developing first protoypes of the barcode recognition and highlighting display for our Desktop Client.

4.3. Proof of Concept Barcode Recognition

Barcode recognition is an essential part of our system so we tried early to define the requirements of our system to the barcode recognition component. These were:

- Recognize multiple barcodes on a picture

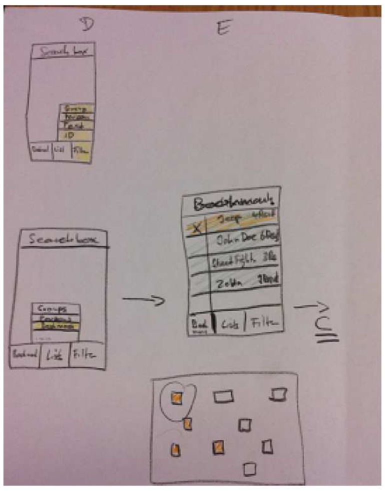
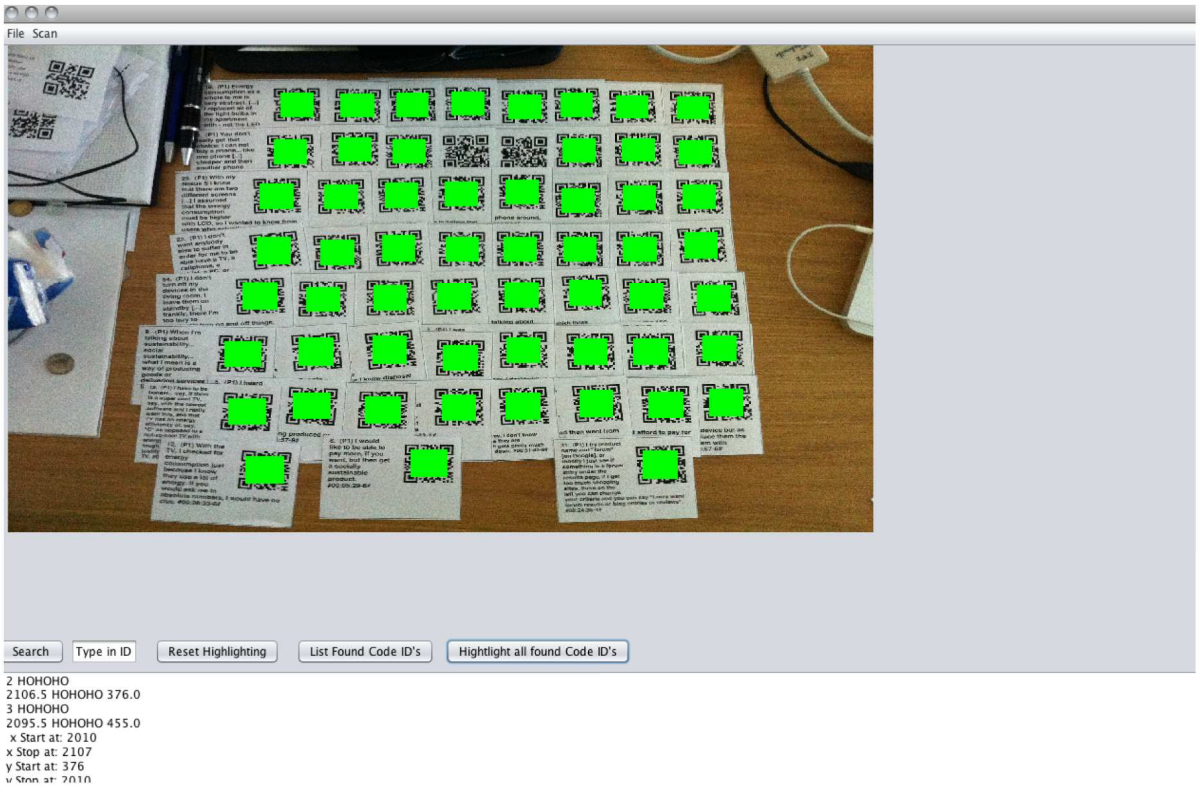


Figure 8: An early paper draft of the Smartphone Client

- Find positional data for the detected barcodes
- Be able to detect a barcode on a picture taken with a Hi-Res Camera so that the picture contains about the displayed area of a beamer
- Do it all in reasonable time
- Be able to store an ID
- Be able to store additional information
- Be able to determine rotation and orientation

We started out trying different barcodes and their methods to get a feel for the capabilities of current hardware and software solutions. We quickly established that 1-D barcodes such as EAN would not be sufficient for our case. We also wanted to use a broadly established type of barcode if possible to facilitate easier support of hardware and software. One barcode type that fit those criteria were QR-Code. With ZXing we also had an advanced library at our disposal which could be used in `JAVA` and had implementations for Android. After making the decision to pursue QR-Codes we wrote a first simple `JAVA`-Application which served as proof of concept that detecting multiple barcodes was possible. Figure 9 shows the prototype in action. We can look for an ID and have it highlighted or have all the detected ID's highlighted. Of course there were still some issues. Mainly, the way the ZXing-Algorithm works to detect color values of matrix points in a QR-Code is prone to noise, e.g. if you stitch an image together and



```
2 HOHOHO
2106.5 HOHOHO 376.0
3 HOHOHO
2095.5 HOHOHO 455.0
 x Start at: 2010
x Stop at: 2107
y Start at: 376
y Stop at: 2010
```

Figure 9: Our proof of concept prototype for Barcode Recognition and Highlighting

save the result in a format that will create noise along the border in the different bitchannels of the image it can be hard for ZXing to detect a QR-Code there. Overall tough the detection rate was high enough that we chose to use QR-Code, especially with the alternatives not yielding better results [5]. As it is, QR-Codes storage capabilities exceed our need but that space could be used in future iteration by adding information unto the note, e.g. the participant profile. We also looked at developing our own barcode solution, but quickly decided that this would be out of scope for this thesis.

4.4. Component Overview

Once all the requirements were defined and the proof of concept that the recognition of multiple barcodes is possible was demonstrated, the system was defined. The main points to consider were to (1) make the system as platform independent as possible, (2) easily enhancable and (3) able to run on off-the shelf hardware. In order to satisfy (1) the communication between the clients and the server is handled through HTTP-requests. That way there is well defined data being transfered without relying on one specific technology or programming language. (2) meant that we used an application server - client architecture where the different tasks where stored as services that could be easily accessed through a web `API`. (3) finally brought us to use JBOSS for our Application Server and a Postgresql-Server for our Database since both would run on conventional end customer hardware. Figure 10 shows the component model for our system and the first concurrently developed addition.

[5]We had about .900 detection rate in different trials when establishing clear light sources and illuminance settings

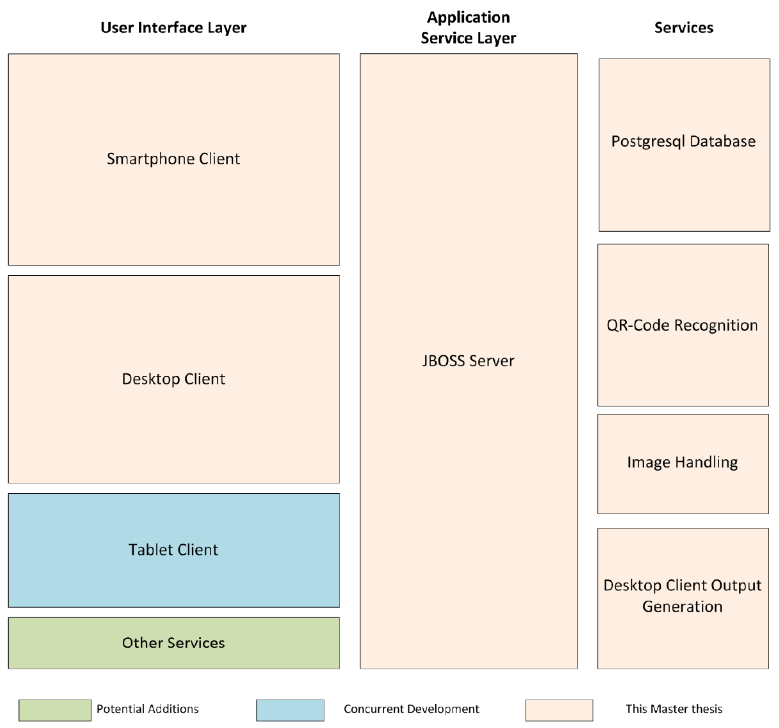


Figure 10: a component model of our system

4.5. Server

On our server hardware we had instances of Postgresql and JBOSS running. To configure those we used Eclipse and the JBOSS Application Server 7 [6]. All incoming communications on our server are in the form of HTTP-Requests which are handled in servlets. On the server there are different interfaces to the various service components. These interfaces can be injected at runtime into the servlets and be used to perform the business logic. Figure 11 shows the flow of a search-query from smartphone to server

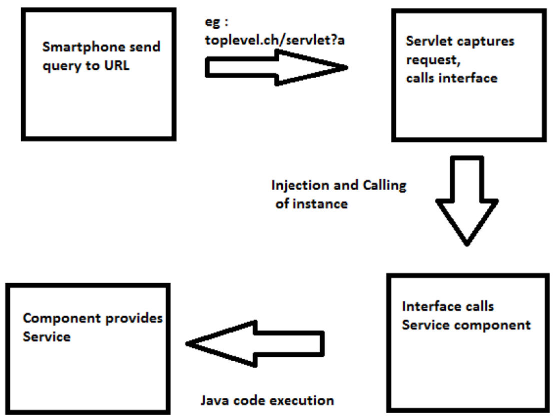

Smartphone send query to URL
eg : toplevel.ch/servlet?a
Servlet captures request, calls interface
Injection and Calling of instance
Interface calls Service component
Java code execution
Component provides Service

Figure 11: Flow of a Servlet Call

For simple cases, like sending a query, we used HTTP-syntax, in instances that were more complex or in returns of queries we used other means such as JSON (Douglas Crockford 2009). Since JSON allows coding a JSON-Object into a single String, we could leave our servlet infrastructure compact and handle the conversion after transmission. In the case of the Desktop Client, we used bytestreams to send the image data to and from the server. The interface layer was realized as conventional Java classes. It served the purpose of structuring the interaction flows within the server so that the different service components could be separated. By doing so, we could decide to repackage the service components into Enterprise Java Beans [7] in further iterations which would be a preferred design pattern (Crawford and Kaplan 2003). The interface layer will run objects that handle the application logic and call the service components for their services when necessary. As an example if a user wants to start the Desktop Client and get a fresh wall image he uses the login function on the Desktop Client. Once the user has

[6]http://www.jboss.org/jbossas
[7]http://www.jboss.org/ejb3/

done that, a servlet is called for a login, which in turn injects the respective interface. This interface enters data into the database, such as the username, the deviceID and the screen resolution. It also sets a boolean update value to `true`. Once the runtime thread on the Client rechecks the database for update (of course with the call of another servlet) it will see that it needs to update the wall which is yet another servlet that gets called. This servlet than injects an interface that will use the Database Component to look up the screen resolution, the Image Handling Component to create a new Image, the QR-Code Encoder to enclose the image with the boundary QR-Codes and finally the Desktop Output Generator that uses the saved image and transmits it to the Desktop Client. On the following pages we take a look at the 4 service components.

4.5.1. QR-Code Service Component

The QR-Code Service has three main responsibilities:

- Decode QR-Codes
- Encode QR-Codes
- Determine Coordinates of QR-Codes

it uses the ZXing-Library to achieve those tasks. Decoding a QR-Code is used to identify the notes on the wall-image and to define the wall boundaries with 4 Corner QR-Codes with content UP_LEFT, UP_RIGHT, DOWN_LEFT, DOWN_RIGHT. The settings for QR-Code detection are configured to reach the most extensive possible result set using triggers in the ZXing-Library called `TRY _ HARDER`.
The encoding function is right now only used to generate the 4 corner QR-Codes on the wall display. It can however provide the functionality for future iterations where we might want to be able to digitally create "note-representations".
Determining the coordinates of a QR-Note is obviously a very important feature for our system. The ZXing library creates a `Result` object that stores multiple `ResultPoints`. Each of those points reference the position and alignment patterns of a QR-Code. Our system uses this points and an algorithm to calculate the midpoint of a QR-Code pattern. The coordinates of that QR-Code pattern normalized to 1000 are then returned so that they can be saved in the database. A special case are the coordinates of the four corner QR-Codes. Their coordinates will be returned by a different method that will be invoked whenever an photo is uploaded from the Desktop Client to get the coordinates of a subimage that represents the displayed wall.

4.5.2. Database Service Component

The database service component enables the server to access and modify our database. It does so over a set of JDBC [8] statements that are sent with the use of a postgresql-java driver. The database itself is run on an postgresql-server instance and can also be modified through the use of tools such as `psql`. We initially created a draft of the database in play and consulted with experts in the field of database design. Their verdict was, that for the scope of our current project and most of its iterations, the current database design should be sufficient. Currently there are 6 tables in the database:

1. affinityrecord
2. analysts
3. beamerclients
4. clientdevices
5. informants
6. notes

(1) serves as a stamp of the actions on the Affinity wall, e.g. there is an entry for every note on every frame (each time a picture of the wall is taken and uploaded to the server) with its coordinates, rotation, the timecode and an event number that defines wheter this note is new, has been moved etc. (2) the analyst table stores the login information for an analyst which is not used right now but will be needed once user accounts are created to store features like highlight color etc. (3) and (4) serve as a way to keep track of all client devices that are currently interacting with the Affinity process. (3) has the columns of (4) as foreign keys with the addition of attributes for the screen resolution. There is also an update trigger in this table which is used to tell the Desktop Client that there has been an update on the Affinity wall (caused by the Smartphone Client users highlighting notes or uploads of new frames) (5) is needed to create a participant profile but has no further use for this thesis apart from the identification code which is referenced in (6). (6) finally is a digital archive for the notes where attributes such as text content, creation time, creator, owners, references to files and also wheter or not the note is currently highlighted or not is saved. All in all with this database we have the option of accessing additional information and notes of Affinity processes with a large quantitiy of notes quicker than if we just used paper.

[8] http://www.oracle.com/technetwork/java/javase/jdbc/index.html

4.5.3. Image Handling Service Component

The Image Handling Service Component is tasked with image modifications. Mainly this will mean changing the pixel color value for an array corresponding with highlighted notes. There are however other methods that can be used for other tasks such as creating a new image, changing the RGB-Type of an image, handling and saving an uploaded Image and others. One of the current deficiencies of the system is the usage of JPEG [9] as a compression format for the history of images saved on the Server. These pictures are as of this thesis not further used, but could potentially be used for history functions and traversion highlighting of notes. To circumvent a potential Bug in the `ImageIO` library of the Java SDK. This component is also responsible to cut an uploaded image once the caller delivers the x- and y- values of the resulting subimage. To do that, the caller will need to use the QR-Code Service Component.

4.5.4. Desktop Client Output Generator

In its current implementation, the system uses singular "screen shots" for the display of the Affinity wall. This means that there is a relatively high load between Server and Desktop Client. The repackaging of images into conventional outputstreams is handled by this component. It will take the current frame (the ID of that frame has to be handed over by the caller since there is a need to look it up in the database) and use a bytearray to create the outputstream. This component is the most likely to undergo large changes in further iterations if the Desktop Client is redone.

4.6. Desktop Client

Figure 12 shows us the two main screens of the Desktop Client UI. It is implemented for now as a Java application running `SWING` [10]. There is a small grid of buttons that can change its visibility state by simple mouseclick anywhere on the screen and that has the basic functions: login, load frame and exit display. Once the user has logged in with a preapproved screen name, the screen resolution and the login information are sent to the server. The server stores these, creates a new Affinity wall image and sets the update trigger. Once the Desktop Client App reaches its next cycle, the new wall image is requested and displayed.The right screen in figure 12 shows an empty screen where no notes have been highlighted thus far. Since HTTP-Requests do not provide for

[9]http://www.jpeg.org/jpeg/index.html
[10]Part of the standard JAVA SDK

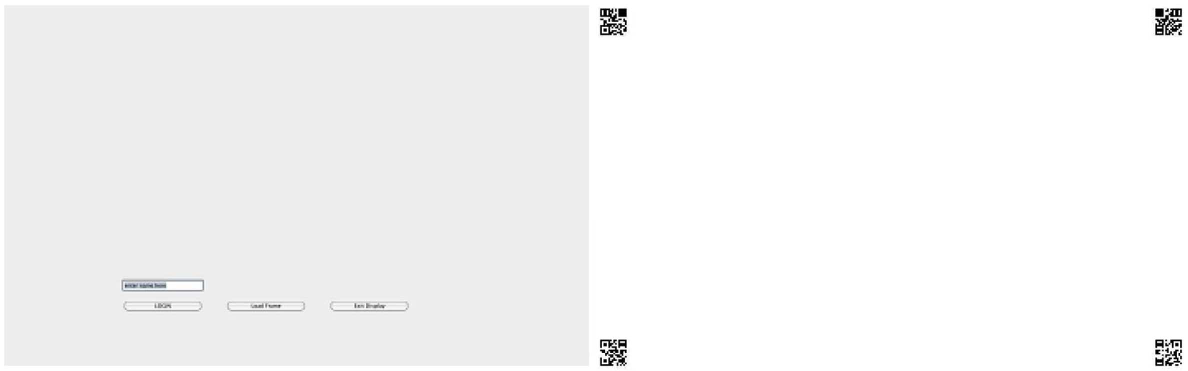


Figure 12: The Desktop Client after starting (left) and after login (right)

an easy and standard way to keep communication channels open indefinately we used the aforementioned update trigger in the database. Our Desktop Client application runs in a while loop that is true as long as the application has not been exited. The while loop condition is a boolean variable that gets its value from the application method `isRunning()`. Every time the loop is run, the application checks for updates, verifies its running state and then sends the loop-thread to sleep for a second. If the application is exited, the `isRunning()` method will change the running variable to `false`. The Desktop Client currently consists of 3 classes: the executing class with a main method and the while loop, the `BeamerClient` class consisting of a `JFrame` extension and `HTTPHandler`, a class that is used for all communications with the server. `HTTPHandler` has methods to check for updates, send login data and to receive and send picture data.

4.7. Smartphone Client

The Smartphone Client was built with the Android Development Kit [11] and Eclipse. Android applications are a set of activities that can be executed (Burnette 2008). As such the main screen of our application (see figure 7) was `StartActivity`, the class that is called first. Within this activity there are multiple views that can contain elements such as the searchbox or a scrollable list. We also used classes to handle `JSON` and string actions and had another activity to start an image representation of a singular note called `NoteActivity`. Every time you call a new activity from one activity and you want to keep accessing data you have to bundle it and hand it over to a class called `intent`. In our system we also had to handle outside events such as the state of the highlights once you enter the `NoteActivity` and once you leave it, since this would mean

[11]http://developer.android.com/sdk/index.html

different highlighting of notes.

5. Usability Tests

After implementing the system as described in the previous chapter, this chapter will focus on the usability tests that we undertook once a early prototpye was done. We wanted to follow the iterative development scheme (Rosson and Carroll 2002) so testing was an important part of this development step. The test form that we chose were the "think-aloud" tests (Lewis and Riemand 1993) where user are voicing their thought process so that we can get insightful input as to what led a user to perform a task a certain way. Because of the scope of our thesis, the tests where of the "Friends and Family"- scenarios described by Kuniavsky (2003).

5.1. Preparation

Before beginning with the tests, we started preparations. That meant that we had to recruit users, create scenarios, create testcontent and write a test protocol. For recruiting, because of the limited availability of schooled Affinity users, we decided to widen the search parameter to extensive computer users. These users are using computers and smartphones every day and are handling tasks on an advanced basis where they are familiar with basic UI-Ascpects such as scarcity of screen real estate. To somewhat counteract the lack of Affinity experience, the users were given short introductions into the Affinity Diagramming process. The scenarios that we designed tried to emulate the need for some of the use cases that we wanted, mainly trying to show a note on the wall to colleagues, searching for notes etc. As content we decided against creating own customized content and instead used interviews on the world wide web about a subject (Steve Jobs in this case) where the interview was fractioned into singular statements.

5.2. Test Design

Our initial test design consisted of a set of 5 questions which were directed towards a user of an Affinity. An example question was

$\boxed{\text{You'd like to know which notes mentioned the name Steve try to find them all on the wall}}$

To plan for the flow of the test session, we drafted a framework and a step-by-step protocol of the test where we outlined the goals for these sessions and the settings in which the tests took place.

5.3. Settings

The tests where all held in the ZPAC-Lab at the Institute for Informatics of the University of Zurich. Users were lead to the ZPAC-Lab where they got a short introduction into Affinity Diagramming and think-aloud testing, if they so desired this part was done in Swiss-German so as to make certain that there would not be any communication problems in understanding the setting. The sessions were all audiorecorded (for some of them, photos were shot) and the users were required to sign a consent form declaring their acceptance of such measures. The tests we're done one user at a time.

5.4. First run

The questions for this first run of tests were as following:

1. Try to start the Android App and browse through the database, search for a term like "framework". Show which notes you have found on the wall to a third person.

2. You want to know which notes mentioned the name Steve, try to find them all on the wall

3. Show all notes about Steve at the same time to a third person. One of the notes about Steve mentions game machines, try to set that one apart from the others.

4. You'd like to know which notes were from P1, show them.

5. You think there's something odd with the statement on note 11, so you show it to your colleague and have him read it to you from the wall so that you can check the statement

This list of questions were read out by the person conducting the tests. All users had to solve all questions.

5.4.1. Results

As per the test format, a think-aloud test would not deliver quantifiable results, but more facets and aspects that gave insights into user behaviour. As such one interesting aspect was that users seemed to stick to the Smartphone. They were usually concentrating on interacting with the cell phone and interpreted tasks such as showing notes in a way where they would show the cell phone and its screen display. This might actually have to do with the lack of Affinity Diagramming experience but was seen even if further emphasis was put on the wall during the introductions. Another aspect was the fact, that our search query function was case-sensitive, something that caused confusion when users were asked to search for “Steve” in the database. As one user put it, he also didn’t like the fact, that the start screen of the Smartphone application had the text “Please enter search term” under the text box since he assumed that he could enter something in this field or at least press the checkbox next to it (check figure 7 left). Aside from one user that made it a point to try out all possible buttons and options there was also the point that the fact that there were different highlighting colors was something that no one noticed. Rather the standard behaviour was to check one note after the other and uncheck them. Also, currently our system is implemented in a way that if you don’t explicitly uncheck a note, it stays highlighted even if you start a new search. Only when the Desktop Client is exited will all note states be reset in the database. This lead to some confusion as to why certain notes were highlighted even tough they contained no relevance to the search. To summarize, the aspects that we would spent further work on would be

- Remove unneeded buttons / menu options
- Make search case-insensitive (this was already done for the second run)
- Find a better workflow-solution to highlighting notes when a new search is started
- Try to emphasize that checking a search result will result in highlighting a note on the wall
- clarify the test procedure so that inconsistencies such as reading something to a user who interprets it as something else can’t happen

5.4.2. Modifications

After taking the input from the first run into consideration we decided to change some things in our test design. Mainly, the question set was altered and was made nonlinear in that it was no longer necessary to answer every task in order to finish. Also we made it a point to demonstrate the collaboration between Smartphone and Desktop Client-Affinity wall before starting the test. We also introduced more data, i.e. created more notes so that the Affinity wall would look more impressive. And finally we shortened the Affinity Diagramming introduction into the parts that were necessary for our test design.

5.5. Second Run

For our second run, we had the following list of tasks:

1. Look at the group of notes in the lower left corner. Pick a label you think fits [from a selection of 2-3 pre-written labels], and add the label to the group.
2. Start the Android app and find the first note from the group you labeled. Use the app to highlight it on the wall.
3. You think you remember a note that mentions "Steve" that could fit in this group. Show all the notes that talk about Steve.
4. (If they haven't shown it on the wall) You want others to help you look through the notes that talk about Steve. Highlight all of the notes that use the word "Steve" on the wall.
5. Of the notes that mention Steve, pick out the one note that talks about "game machines". Indicate it on the wall.
6. (If they don't use the prototype) Can you think of a way to use the prototype to set it apart from the other highlighted notes?
7. You want to see where all the notes from the interview with P1 are placed. Show where they all are.
8. Get Gelek to read you note number 11. (You may have to help him find it!)
9. Clear all the highlighted notes and close the Android app.

Different from the first run, the users were now given paper notes with the task instructions. Each note contained one task and was put onto a wall so that the user could reread the assignment if (s)he so desired. We also introduced some small interactions with the Affinity wall that weren't intended to test our system but rather give the user a way to familiarize with the process (there were still users that made use of our system for it).

5.5.1. Results

As with the first run, one point of contention was the search function. Users wanted to search for non-consecutive terms, something that would not be possible with the current implementation since it checks the database for a statement that contains the query term as a sequence. Users were also a bit confused about the coding scheme of an Affinity note where they had trouble to identify a note because they used the participant code, something that should be made more clear during the instructions. One user formulated multiple times that he was checking for ways to filter the search content, so that he would be able to only search for names or ID or text. Another thing that came up was the fact that users expected realtime updates, e.g. one user for task (2) wanted to search for the group label that he chose in task (1). Sometimes users tried cross-referencing methods to get to information instead of using intended features such as the singular highlighting of a selected note. Also apparent was, that users were still having trouble using both the wall and the Smartphone in combination to look for information for example showing us the Smartphone with the selected notes to solve task (3) instead of highlighting. In any case the interface wasn't intuitive enough that it could be used without a manual or instructions. One problem that will prove the limiting factor in having too many additions to the UI is the fact that Smartphone Screens are not that large. It could be interesting to see if the UI on a tablet could be more sophisticated while still being clear enough to handle. So amongst the points taken from this second run were:

- Enhance search capabilities to allow filtering and searching for non-consecutive terms
- Have some way to emphasize that checking a box results in highlighting a note, maybe introduce an audio representation of the fact or have a more elaborate animation.
- Maybe introduce a way to bookmark searches
- Create a manual with example flows and concise instructions.
- Create a larger data set so that the advantages of our system can be more accentuated (some users tried to find a note by reading through all of them)

6. Discussion

The current state of the system is that of a very early prototype. Still the ability to highlight a note on the wall is for the lack of a scientifically appropriate word, “cool”. We proved that with current technologies it is possible to create a layer that is able to help locate notes on a wall and to browse through note content without changing the Affinity Diagramming process. Of course, there are some limitations, mainly that the Affinity wall is restricted to the displayed areas that a beamer can reach and the fact that you need to take new photos to update the note movement. These are things that can benefit from technological progress. What is apparent is that there is still a need for manual instructions. Something that could be made less important if the UI went through further design iterations but will still be necessary since the introduced interaction is not grown “organically” from the existing Affinity Diagramming process. But our system has also another evaluation aspect in that there are multiple potential additions to it that will rely on some of those basic service components. There is currently already a second master thesis in progress that tackles another use case but roots on the same technological basis.

6.1. Further Work

When talking about further work there are two categories: Addition to the underlying system that fullfil other use cases and enhancements and improvments to the current system to improve productivity and usability of the current end user implementations. The possible developments of the first category are somewhat out of scope for this master thesis yet they might prove especially intriguing. As for the current system, there are naturally multiple ways to improve the system. From our perspective the most fruitful improvements are:

1. Adapting the UI of the Smartphone Client to address some of the feedback of the test run (mainly search features).
2. Building a UI for Tablets that make use of higher screen resolutions
3. Building a new Desktop Client that is “smarter” than the current thin client and significantly reduces bandwidth load
4. Introducing security measures to protect and encode content that is sent through HTTP-Requests and stored on the database

5. Capsulating the Service Components into Enterprise Beans so that they can be used project independent on the server
6. Introduce a way to bookmark searches so that a user can load the bookmarked notes configuration onto the wall to show it to others
7. Introduce a way to have multiple Affinity processes on the same server instance / or find a way to automatically create multiple instances and assign everyone to the correct one
8. Introduce a way to ammend notes or to create tags/comment to them

Of course there are still more points but these would address most of the needs that arose during the thesis timeframe.

7. Conclusion

In this thesis we looked at the Affinity Diagramming process and its weaknesses. We then addressed some of them by formulating use cases that we wanted to solve with an additional layer of tools that would provide optional assistance. After introducing this system and showing its components we looked at its implementation and how the usability was tested. Affinity Diagramming is a method that makes use of paper and its advantages. In this way it is something of an oxymoron in that one tool to get to design ideas for digitally driven design is to use paper. If you accept that fact and its elegance, the way that an Affinity can help structure data is impressive. The concept of not touching the underlying Affinity Diagramming process and adding additional layers is the consequently driven conclusion of that. And having an additional layer to look for a note is certainly a useful feature, having a convenient way to browse through the notes is too. It will however be most intriguing to see how future developments will interact with one another. There may even lie a potential trap in that too many layers will start to create a noise such that the Affinity Diagramming processes biggest gains of using paper will no longer be valid. In any case, the never-ending search for an ideal system to assist a method to find design improvements is surely a fitting analogy!

8. Acknowledgments

The author would like to thank everyone involved for the cooperation mainly Gunnar Harboe and Prof. Elaine Huang for the supervision and Lukas Keller for always being available to throw around implementation ideas and concepts. Special thanks also to Prof. Elaine Huang for building and leading a spirited group of people that were always willing to invest time and effort to be helpful. Furthermore everyone else at the ZPAC-Lab from Christian, Sarah, Silke and Simon for their great attitudes and willingness to always provide assistance and ideas (and of course the great food every wednesday). Also further thanks go to Dietrich Christopeit for taking the time to look at the Database design and for all test users who shall not be named to honor the consent form.

References

Azuma, Ronald T., 1997. “A Survey of Augmented Reality” *In Presence: Teleoperators and Virtual Environments*, 6,4 (August 1997): 355-385

Beyer, H, and Holtzblatt K., 1998. *Contextual design: defining customer-centered systems.* Morgan Kaufmann

Burnette, Ed, 2008. *Hello, Android; Introducing Google’s Mobile Development Platform* The Pragmatic Bookshelf.

Crawford, William and Kaplan Jonathan, 2003 *J2EE Design Patterns; Patterns in the Real World* O’Reilly

Denso Corporation, 1997. *QR-Code Standardization*, http://www.denso-wave.com/qrcode/qrstandard-e.html, retrieved on January 27th 2012.

Douglas Crockford, 2009. *Introducing JSON*, http://json.org, retrieved on January 27th 2012.

Kuniavsky, Mike, 2003 *Observing The User Experience; a practitioner’s guide to user research* Morgan Kaufmann.

Lewis, Clayton and Rieman, John, 1993. *Task-Centered User Interface Design: A Practical Introduction*, http://grouplab.cpsc.ucalgary.ca/saul/hcitopics/tcsd-book/, retrieved on January 27th 2012.

Rosson, Mary Beth, and Carroll, John M., 2002. *Usability Engineering; Scenario Based Development of Human Computer Interaction* Morgan Kaufmann

A. Code

The source code of the components and javadoclets for our project can can be found on the attached CD. The script for the database creation is also on the CD.

B. Test

The consent form and the Test protocols for the two test runs are on the following pages, the audio recordings can be found on the attached CD.

Consent form

Data collecting

During the test, data will be collected either by hand written notes or by voice recording or by both methods. The collected data are basically observations about the interaction with the app that is tested.

Confidentiality

The data collected during this test will be analysed by researchers from the ZPAC Research Group at the University of Zurich. If the data will be shown to outside persons, it will always be anonymised.

Participation

The participation in this test is voluntary and any participant can withdraw from the test at any time.

I have read and understood the consent form:

Name (Block letters)	________________________________
Signature	________________________________
Place and Date	________________________________

Think-aloud Usability Test:

Framework:

The system is trying to support and enhance the affinity analysis with visual feedback and simplified searching through content. As such there is a smartphone android app and a beamer-client which are both connected to a database in which the affinity data is stored.

Setting up the system (getting the affinity-data into the db and capturing qr-codes with a camera and uploading them is not part of this test run).

Users will be skilled in usage of common computer systems and have a background in CS.

Users be familiarized with an affinity during the test and a short demonstration of the main use cases but will not be given any further training / documentation or help in how to use the system.

We want to see if using the test system is something that the user prefer to other means (e.g. do they use the highlighting of a note with their smartphone client or do they point with their fingers towards the note)

Usability Goals: We want all test groups to be able to fulfill all tasks

Infrastructure:

ZPAC Lab, camera, beamer, laptop with Server (JBOSS Server / Postgres Server) + Desktop Client / recording device / Instructions for test (Tasks etc) /

Usability Test:

You have volunteered to take part in this Usability Test. Thank you. As Part of this test, you’re actions will be recorded on an audio recorder.
The test will contest of 2 phases, first you will get a quick introduction into affinity diagrams, then you will be asked to do a couple of tasks with the help of a system
---

You just have been given a quick introduction into affinity diagrams and how to use them.

For the following tasks you should try to use the Android App that you were just shown:

- Try to start the Android App and browse through the database, search for a term like “framework”. Show which notes you have found on the wall to a third person.

- You want to know which notes mentioned the name Steve, try to find them all on the wall.

- Show all of the notes about Steve at the same time to a third person. One of the notes about Steve mentions game machines, try to set that one apart from the others.

- You’d like to know which notes were from P1, show them.

- You think there’s something odd with the statement on Note 11, so you show it to your colleague and have him read it to you from the wall so that you can check the statement.

Step-by-step program:

Before test:

- Prepare System (connect Server, Desktop Client beamer, Smartphone Client)
- Prepare Affinity wall
- Prepare Room (check lightning etc.)
- Check recording device and materials

Test

- introduction
- overview of test / test method
- affinity diagrams
- (if necessary) prototype demonstration/ android demonstration
- 2nd round of tasks
- conclusion

After test:
- check if recordings ok
- check notes if made

Think-aloud Usability Test Run 2:

Framework:

The system is trying to support and enhance the affinity analysis with visual feedback and simplified searching through content. As such there is a smartphone android app and a beamer-client which are both connected to a database in which the affinity data is stored.

Setting up the system (getting the affinity-data into the db and capturing qr-codes with a camera and uploading them is not part of this test run).

Users will be skilled in usage of common computer systems and have a background in CS.

Users be familiarized with an affinity during the test and a short demonstration of the main use cases but will not be given any further training / documentation or help in how to use the system.

We want to see if using the test system is something that the user prefer to other means (e.g. do they use the highlighting of a note with their smartphone client or do they point with their fingers towards the note)

Usability Goals: We want all test groups to be able to fulfill all tasks by using our system.

Infrastructure:

ZPAC Lab, camera, beamer, laptop with Server (JBOSS Server / Postgres Server) + Desktop Client / recording device / Instructions for test (Tasks etc) /

Usability Test:

You have volunteered to take part in this Usability Test. Thank you. As Part of this test, you’re actions will be recorded on an audio recorder.
The test will contest of 2 phases, first you will get a quick introduction into affinity diagrams, then you will be asked to do a couple of tasks with the help of a system
---

You just have been given a quick introduction into affinity diagrams and how to use them.

For the following tasks you should try to use the Android App that you were just shown:

---

1. Look at the group of notes in the lower left corner. Pick a label you think fits [from a selection of 2-3 pre-written labels], and add the label to the group.
2. Start the Android app and find the first note from the group you labeled. Use the app to highlight it on the wall.
3. You think you remember a note that mentions "Steve" that could fit in this group. Show all the notes that talk about Steve.
3b. [If they haven't shown it on the wall] You want others to help you look through the notes that talk about Steve. Highlight all of the notes that use the word "Steve" on the wall.
4. Of the notes that mention Steve, pick out the one note that talks about "game machines". Indicate it on the wall.
4b. [If they don't use the prototype] Can you think of a way to use the prototype to set it apart from the other highlighted notes?
5. You want to see where all the notes from the interview with P1 are placed. Show where they all are.
6. Get Gelek to read you note number 11. (You may have to help him find it!)
7. Clear all the highlighted notes and close the Android app.

Step-by-step program:

Before test:

- Prepare System (connect Server, Desktop Client beamer, Smartphone Client)
- Prepare Affinity wall
- Prepare Room (check lightning etc.)
- Check recording device and materials

Test

- introduction
- overview of test / test method
- affinity diagrams
- (if necessary) prototype demonstration/ android demonstration
- 2nd round of tasks
- conclusion

After test:
- check if recordings ok
- check notes if made