Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science

Map-Based Interaction using Digital Pen, Stylus and Touch Screen: A Comparative Evaluation of Usability and Performance

Dissertation for the master degree of Computer Science

Szabolcs Becze

Promotor: Prof. Dr. B. Signer
Advisor: Dr. B. Dumas

June 9, 2012

Contents

1 Abstract

The digital pen has been around for more than one decade, it has been the topic for many researches. Usability and performance evaluations do exist, the digital pen was compared with touch inputs and it was integrated with mapping applications as well. However we found no researches of evaluating digital pen, stylus and touchscreen inputs. In this thesis we make a usability and performance evaluation of these three input methods.

The results of the evaluation shows that digital pen was significantly faster and more accurate than stylus and touchscreen inputs. We also found that those participants who had less experience with computers were significantly slower than intermediates and experts and the same applies to those participants who had no previous experience with smartphones. These were expected results, however not everything what we expected was reflected by the measured data. For example we wound no significant difference between the performance of stylus and touchscreen input.

We concluded that, in certain use-cases when printed maps can be applied, digital pens can greatly enhance the usability of the annotation process and even those who have less experience with computers can use it easily end efficiently.

2 Acknowledgment

I like to dedicate my thesis to my girlfriend Julia, and to my friends in Brussels, who always encouraged, and cheered me.

I would like to thank also to Bruno, who constantly pushed me toward the end of this project, and helped me with advices.

3 Introduction

The invention of the digital pen & paper technology goes back to 1996 when Christer Fåhræus envisioned a digital pen which makes it possible to capture written data on a special paper[4]. Small dots are printed on this special paper, which are encoding coordinates on a large virtual surface. On the pen a built-in infrared camera can sense these small dots and it is able to decode it with image processing techniques.

Using this digital pen & paper technology many companies developed their own digital pens adding a display, a speaker or Bluetooth/WiFi communication. Since then many researches have been done on digital pen and paper, in this thesis we see some of them in more detail.

The touch screen's history goes back to 1965[9], when E.A. Johnson described his work on capacitive touch screens in a short article. Many different technologies have been used since than: resistive, surface acoustic wave, capacitive etc. The touchscreen has two main advantage: it enables one to interact directly with what is displayed, rather than using a pointer controlled by a mouse or touchpad, and it does not require any intermediate device that would need to be held in the hand (other than a stylus). Touchscreens are popular in the medical field, in heavy industry, museum displays or room automation where rapid and accurate interaction is needed and the usage of mouse and keyboard is circumstantial. It became very popular with the spreading of handheld devices such as iPads and smartphones, also many researches has been done on the usability of touchscreens.

Stylus is a small writing utensil used to assist in navigation and provides more precision when using them on touchscreens. It also protects the screens from the natural oil from one's hand. It became very popular in the smartphone industry, for example the new Samsung Galaxy Note supports both touch and stylus inputs. There were many researches on the usability of styli[30][27], we discuss some of them in a later chapter.

3.1 Research objective

Until now these technologies were not mature enough to perform a comparative evaluation on each of them. The map interaction with digital pens is also not well explored, so we imagined a similar application to TripAdvisor[10], where users can annotate places related to their trip, attaching a comment, image, a rating and a timestamp.

Our research objective is to perform a usability and performance evaluation on digital pen, stylus and touchscreen inputs with the TipAdvisor-like application, where the users can annotate places on a digital printed map.

The target of this evaluation were two different user groups: users with different level of computer experience, age, gender etc.

3.2 Research questions

With this evaluation we tried to find answers to the following questions:

- Is digital pen significantly faster than touch or stylus input?
- Is digital pen significantly more accurate than touch or stylus inputs?
- Is stylus faster than touch input?
- Is stylus significantly more accurate than touch input?
- Are computer expert users faster than beginner intermediate users?
- Are computer expert users more accurate than beginner intermediate users?
- Are users with smartphone experience faster than other users?
- Are users with smartphone experience more accurate than other users?
- Are users younger than 30 faster than users older than 30?
- Are users younger than 30 more accurate than users older than 30?

3.3 Thesis structure

In chapter 4 we talk about the state of the art of digital pen, stylus and touch inputs and we will see some of them in more detail.

In chapter 5, we talk about the technical realization of the system, defining first the functional and non-functional requirements then we introduce the system architecture.

In chapter 6, we see how the evaluation has been prepared, executed and how the data was analyzed. Based on the results, we give answers to our research questions.

In chapter 7, we draw some conclusions from the research and we give some advices for the future work.

4 Background (10 pages)

Many articles have been written about the usability of stylus, digital pen and touch based inputs, but we found none which compared all three. We also noticed that very few articles have been written on the usability of digital pen and map interactions. In this chapter we take a closer look at the state of the art of digital pen & paper, stylus and touch inputs and we also see some researches on interactions with maps.

4.1 Digital pen & paper

In this section we discuss about different projects realized with digital pen & paper.

Note: we discuss about the technical aspects of the Anoto technology in a later chapter.

Everything started in 1996, when the Swedish Christer Fåhræus wrote his doctorate. He envisioned a digital pen which makes it possible to capture written data with a digital pen. The first prototype of the pen was completed in 1997, using real time image processing. In 2000 the Anoto started to work with more partners, and by 2003 companies like Nokia, Sony Ericsson or Logitech released their own digital pens base on Anoto technology. In 2007 Livescribe released it's own digital pen (we used at our evaluation a Livescribe pen)[6]. Since then Anoto was growing continuously, dominating the digital pen & paper market [3, 4].

Parallel with the growth of the Anoto corporation many researches has been done in various domains with digital pen & paper. Let's see the most important ones:

Paper Augmented Digital Documents:

The project Paper Augmented Digital Documents(PADD)[18] was among the first researches in 2003 with digital pens. They where investigating how digital documents could be manipulated easily either on a computer screen using mouse and keyboard or on paper using an digital pen. The user could make annotations on a paper, the digital pen saved these strokes and sent it to a "stroke collector", which saved it into PADD database. On the Fig. 1 you can see the two different representations of the same document. We can immediately notice the accuracy of the digital pen when we look at the stroke input. This accuracy made it well suited for many tasks such as proofreading, editing, and annotation of large format documents like blueprints.

PaperBased Mobile Access to Databases:

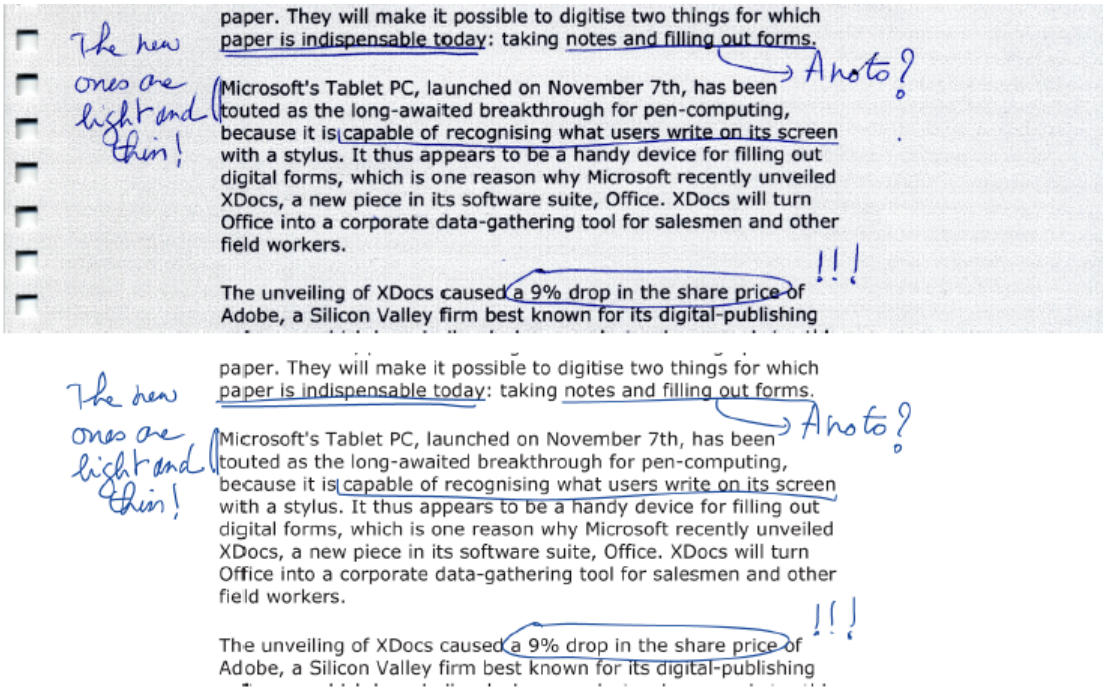

Figure 1: Two representations of the same PADD. Top, we can see the scanned version of the paper document annotated with digital pen. Bottom the digital equivalent after synchronization.

In 2005 Anoto pens were used at the world largest international art festival in Edinburgh[13], where they created an interactive brochure, a city map and a bookmark. They defined active areas within the documents, and with an Anoto digital pen they could interact with these, sending query requests to an application database. The response to this request was sent back to the visitor in form of a generated speech output. In fact they created much more, they defined the iPaper framework. On the image Fig. 2, you can see the different components of this framework. The iPaper client which defined active components sent requests to the iServer. The iServer linked these requests to not only static information but also to dynamic program code, so the selection of an active component resulted the execution of a program code on iServer, for example it could access to an external database retrieving some application information.

Print-n-Link: Weaving the Paper Web:

The same iPaper framework was used to enhance the article reviewing process, since the actual reading of the articles often takes place on paper and frequently on the move[22]. Let's look at the Fig. 3. The user points to a citation on an interactive paper document with a digital pen, and

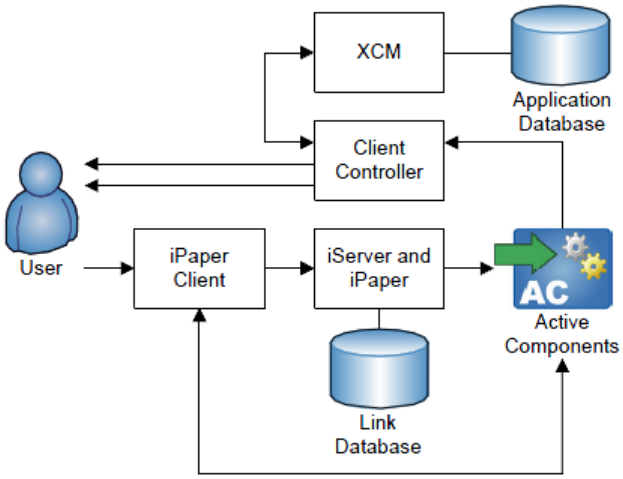

Figure 2: iPaper framework components

this position is sent to the iPaper Client, which issues and HTTP Request with the coordinates to the iServer. The iServer accesses the application database and retrieves metadata about the requested item (for example about a citation).

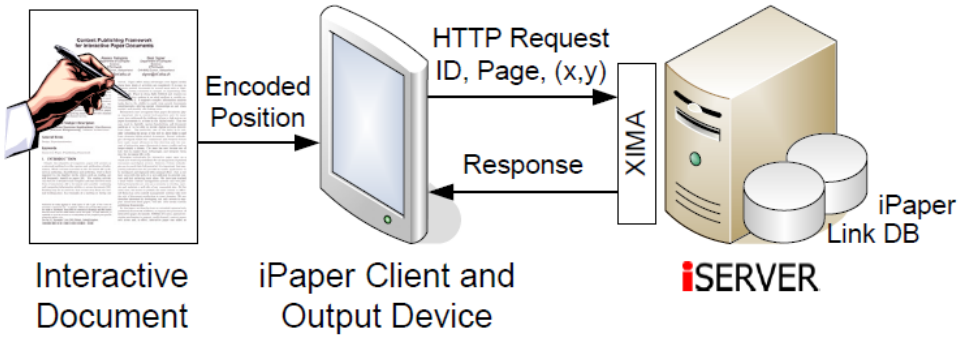

Figure 3: The user sends to the iPaper Client the encoded position, and the device sends further HTTP requests to an iServer.

The Digital Voting Pen at the Hamburg Elections:

In 2008, digital pen was used to accelerate the very time consuming election process for the state parliament[11]. The voting and data collection process is shown in the Fig. 5. The pen was directly connected to a local PC with a USB cable, then the written data was physically transfered to the Main Administration building where the data could be processed through

LAN. They needed these static connection due to security reasons. They claimed that by introducing the digital pen saved at least 2-3 million euros.

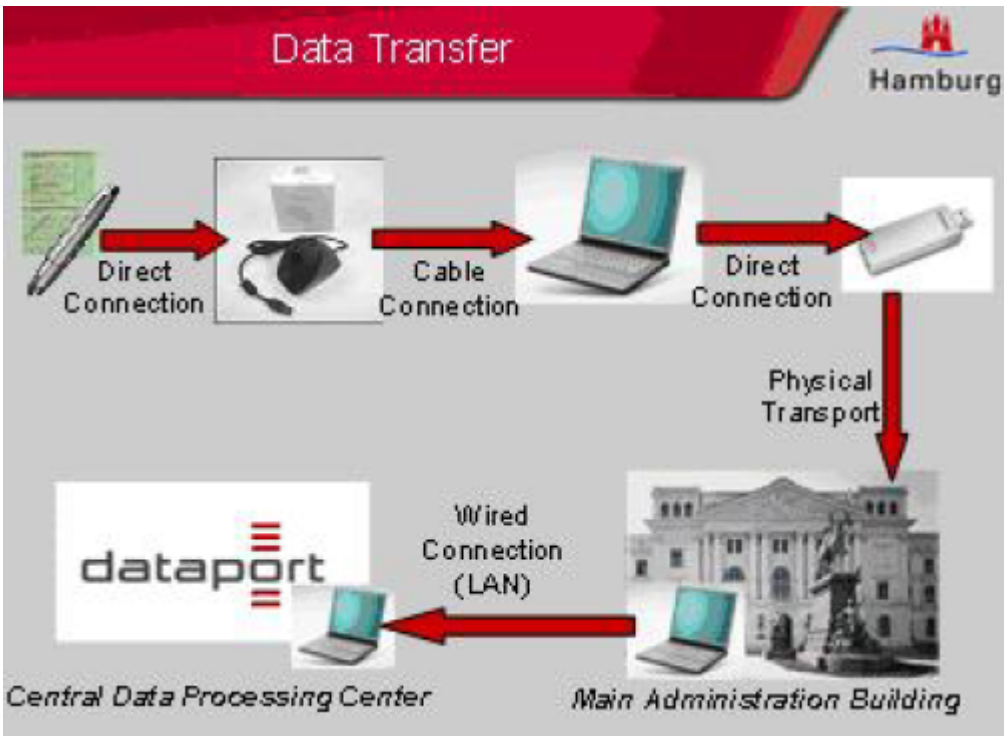

Figure 4: The process of how a vote is saved and delivered to the data center.

Practical Environment for Realizing Augmented Classroom with Wireless Digital Pens:

Wireless digital pens were also used to augment the work flow in classroom[21]. The students could interact with a screen using the pen, increasing the interactivity of the participants and the level of attention.

Enhancing UML Sketch Tools with Digital Pens and Paper:

They enhanced the seamless integration of paper-based digital UML-sketching[16] by combining special UML sketchbooks (printed with Anoto dot pattern) with tabletops.

Iterative Design and Evaluation of an Event Architecture for Pen-and-Paper Interfaces:

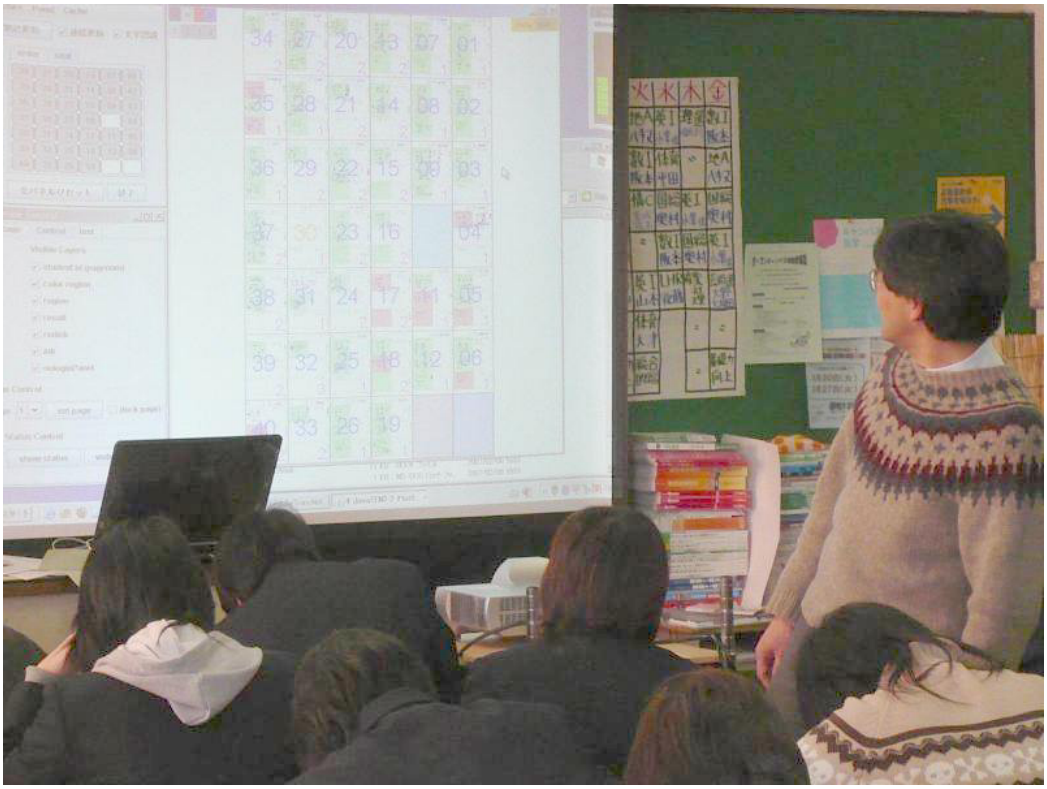

Figure 5: Interactive classroom, the student hared a projector, and interacted with it using a digital pen.

They developed the PaperToolkit, a Java framework to use digital pens together with PC[31]. Their event based approach let the developers to de-

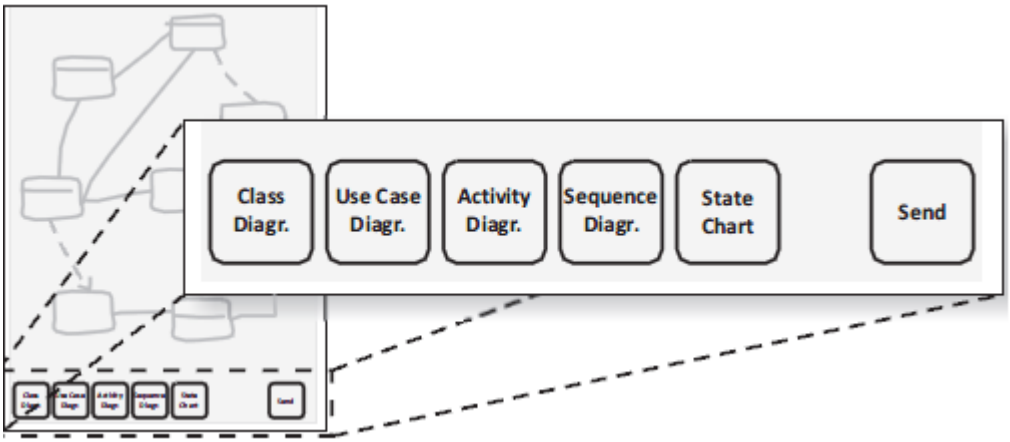

Figure 6: Control buttons were defined in the bottom of the printed paper.

fine active regions which dispatched pen events (like *click* or *stroke created*). An event handler caught these events, and run the attached code. The result then could be sent back to the developer/user. In the Fig. 7, you can see an example request/response.

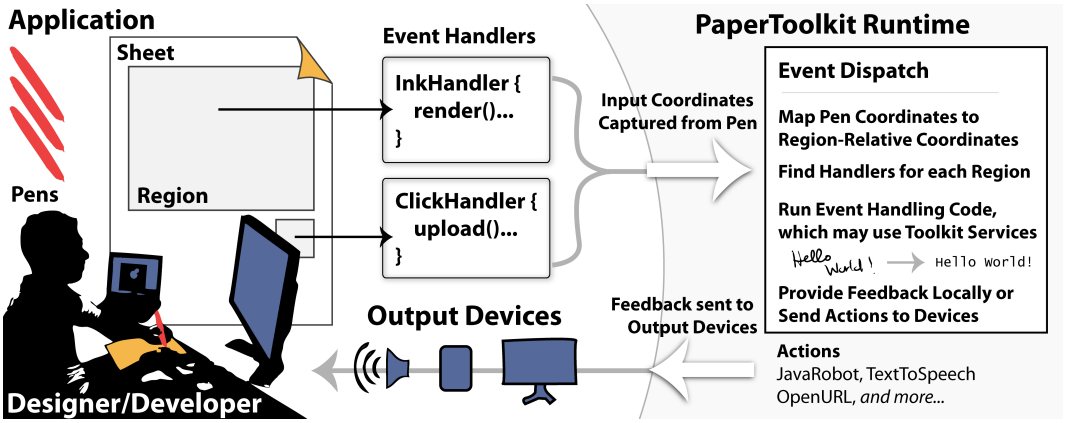

Figure 7: Example how the PaperToolkit handles events.

Designing Pen-and-Paper User Interfaces for Interaction with Documents:

In 2009 there were a study about the usability evaluation of a digital pen, about how can be gestures and symbols enhance it[26]. They provided an interaction framework, which helped in the interaction and tagging of both paper and digital version of documents.

Many other projects could have also been mentioned here in more detail[14][15][17][19][25][24][28], but our intention was only to indicate which direction the development of the digital pen is going.

4.2 Map interactions with digital pen

Since in this master thesis we are implementing also some map interactions, we will present a couple of projects which are related with digital pen and paper technology and printed maps.

Digital Pen Mobile Mapping:

In 2006 there were a project which investigated how smartphones and digital pens can be integrated[12]. The maps were printed to an Anoto dot patterned paper, and a Nokia digital pen were used to interact with it. The digital pen communicated through bluetooth with a smartphone, which accessed an application server through WiFi.

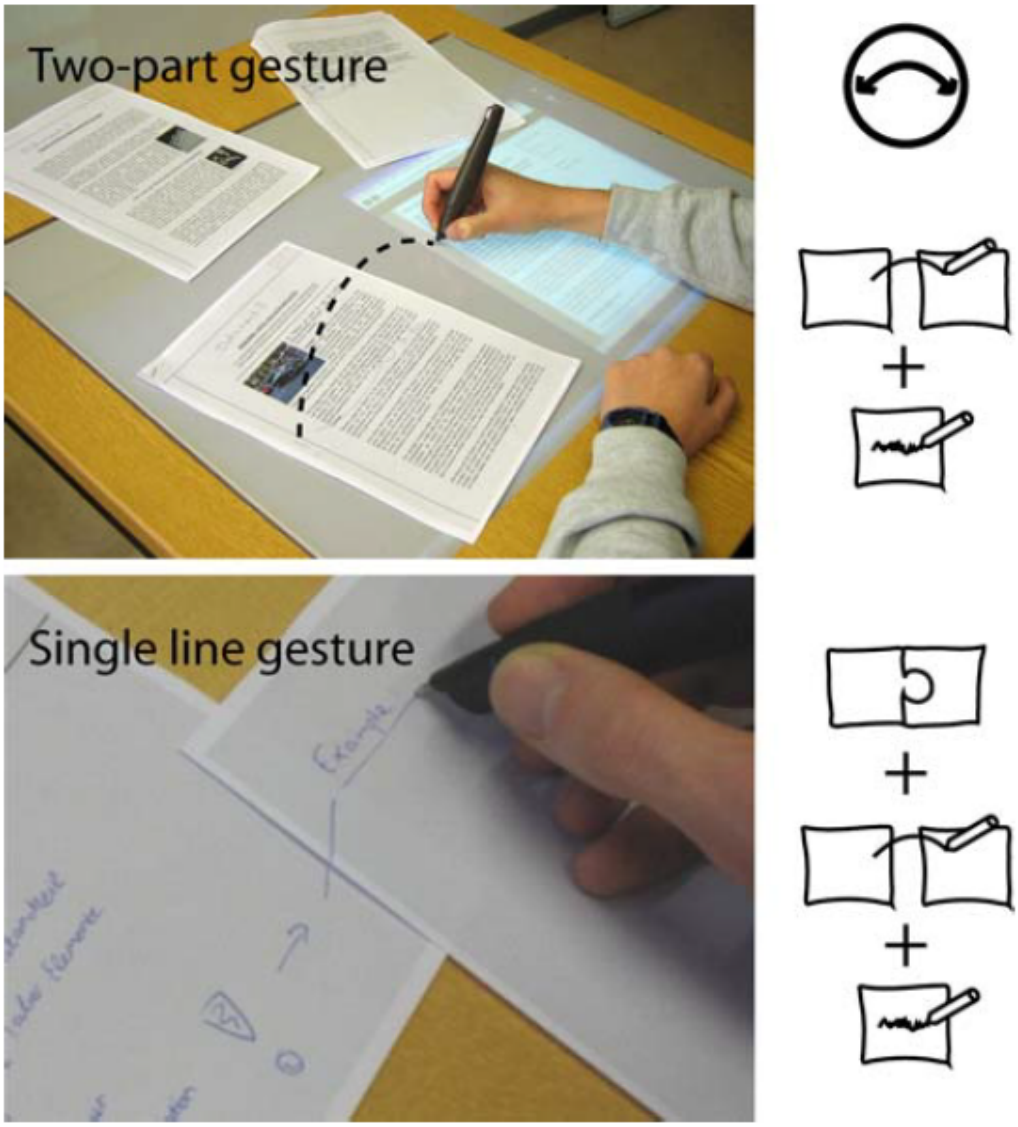

Figure 8: Example gestures of PPUI.

Performance evaluation of digital pen for capturing data in land information systems:

The accuracy of the digital pen makes it as a perfect choice to annotate on the map, even on very high resolution aerial photos. In 2011 an Anoto pen was was evaluated(usability, accuracy, reliability, compatibility and functionality) at updating cadastral maps in a Land Information System[23]. We would like to highlight this research since our research goal is very

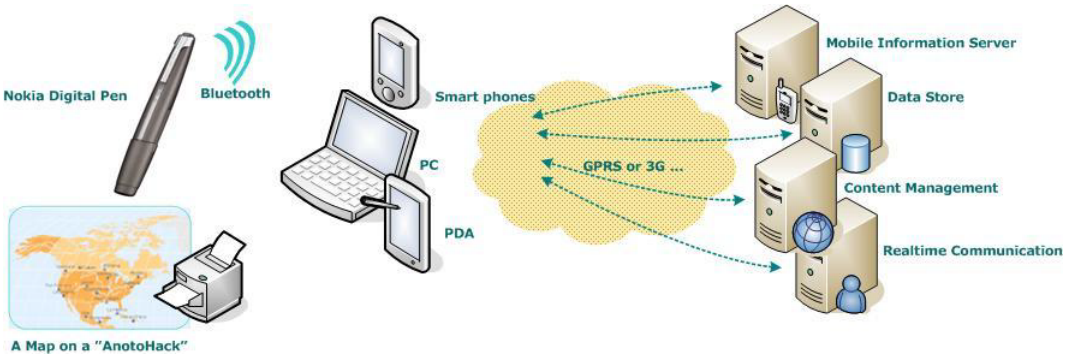

Figure 9: Conceptual model of the system.

similar with this one, with only the difference that we would like also evaluate stylus and touch inputs.

Figure 10: This image has been taken from the thesis documentation: Comparison between the drawing(yellow by free hand, green by ruler), the existing cadastral map (red) and the aerial photo.

Local Ground: A Paper-Based Toolkit for Documenting Local Geo-spatial Knowledge:

Finally there were a study about how digital pen can be used to interact

with a map, making annotations on it and how these annotations can be submitted later on to OpenStreet map[29].

4.3 Stylus and touchscreen inputs

Let's see some other researches which were evaluating touch and stylus inputs.

There was a research how digital pen and touch inputs can be combined together to enrich the interaction with digital documents[20]. They defined new gestures which were the combination of the two input modalities. For example, the user can hold a photo and drag off with the pen to create and place a copy (see Fig.11). We have to note that here not the Anoto digital pen and paper technology was used.

Figure 11: The pen and touch interaction: the user drags of the image creating a copy of the document.

Digital pens were also compared with mouse and keyboard inputs[30], but again here not the Anoto digital pen and paper technology was used. Finally we have to mention a study which were investigating the usability of the stylus input, comparing different types of styli (with five different pen lengths 7, 9, 11, 13, 15 cm, and three pen-tip widths: 0.5, 1.0 and 1.5mm, and two pen widths: 4 and 7 mm).

5 Description of the tool

In this chapter we discuss in detail how the system was realized, starting with the functional requirements we propose an architecture. To explain this architecture we need to speak a little bit about Digital pen & Paper development, the tools what the Livescribe has provided for us, about smartphone and Android OS architecture, and application development, and how to send data to the phone from the pen.

5.1 Requirements specification

In this section we consider all the functional and non-functional requirements of the system. From these we propose an architecture of the system.

5.1.1 Functional requirements

We imagined an application similar with TripAdvisor[10] where the users can annotate places on map, attach a comment, attach an image to and rate it if it's the case. The users can save these annotations and search among them later on. The possibility of the functionalities what we could have added is endless, however we decided to stay with only these. All these annotations should be able to be inputted with with digital pen, stylus and with touchscreen.

More specifically let's see the full list of functional requirements:

- The user should be able to annotate a place, a path or an area on the map.
- The user should be able to attach an image to the annotation.
- The user should be able to write a comment and attach to the annotation.
- The user should be able to save the annotation.
- The user should be able to search among these annotations.
- The user should be able to input an annotation using a digital pen & paper, stylus or touch-screen.

5.1.2 Non-functional requirements

Since the system will be used at an evaluation the non-functional requirements[7] have specially high importance, because the user should not be constrained (or should be as little as possible) with the usability, understandability, predictability, efficiency, responsiveness etc... Finally we have decided to take care especially to the following usability requirements:

- The system should be usable:

  The approach how to annotate on paper and on a touch screen input device can be very different. We must ensure that for users is comfortable and straightforward to to use the tools. We have to take into the consideration that some users doesn't have much experience with smartphones or digital pens, and they should be able to make annotations after a small training.

- The system should be highly understandable:

  Since on evaluation we target non-computer experts too we have to be sure that the users understand the tasks as quickly as possible. It means for example we need large meaningful icons, comments next to the buttons on the paper, textual and audio feedback on the pen etc...

- The system should be responsive:

  The system should react quickly on user interactions, give them visual or audio feedback, by using the pen or the phone. The response time for any user interaction should be less than 500ms.

- The system should be predictable:

  The menu should be logically organized in such a way that the common functionalities should be next to each other. The icons must suggest the functionalities associated with them etc...

- The system should be extensible and scalable:

  In case if somebody would like to continue this project, the system should be highly extensible. We can ensure this by using common protocols at the communication, by good modular program structure and by documenting well the source codes...

5.2 System architecture and design

By analyzing the system we need for surely a smartphone application which handles the annotations, a pen application which collects the annotations on the paper document, a desktop component which accesses to the pen application data and a server component which saves everything into a database.

Let's have a look on the architectural diagram of the system(Fig. 12). We can immediately observe that we have to use 3 devices at the architecture. A digital pen, a smartphone and a local PC which acts as a PHP server, a database server and it also gets data from the digital pen. Let's see all of these devices in more detail in the following sections.

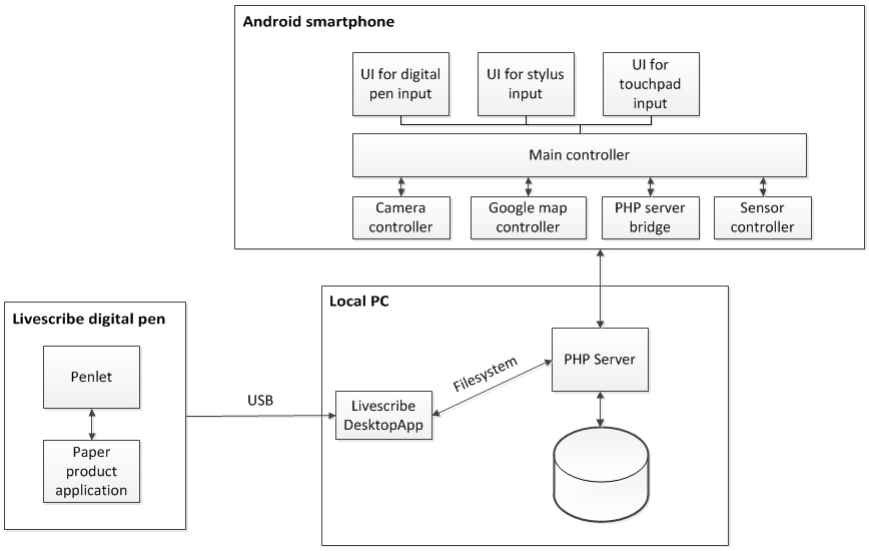

Figure 12: Architectural diagram of the system

5.3 Digital pen & paper development

In this section we see in more detail how the digital pen is integrated with our system. First we introduce the digital pen used by us, how it's working, and how can developers write custom application to it.

5.3.1 Livescribe digital pen

We use in an Echo Smart pen from Livescribe[6][5], which is based on the Anoto[3][4] pen and dot pattern technology.

On the image below you can see how this dot pattern is working: A small camera mounted at the head of the digital pen is able to detect the dot pattern, which encodes the absolute position on the paper. With this unique dot patter it's possible to identify each point in a $60km^2$ imaginary surface (which should be enough for all possible applications written to the pen). These dots are so small that they are hardly visible (theirs diameter is 0.08mm) so they are not disturbing at all at reading or writing.

On the image below you can see an Echo Smart pen, and it's functionalities[5]. It has a built in speaker what we are using to give audio feedback, it has a built in memory storage which can capture the handwritten data and here are stored our application files too. It also has a small display where we can give visual feedback about the internal state of the application. Finally the pen can be connected to a computer with a Micro-USB connector. Unfortunately this model has only this type of communication, so at this evaluation the mobility of the pen device is restricted to a cable length. An important future improvement can be introduced here with another type of pen which

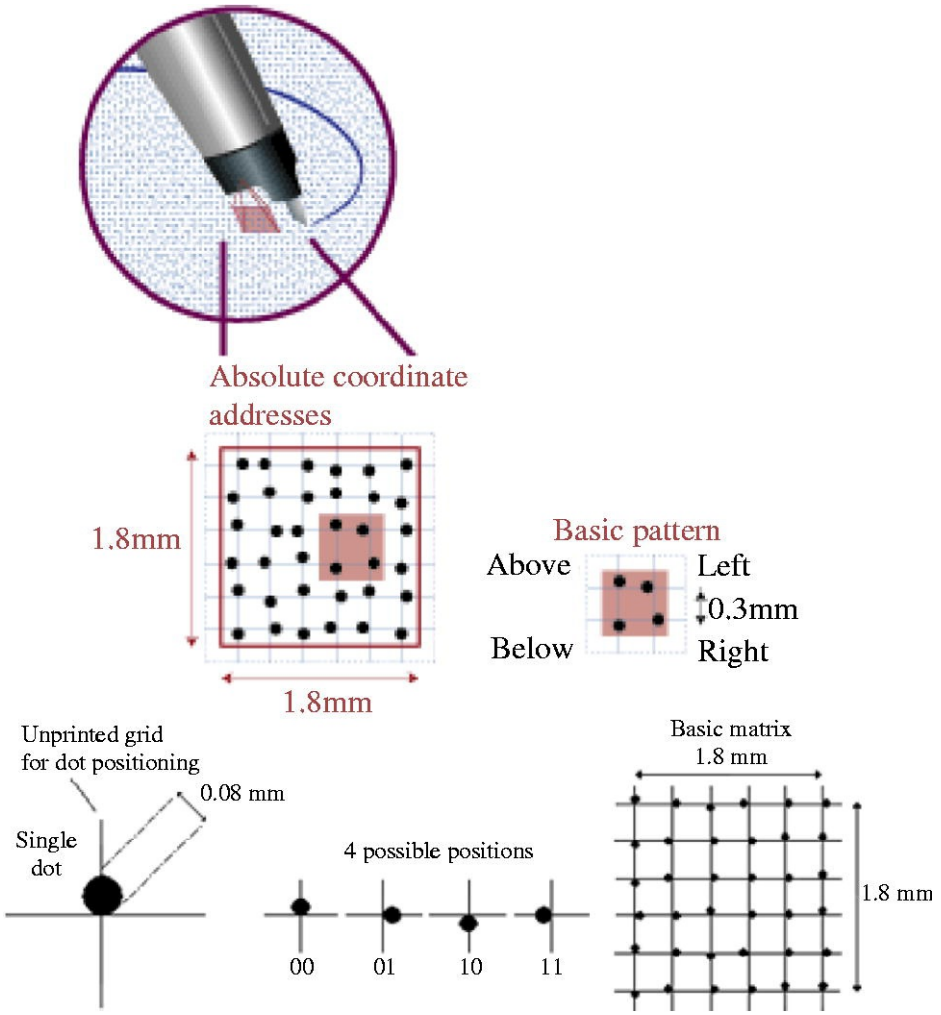

Figure 13: Anoto dot pattern

support wireless communication, enabling the user to not to be constrained physically by a cable connection to a local computer.

The pen can run Java based applications called penlets, which can access the pen resources and stroke data. They also provided tools to develop Paper Product, where the developer can design and print out user interfaces with a printer to a Livescribe Dot Paper. Since both the penlets and the digital representation of the paper product are deployed to the pen, they can interact with each other. The developer can also specify handler function for the different penlet event, and connect it with the paper project. All the data can be stored on the pen, and it can be accessed by a desktop application written in C#, from a PC, when the pen is docked with a USB cable.

Figure 14: Echo smartpen structure

Note: we must note that the Livescribe has discontinued the penlet development project in 2010, and all the forums and blogs were unpublished. During development our only sources were the documentations and codes samples.

5.3.2 Application UI: paper project

We designed our interface to an A3 paper format (The active area is 42cm x 29.7cm). On the Fig. 16 you can see the UI of our paper project. We can immediately see that the UI can be divided to three sections: a map, a control panel, and a comment section.

The most important element of the UI is the map, and we tried to leave a space as large as possible. The map is sensitive to many pen interaction events to be able to correctly distinguish between an annotation, a single

tap, double tap, a stroke data, and handwritten data. These events are handled by the penlet, and they are discussed in more detail in a later chapter.

At bottom we put the control panel. To the left corner we put our control buttons: Delete path, Delete comment, Annotation mode, and Search mode. To improve the usability and learnability of the system we tried to find simple and meaningful icons, and since we don't have a hover option to display more information in tooltips, instead we printed the tooltips below the buttons.

At the right part of the control panel, we have the 5 star rating bar.

We also put a different section for the comments, if for the user is more comfortable to write there instead directly to the map. Here we must note that the orientation of how the comment must be written is restricted. This restriction is due to the built in handwriting recognition.

In general we tried to give as much freedom to the user as possible, so at an annotation nothing is mandatory and the order of the different operations are independent. So it's perfectly correct and equivalent if the user first rates the annotation and then marks it on the map or the other way around.

5.3.3 Application design

Penlet development:

The paper project is the UI of the application, it acts as a view component in a model-view-controller architecture patten. Penlets are listening and

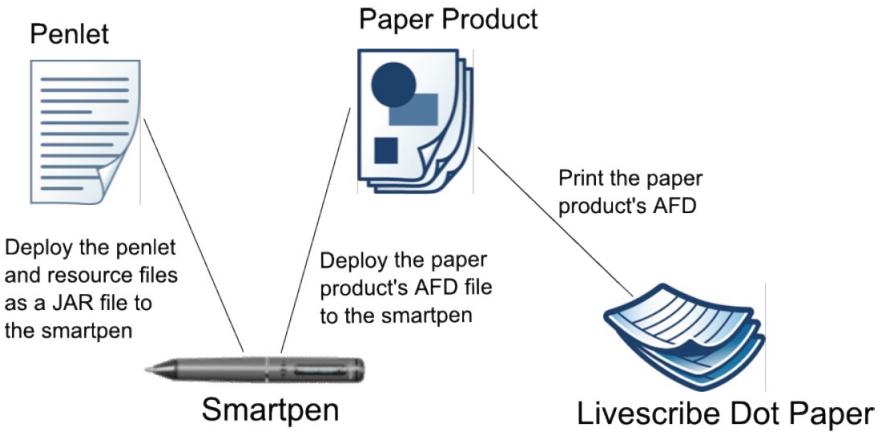

Figure 15: Relation between the paper product, smartpen and Livescibe dot pattern

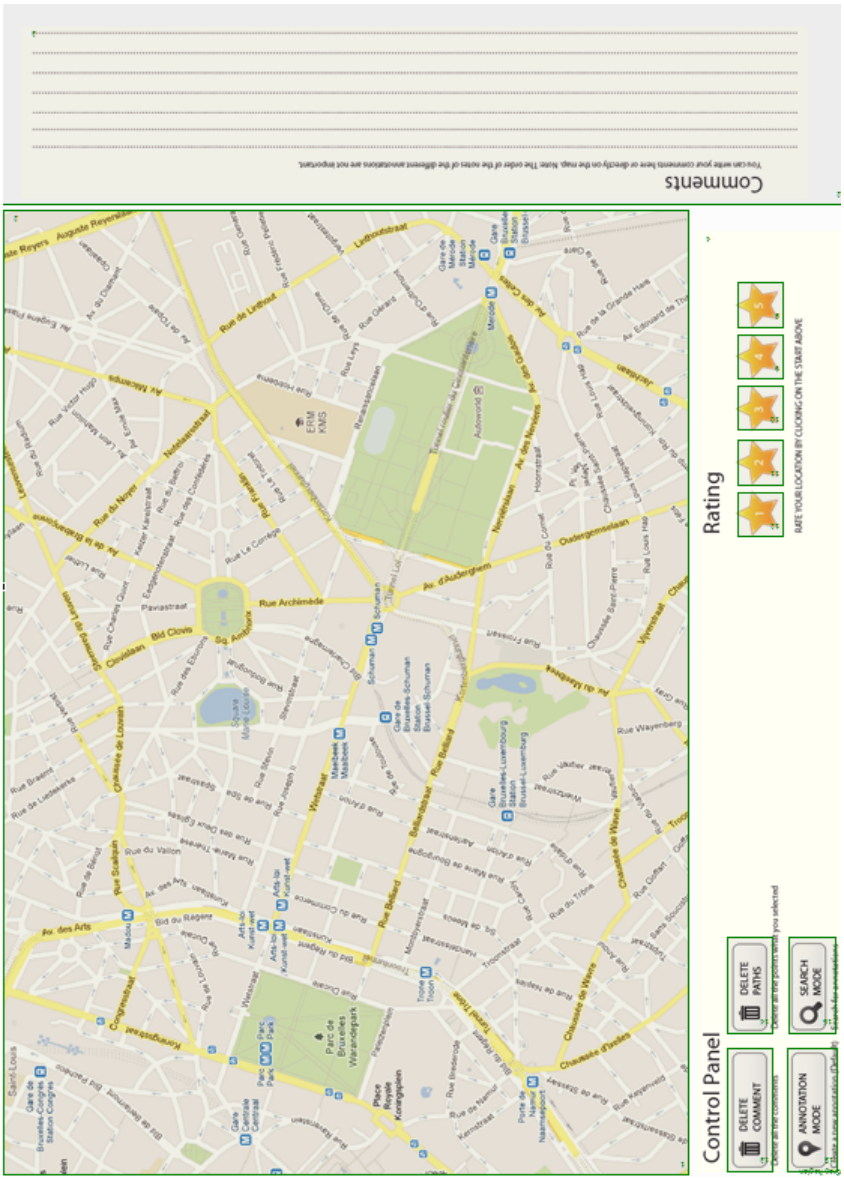

Figure 16: Paper UI of the project

handling the UI events of the paper project, so it acts as a controller and a model.

A pen can hold several penlets, but only one penlet can run in a given time. On the image below you can see the lifecycle of a penlet: initialization, activation, deactivation, destroy.

These lifecycle states has a huge impact on our system architecture, and it determines the way how our penlet communicates with the Desktop

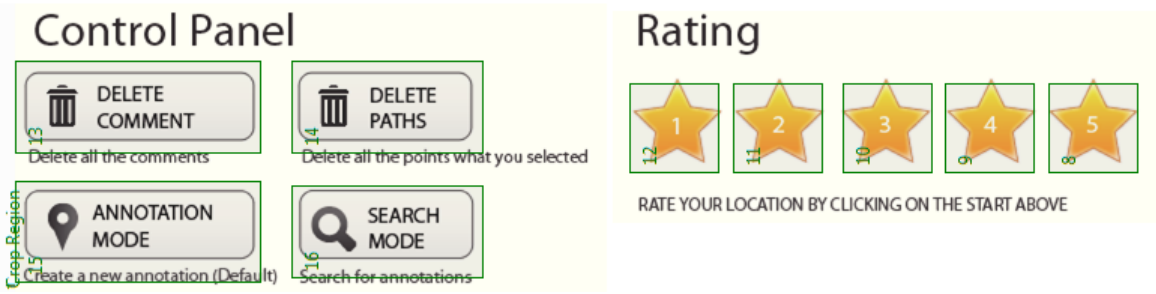

Figure 17: Control panel and rating bar from the paper interface

application. Read more about this at the data synchronization part.

To meet the usability requirements, we designed the system to be scalable, and we documented the code as well, so later on anyone can understand, reuse and add new functionalities. We had to think also about the optimization, since the pen is not designed to process a large amount of data. We are listening to multiple events, which should give user feedback immediately (for example creation of a stroke). Let's see the most important penlet events what makes it possible to detect, analyze and store stroke data. See the full code which handles the events in the annex B.

The user should be able to write comments, draw paths, areas and place a simple marker. We wanted to enable the user to do this on the map directly, so somehow we should be able to distinguish these events. Placing a marker is not an easily detectable user interaction, since the dot what the user makes for a marker can be mixed up with the dot on the letter "i" and with very small path or area. For this we are using the onTap() event handler function

To detect a path we take samples from the strokes. By analyzing a stroke we can determine from it's size whether it's a path / area (the area is simply a closed path) or a handwritten note. When a path or an area is detected, the pen displays a success message on it's OLED screen, and plays a success sound. This way the user always can be updated about the internal state of the pen.

We have four control buttons: Delete the stroke data, delete the comments, annotation mode and search mode. Every time when the user presses a button we give both visual feedback(through OLED screen), and audio feedback.

We use the built in handwriting recognition, which is basically a character recognition, and it tries to match the detected stroke data from a local dictionary, what we can give as an input. Because we are not measuring the

performance of the handwriting recognition, we haven't optimized it this part, and it will not be taken into consideration at evaluation.

When the desktop application accesses the penlet, the penlet is being deactivated, so the data need to be present in some ways. Unfortunately for some reasons, the application variables are no longer available in that moment, so we need to be sure that the data is present all the time in a file. It means that after each relevant user interaction we should reserialize the data and save it into a file, which also has some penality in performance.

we have chosen to represent the data in XML format, to represent the annotations, and since we representing stroke data and some custom data too we have chosen to integrate with inkML[1]. The Ink Markup Language is a data format for representing ink entered with an electronic pen or stylus, so it was a perfect choice for us and we increased the scalability and extensibility of the project too. Below you can see an example inkML representation of an annotation generated by the pen.

```
<ink xmlns="http://www.w3.org/2003/InkML">
  <traceGroup>
    <trace type="path"> 50.312343 4.2462496,  ... ,50.565458 4.3634896  </trace>
    <trace type="path"> 50.312345 4.2562546,  ... ,50.897899 4.3653234
</trace>
    <trace type="path"> 50.365854 4.6645356,  ... ,50.236844 4.3641233  </trace>
  <traceGroup>
  <annotationXML type="annotation">
    <comment> This was an awesome trip </comment>
    <rate> 4.5 </rate>
  </annotationXML>
</ink>
```

Listing 1: An example of annotation in inkML format

An important note here is that the coordinates recognized by the pen are from the paper project coordinate system, and we have to transform them to geocoordinate system (latitude and longitude). The pen was so accurate that we could translate the detected coordinates it with around 3-5 meters in the map.

Desktop application:

The Livescribe also provides a DesktopAPI to access the penlet data (both stroke data and file data). It is written in C#, and runs on a desktop computer and it allows us to create applications that do the followings:

- Access strokes stored in AFDs that are installed on one or more Livescribe smartpens attached to your desktop computer.

- Get smartpen status information, such as battery and memory status.
- Get and set smartpen properties.
- Manipulate data in ZIP files on the desktop computer
- Create a paper product programmatically.

Here we just needed to use the data transfer functionality.

To explain how we exactly transfer data from penlet to the desktop application, we need to speak some words about the penlet lifecycles. On the state digram below the most important states and transitions between the states are listed: initialization, activation, deactivation and destroy of the penlet instance. One of our requirement is, that on a smartphone request the desktop application should pull the filedata from the pen, and should make it available to use by anybody. When it accesses a penlet, first it calls the executeCommand() (see in more detail in the annex A), then it deactivates it. This deactivation had a very important impact to our architecture. It means that on the pen we have to store everything to a permanent storage because on deactivation the penlet looses it's state. This deactivation can come at any moment so it means that we need to refresh the temporary storage with the last version of data continuously, and this has an impact to the performance.

We also tried to continuously read data from the pen, but this led to a dead end too due to the continuous deactivation of the penlet (it takes around 2-3 seconds to activate again the penlet, which is of course makes it unusable in this scenario). We discuss in more detail this process of data synchronization between smartphone and penlet in a further chapter.

5.4 Smartphone development

To be able to annotate easily on a map we have chosen a smartphone which has the largest display. Our choice has been the Samsung Galaxy Note (see the image below) which has a very high resolution (1280x800 pixels). It supports both stylus and a full touch inputs too so we are be able to evaluate both input methods on smartphone. It runs Android OS, so we had to design an application under Android.

5.4.1 Android OS and android applications

Before we present the application design we need to see some details about Android OS and Android applications.

Android applications are running on top of an AndroidOS. This Linux based OS are designed specifically for mobile devices owned by Google. It

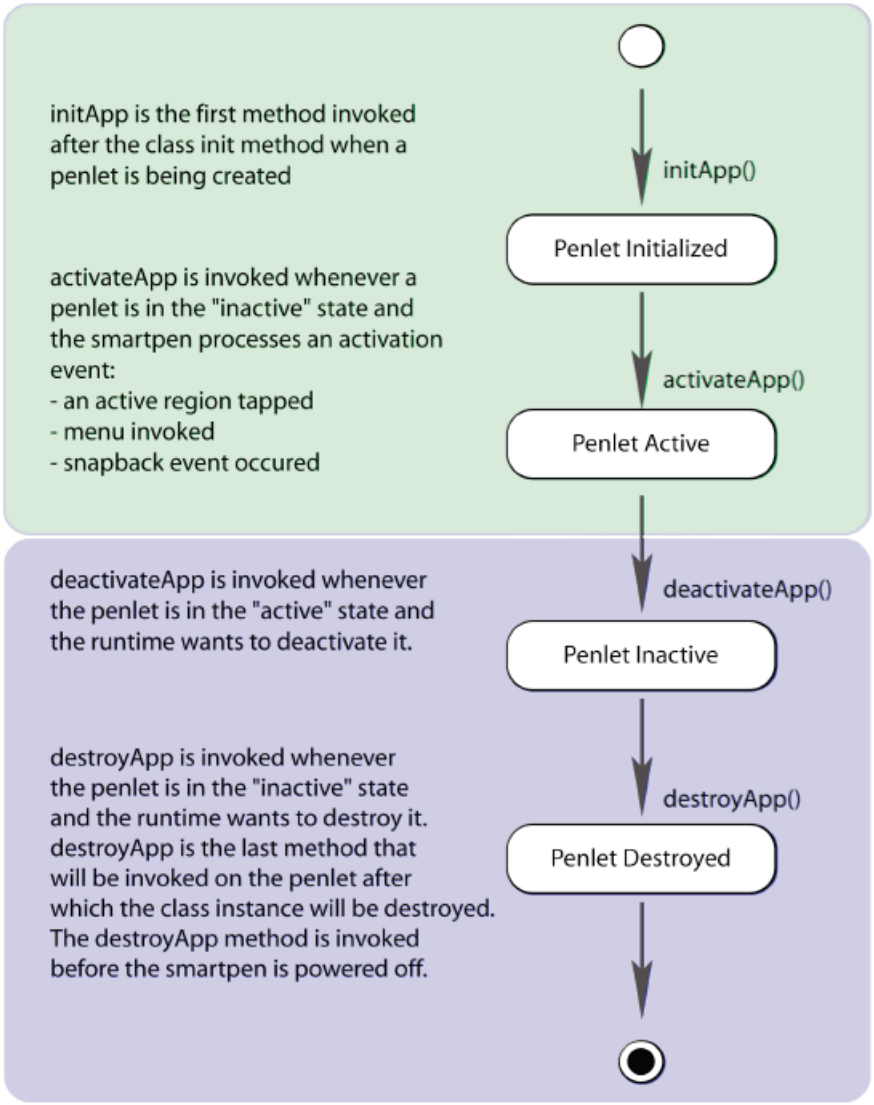

Figure 18: The most important state transitions during the life cycle of the penlet. Image was taken from the Livescribe Penlet documentation

was first introduced in 2005, and in 2007 it become an open-source project under Apache Open Source Project license. I think this is why currently the dominant mobile OS on the market. On the image below we can see how the Android market share has changed over the last couple of years.

Android has a large community of developers writing applications that extend the functionality of the devices. Developers write primarily in a customized version of Java. These applications can be downloaded from the Android Market for no cost or for a small amount of money. As of

Figure 19: Samsung Galaxy note

February 2012 there were more than 450,000 apps available for Android, and the estimated number of applications downloaded from the Android Market as of December 2011 exceeded 10 billion.

Android is a Linux kernel based OS, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvik dex-code (Dalvik Executable), which is usually translated from Java bytecode.

Although the version 4.0 is already launched the current most popular version is 2.3.3 called ”Gingerbread”. On the Figure 3 you can see how the popularity of the different versions are shared.

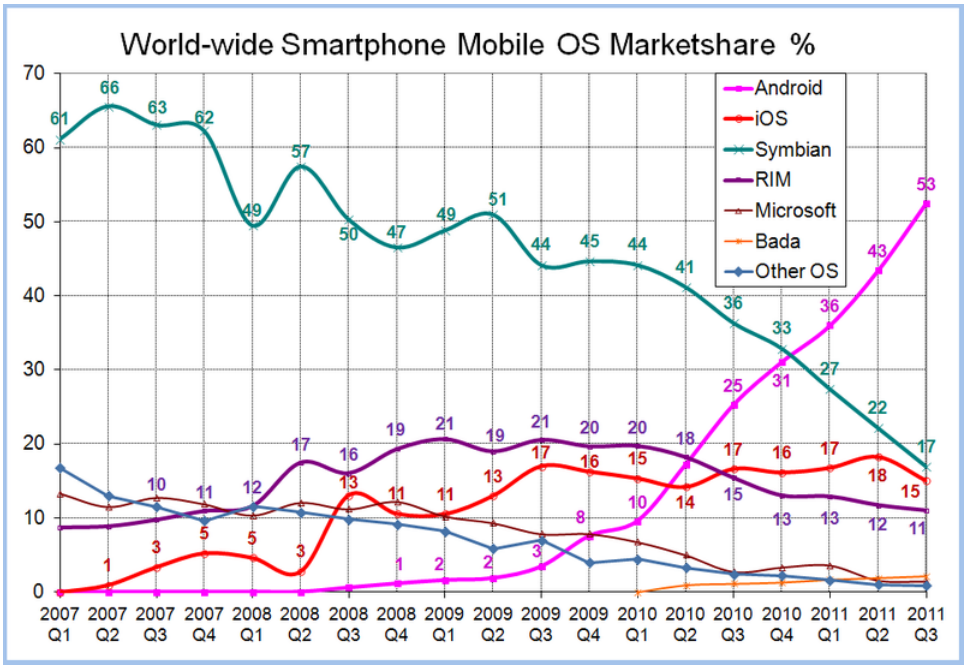

Figure 20: Smartphone Mobile OS Marketshare

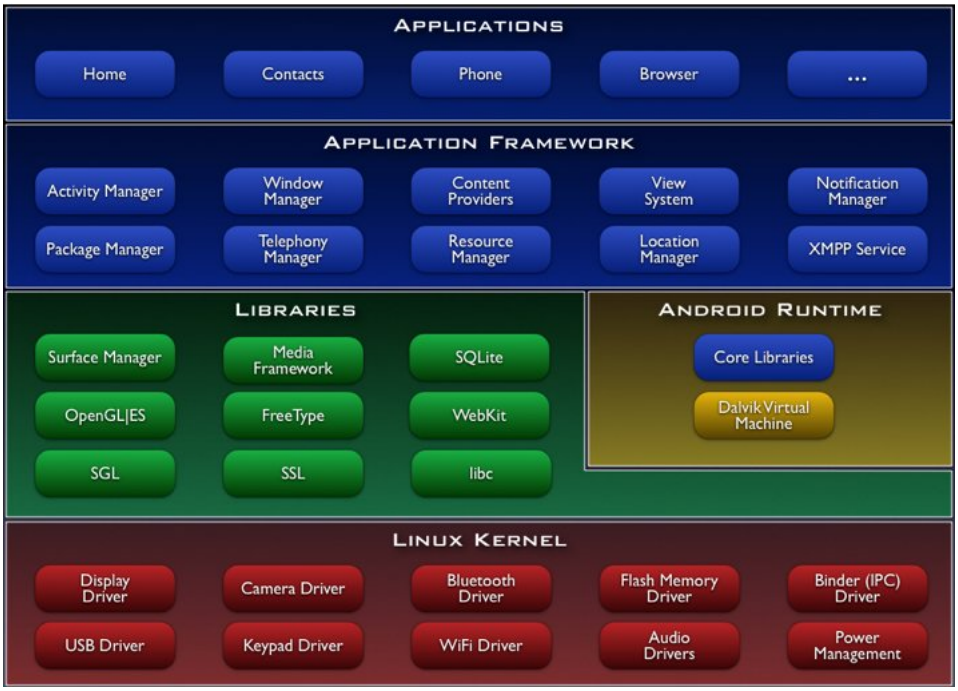

Figure 21: AndroidOS architecture

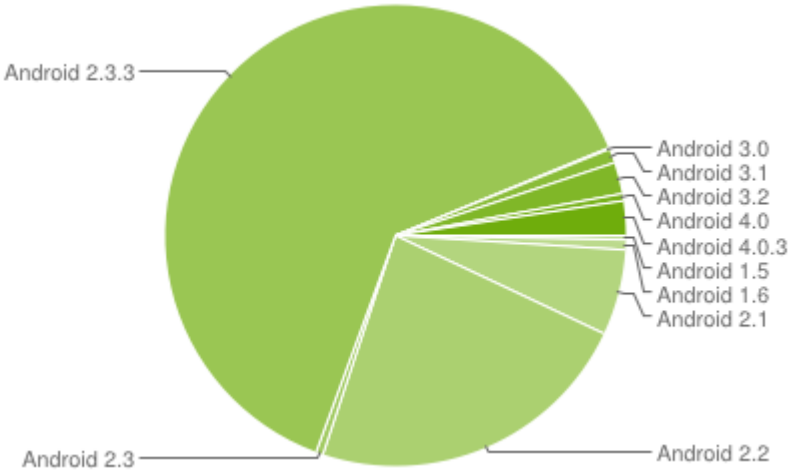

Figure 22: Usage share of the different versions as of 2 April 2012 (image taken from Wikipedia)

5.4.2 Application requirements and UI

This phone application should be the center point in editing the annotations, it should receive data from the digital pen, and save the annotation on the server. It should allow the user to modify the annotation received from the pen, and also should be able to provide enable the users to input both with touch and stylus inputs. These requirements are well defined in the section 3.1. During the evaluation we have to able to distinguish between the users, so we need to have some kind of (very primitive) user management too.

From these requirements and specially from the usability requirements we could design a user interface(see the images below). Since the touch based interface doesn't enable us the "mouse over" functionality hence we don't have tooltips neither, we designed the buttons such that the images should be self explanatory, during the evaluation we ask the participants if they find the UI and the system usable, so we get feedback on this too. On the images below we can see the most important pages and a description next to them.

Login page:

This is the first page on application start. It takes as an input a user id and an input type. The user id is used to identify the annotations made by different users. The user id is a number from 1-21 identifying the participants. We kept the user id 1000 for debug purposes(See the login page on the Fig. 23).

Note: It uses the *login.xml* layout

Figure 23: Login page.

Annotation page:

After a successful login the annotation page is started, where the user can create and save new annotations. We tried to avoid texts and replace everything with icons where it's possible. In this interface the user can mark paths on the interface, set the time of the annotation(the default time is the current), add comment, rate, attach an image, request data from the penlet, search, delete and save annotations(see Fig. 24).

Note: It uses the *annotation_view.xml* layout

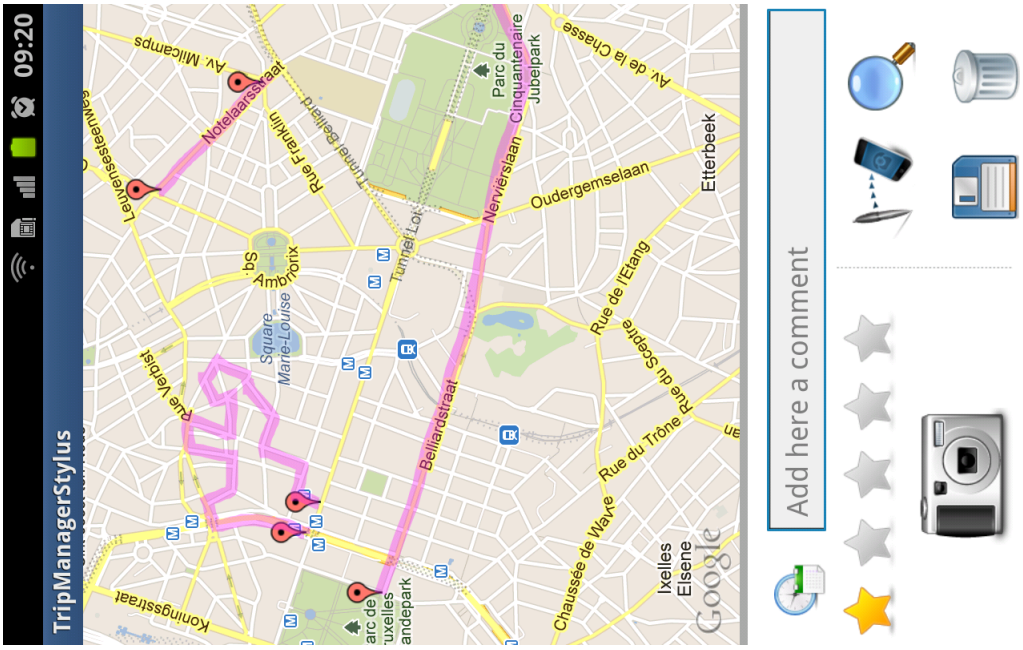

Figure 24: Annotation page.

Search page:

When the user selects the search mode, it will be redirected to this page. Here the user either requests data from the pen or select an area manually. You can see an example of selecting a search area. First the user selects the center of the circle, then the radius. By pressing the search button the result should be displayed(see Fig. 25).

Note: It uses the *search_view.xml* layout

Search results:

Here we can see some search results marked with red. These annotations were made during the evaluation, by 8 user. If the user wants to filter among the search results not only with area restriction but with comment, than he just has to type in some keywords and submit a new query(see Fig.26).

Note: It's enough that only a small fraction is inside the search area, in order to be displayed.

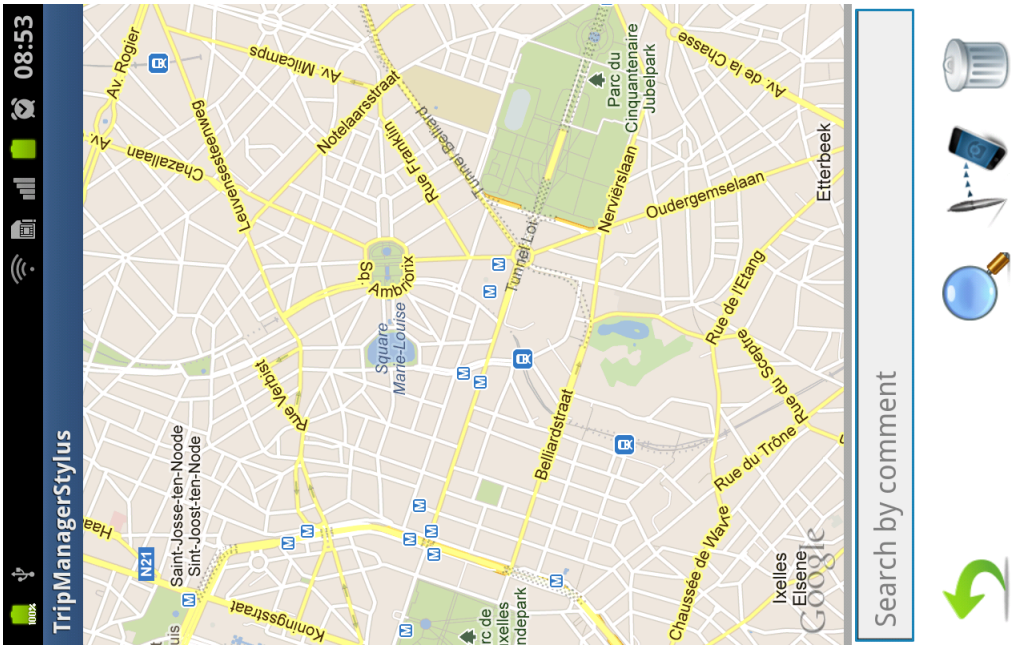

Figure 25: Search interface.

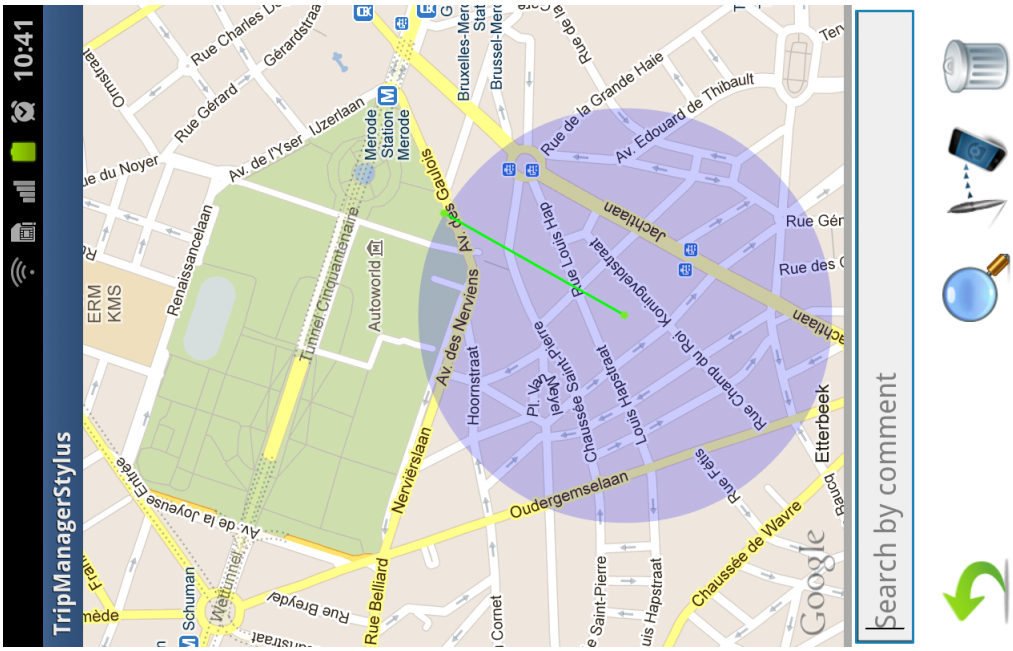

Figure 26: Search area selected.

5.4.3 Application design

At the previous section we described the UI of the application. In this section we dig in more deeply into the core of the application. Let's first

take a look on the class diagram below and the small description of each class

LoginActivity:

Handles the login page, and it uses the *login* layout. (See the previous section about this layout). It checks and saves for a valid user id, and a condition to the ApplicationModel and starts the AnnotationActivity activity. Later on this user id and condition is saved with the annotation to identify each annotation.

AnnotationActivity:

It controls and manages the creation of new annotations and it uses the *annotation_view* layout. Here the user can mark a path or an area on a map take an image, write a comment, set the annotation time, synchronize the data from the penlet, delete the current annotation (reset the screen), and save the annotation. To provide all these functionalities it uses the *map, server, time* and *utils* packages described below.

The *SearchActivity* activity is started from here.

SearchActivity:

It controls the search among the annotations, and it uses the *search_view* layout. By get an area from the penlet, or drawing a circle (by specifying it's center and radius) we can define the search area. Within this search area we can display all the annotations, filtered by a comment. We also can delete the last search result or go back to the *AnnotationActivity* activity;

MapActivity:

A map activity contols the map(this is provided by the Android SDK, it directly supports interaction with a Google map). Every map controller must extend this MapActivity, and we have defined our own map controller to hold our common attributes (see the *TripMapActivity*).

TripMapActivity:

Both *AnnotationActivity* and *SearchActivity* is a *TripMapActivity* which is a *MapActivity*. This class defines the most important utility classes what we use to control a map, so synchronize with a server and a pen, and to parse XML. See about each of these at the end of this list in more detail below.

IMapOverlay:

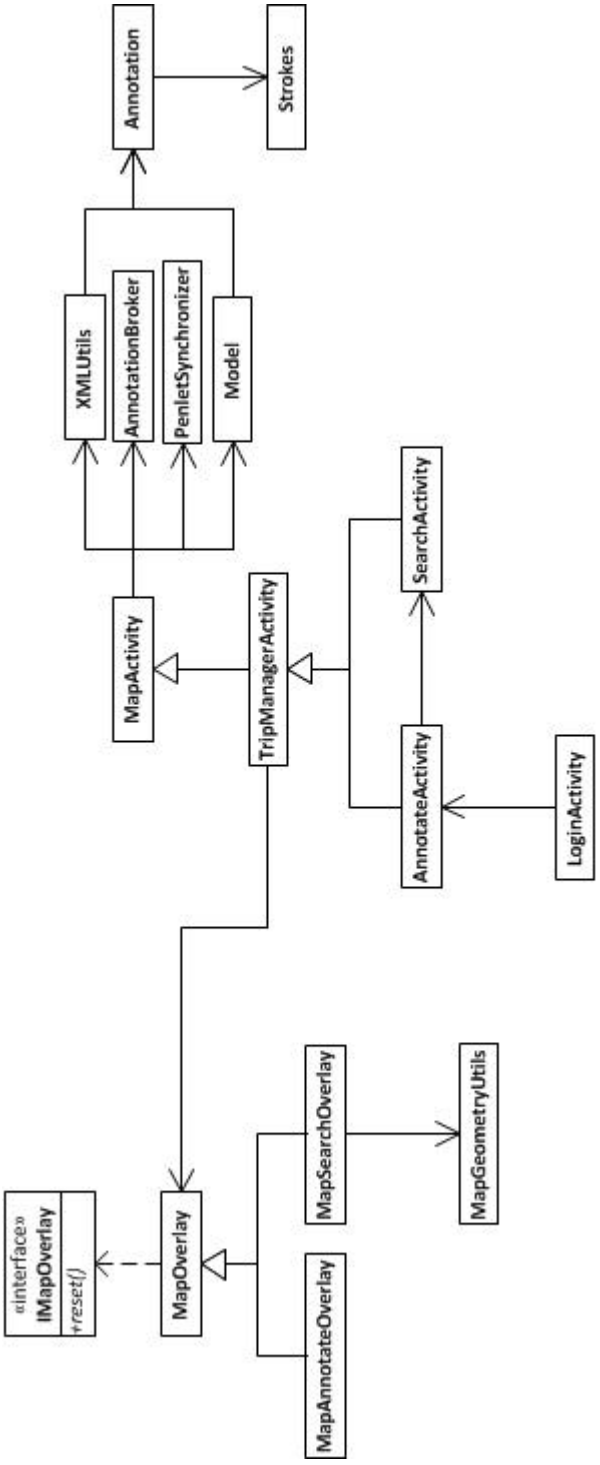

Figure 27: Class diagram of the android application

A simple interface which defines the common methods for a MapOverlay.

MapOverlay:

Is responsible for displaying custom annotation on a Google map. It display a custom stroke data (See more the *Annotation* and *Stroke* classes) and other shapes like lines, points and circles specifying the color, alpha and line width of the annotation.

MapAnnotateOverlay and MapSearchOverlay:

Both classes inherits form MapOverlay. They are the concrete implementation of how to interact and how to display annotations depending on the scenarios. At map annotation, we define a series of point on the map connected with a line, representing an path or an area (on double click it automatically closes the path and fills it as an area).

On search the user selects a search area withing he/she would like to display the search results(annotations). With touch inputs this is done by specifying a central point of the search and a range too.

ApplicationModel:

So far we have described the view part of the application (see the previous section), the control classes which controls the map, overlays etc... . This class is a singleton class which represents the application model from the MVC architectural pattern.

This class stores the userId, condition, annotations, and other utility variables. See in more detail in the annex C.

Annotation:

An annotation is a value object which holds a list of strokes, a comment, rating value, and a path to the image file. The annotations can be displayed by the *MapAnnnotationOverlay*, and the *AnnotationActivity*. The XMLUtility class also can convert an inkML annotation to this value object Annotation.

Stroke:

The stroke class holds a list of GeoPoints (a representation of a point with latitude and longitude), and a flag which tells whether it's a path or an area. A stroke can be serialized(and deserialized) to a String, which is used to store it into the database.

MapGeometryUtils:

Contains utility methods to make geometry computations on a map, with GeoPoints.

- Determining if a point is inside the polygon. Here we are using a simplified version of Ray casting algorithm[8].
- Determining if two polygons intersect each other.
- Determining if a polygon intersects a circle
- Determining if a point is inside the circle
- Determining the distance of two points

XMLUtils:

Using DOM parser we can parse the results from the digital pen in inkML format. See the inkML format defined for annotation in the chapter 3.3.2.

AnnotationBroker:

Saves and searches for annotation on a server, by making HTTP GET and POST requests.

PenletSynchronizer:

Issues a get request for penlet data to the server. See in more detail how this is working in the background in the section 5.6.1.

5.5 Server and database

We mentioned at the previous sections that we use a PHP server for two purposes:

1. To get the annotation data from the penlet with the help of the desktop application
2. To save and search for annotation using a MySQL database

The smartphone communicates with this server by making simple HTTP requests. Due to it's simplicity are not discussing it here but in the annex E you can read the actual PHP code which handles these requests.

To store the annotations and later on, store the evaluation form results, and the measured time results we have created 3 database tables. On the Fig. 28 you can see the structure of the table where we stored the evaluations. We also stored the time needed to make the annotation in the table below (Fig. 29). The evaluation questionnaire results were also stored in database. On the Fig. 30, you can see the table structure.

Column	Type	Null	Default	Comments
id	int(11)	No		Id of an annotation
points	longtext	No		List of points serialized into a character sequence
shape	varchar(50)	No		The clesest match with another predefined shape
accuracy	float	No		the score given by the accuracy test
comment	longtext	No		textual comment attached to the annotation
rate	int(2)	No		an integera rate velue between 1 - 10
user_id	int(10)	No		The id of the user who made the annotation
condition	varchar(50)	No		The condiion when the annotation has been made

Figure 28: Databse table which store the annotations id, stroke, shape, accuracy, comment, rate, the id of the user and the condition used.

Column	Type	Null
id	int(10)	No
name	varchar(40)	No
condition	varchar(40)	No
path_level1_time	float	No
path_level2_time	float	No
path_level3_time	float	No
area_level1_time	float	No
area_level2_time	float	No
area_level3_time	float	No
comment	float	No

Figure 29: The time needed to make the different annotations, here specifically we were interested in the 6 specific annotation, so we created separate columns for them.

5.6 Related developments

In this section we talk about other technical challenges what we faced with during development, and evaluation.

5.6.1 Sharing data between the pen and smartphone

In this section we discuss about how the data from the pen is sent to the phone, and what issues and limitation did it have.

We already spoke about how the annotations serialized to inkML representation is stored into a file on the digital pen. This file can be read by the desktop application which lies in the localPC.

Before we continue we should discuss a little bit more about this data transfer. Initially we wanted to stream stroke data continuously to the smartphone, but all the possibilities to read data from the pen (offered us by Livescribe) first deactivated the application and then read the data. Due to this deactivation we had to store everything in a temporary file, which will be still in the memory after the penlet deactivation. This deactivation had another very important impact in our architecture, namely it takes 2-3 seconds to activate again the application, so we couldn't mimic the streaming with a continuous reading in an infinite loop.

Column	Type	Null
id	int(10)	No
name	varchar(40)	No
age	int(10)	No
gender	varchar(2)	No
computer_xp	varchar(20)	No
multitouch_xp	int(1)	No
dp_usability	varchar(40)	No
dp_responsivity	varchar(40)	No
dp_precision	varchar(40)	No
dp_understandability	varchar(40)	No
dp_quickly	varchar(40)	No
s_usability	varchar(40)	No
s_responsivity	varchar(40)	No
s_precision	varchar(40)	No
s_understandability	varchar(40)	No
s_quickly	varchar(40)	No
t_usability	varchar(40)	No
t_responsivity	varchar(40)	No
t_precision	varchar(40)	No
t_understandability	varchar(40)	No
t_quickly	varchar(40)	No
top_usability	varchar(40)	No
top_responsivity	varchar(40)	No
top_precision	varchar(40)	No
top_understandability	varchar(40)	No
top_quickly	varchar(40)	No

Figure 30: For each of the questionnaire we store the scores given by each participants to the usability requirements and to each conditions.

If we can not stream, than we have to think about something semi-automatic way to transfer data: when the user finished the annotations with one user interaction the phone should receive the data from the pen. We decided that the phone is requesting for the pen data, so with a simple button click we should be able to synchronize with the pen. The problem here was that we had no way to directly access the pen from the phone, so we needed to make a small workaround. Namely only the desktop application has access to the penlet, and if we can make a link between the desktop application and the PHP server we are ready.(since we can make a request from the phone to the server). Finally our solution was the following:

1. The Desktop application waits for a request to read the penlet data. This runs all the time in the background on the PC, and no user interaction is needed. What it does in fact is that it checks in each second if a request has been issued by the PHP server or not.

2. The user makes some annotations on the paper with the digital pen, and these annotations are automatically saved in inkML format in a file and on the pen's memory.

3. The smartphone makes a HTTP GET request to the PHP server for the annotations.

4. The PHP server signalized to the desktop application that a request has been made by the user and waits for 5 seconds (checking in each second) for a return value from the desktop application.

5. The desktop application receives the request for penlet data, and reads the data from the penlet, by deactivating it and copying the content of the temporary file on the pen, to another file which is reachable by the PHP server.

6. The PHP server receives the stroke data from the desktop application in a file, and sends back as a return data for the GET request made by the phone.

7. On the smartphone we get the inkML annotations and we can parse them and display them as *Annotations*

Everything is automated and what the user sees is that only with one button click the annotations are transfered from the pen to the smartphone, and it's automatically displayed.

5.6.2 Calculating the accuracy of an annotation

On evaluation we would like to measure the accuracy of the annotation too. For this we needed an algorithm which compares two polygons and scores it very accurately. On possible algorithm was the $1 algorithm[2], which is a very fast symbol recognizer which can be trained easily. It very powerful because the two symbols can have, different position, different size and even the angle can be arbitrary. To be able to do this is males some elementary geometry transformations: forces the number of points to be constant in the polygon, it shifts the polygon to the 0 coordinates, scales it to fit into a square with edges of unit length, and rotates it until the best match is given.

We know that our polygons are be paths and areas on the map, in a well defined coordinate system, and we are interested how accurately the path or are is annotated if the participant has been given a template. It means that in our case the position, size and angle must be taken into consideration, therefore we had to modify the $1 algorithm in order to take these aspects into consideration too.

Another important change in the algorithm that the strokes in $1 are ordered, and for us it's important to recognize the paths / areas even if they reversed the order. We solved this issue by reversing simply the templates and running the algorithm on them too.

We had further issues with the areas, since an area is a closed path. But the participants most probably will start in different places the annotation of an area. We solved this issue by determining the closest point in the sample to the starting point of the template, and shift the elements in the sample to get the best match with the template.

Let's see the most important code sample of the recognizer on the Fig. **??**, but you can read the full code in the annex D. The first thing what we do is to "resample" the sample, forcing that the nr. of points in the sample to be exactly the same as in the template stroke. After that we go through all of the templates and calculate the average distance between the template and the sample(between each point of the sample and the template since they have the same number of points). Finally we calculate the score, simply just by dividing the smallest distance with 100, to bring down the values to fit to the interval [0,10].

```
public Result Recognize(Vector points)
{
  // Forcing the number of point in the sample to be constant
  points = Utils.Resample(points, NumPoints);
```

```
  // display the sample
  model.helperDisplay.add(points);

  // the template id of the best match
  int templateId = 0;
  // distance between the template and the sample
  double distance = Double.MAX_VALUE;
  for (int i = 0; i < Templates.size(); i++)
  {
    Template template = ((Template)Templates.elementAt(i));
    double d1 = Utils.rotatedPathDistance(points, template.Points);
    double d2 = Utils.rotatedPathDistance(template.Points, points);
    // get the min of the two distances
    double d = (d1 < d2)? d1:d2;
    if (d < distance)
    {
      distance = d;
      templateId = i;
    }
  }
  // we just devide it with 100 to fit into the interval [0-10]
  double score = distance / 100;

  //display the template
  model.helperDisplay.add(
          ((Template)Templates.elementAt(templateId)).Points);

  return new Result(((Template)Templates.elementAt(templateId)).Name,
                      score, templateId);
};
```

Listing 2: The modified recognizer function, of the $1 implementation.

On the Fig. 31 we can see the template stroke with red, and an annotation made by a participant with touchscreen input.

5.7 Summary

In this chapter we defined and described in detail the architecture of the system used to evaluate the participants. First we have seen the functional and non-functional requirements of the system. From these requirements we designed the system architecture, describing each component and sub-system and the communication of the different devices.

We discussed about how to develop an application to the digital pen

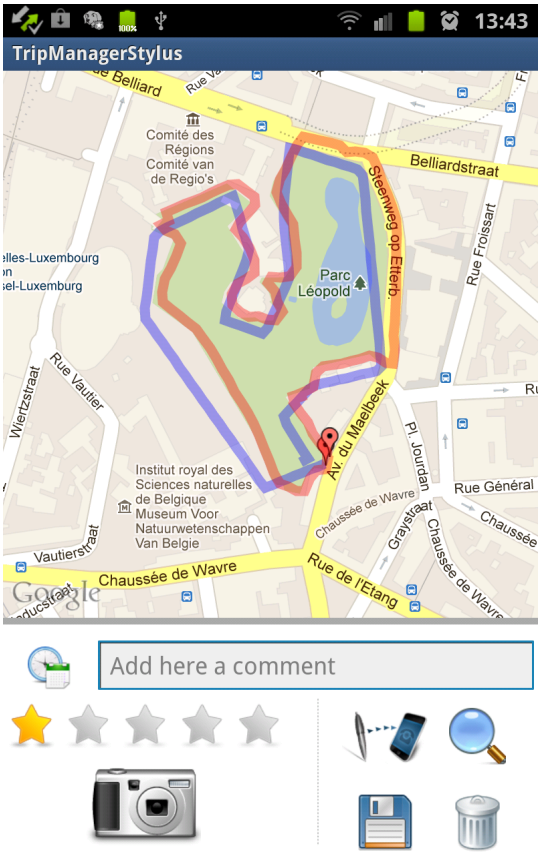

Figure 31: With red marker the template while with blue the annotated area with touchscreen.

(penlets), how to design and print out dot patterned user interfaces and how to access this data from a desktop application. We have shown in detail how our penlet and the paper project can be connected.

We also discussed about the AndroidOS and about the development of Android applications, and in more detail about our application which handled the annotations inputted both with digital pen and touch inputs. A separate section were kept to the communication between the penlet and the Android application. We discussed the issues what we have faced with and our workaround. We also have seen a workflow of this communication using an inkML format.

In the last section we have shown the PHP server which handles the annotations and helps us to transfer data from the pen to the phone.

6 User evaluation

In the previous chapter we have discussed about the system architecture, design and implementation. In this chapter we will focus on the evaluation of the system. In fact the focal point of the thesis is this evaluation. What we have done so far is just the preparation of the system to be able to perform the evaluation.

In the followings we will first discuss how we planned and prepared the evaluations (we will see a concrete example too). We will also discuss about the participants, how we grouped them, and how we adapted the system to be able to evaluate with it (user login, how we measured the accuracy etc.). At the last part we will analyze the gathered data, and we will try do draw conclusions.

6.1 Preparation for the evaluation

6.1.1 Goals of the evaluation

In this section we will discuss about what we wanted to measure. Since the system is quite complex, it allows the users to make annotations on a map with three different conditions, save them, search among them, attach images etc. We wanted to compare the usability of each conditions, so we had to restrict our evaluation to only those aspects of the system which appears at each condition. So for example drawing a path is possible using each condition, but saving it although the system supports digital pen input, is happening through smartphone and not using the digital pen.

We were interested mainly in the usability of each condition, how fast, accurate, responsive, precise, easy to understand and to learn, they are compared to each other. Although the interaction with the map, an annotation itself can be very complex it can be decomposed to elementary tasks like drawing a line or an area, or writing a comment, so we are targeting these elementary tasks to measure.

Since our participants will have different background, age, experience, language etc. we had to make sure that at the time when we measure them each of them get some kind of training. Therefore we started each evaluation with training where at the first part we explained them how the system was functioning, show them examples (this took up about 5 minutes), and after that we asked them to annotate a path, an area and to write a comment.

Note: It was important to highlight to the participants that it's not them or their performance what is going to be measured, but how usable the different conditions are.

We also wanted to know how the conditions will perform if we increase the level of difficulty of an elementary task, so we selected three different paths with increasing length, and complexity, and three different areas. At writing we were only interested in how efficiently the users can write with the given condition.

So far we have measured the raw performance of the conditions in different scenarios, but we were also interested about the participant's opinion. We asked them to rate each conditions from 1 to 5 (1 strongly disagree and 5 strongly agree). We tried to target those non-functional requirements which were listed in the section 5.1.2.

The question were the following:

- I found the system usable: Here we were interested in the subjective opinion of the user about how usable the system was for him/her from a general point of view.
- I found the system responsive: Did the system give me feedback about it's internal state. Did I know all the time what was going on, and how can I correct if I made a mistake.
- I found the system precise: What was the precision of the tool when I was drawing the paths, areas.
- The system is easily understandable: Was it straightforward how to draw a path, how to delete the annotation if I made a mistake, how to save it?
- I found the system quick: How easy was the process of annotation with the given condition. Was it quick to annotate a park or a street?

Since giving a mark to each of these usability requirements are highly subjective we also wanted to somehow double check them, so in case of each usability requirement we asked their opinion again, but this time we asked them to select only one from the three conditions, which they think it's the most usable, responsive precise etc.

We also collected some personal details about the participants, namely their age, computer experience level, whether they had experience before with digital pen, and whether they are left or right handed(see the full evaluation form in the annex G).

6.1.2 Participants

We evaluated the system with 21 participants, with different computer experience, smartphone experience, digital pen experience, age and gender. We also asked them whether they are left or right handed, but incidentally all of them were right handed. On selection of the participants we focused more on their computer experience level, and we selected them such that three equal groups could be formed: 7 beginners, 7 intermediates and 7 experts.

Some of the participants were Hungarian not English speakers while others although were from all over the world they spoke English in a level which was enough for testing.

6.1.3 Additional modification in the system

In the previous sections we talked about how we decided what to measure, how we prepared the logical flow of the evaluation and how we selected the participants. Now in this section we will talk about how we prepared the previously developed and tested system, to be able to use at the evaluation.

User login:

We stored every stroke data in a database, to be able to search among them, and later on to be able to analyze their precision. We also wanted to identify each annotation later on, to be able to assign to control groups, and calculate statistics on these groups. We also wanted to identify the condition which was used at the annotation. Therefore we created a login page, where the user can enter it's unique identifier and the condition used.

Guidelines:

We asked the users to annotate paths with increasing level of difficulty, but somehow we had to mark these paths both on the paper map and on the digital version of the same map. Therefore we created guidelines which helped the users in the annotation process, see on the image below. With the areas we had an easier tasks since we selected parks which were already marked on the map with green. On the images below we can see all the six basic annotation tasks (3 paths and 4 areas) with increasing level of difficulty, length and complexity.

Note: These guidelines where used when we calculated the precision of an annotation.

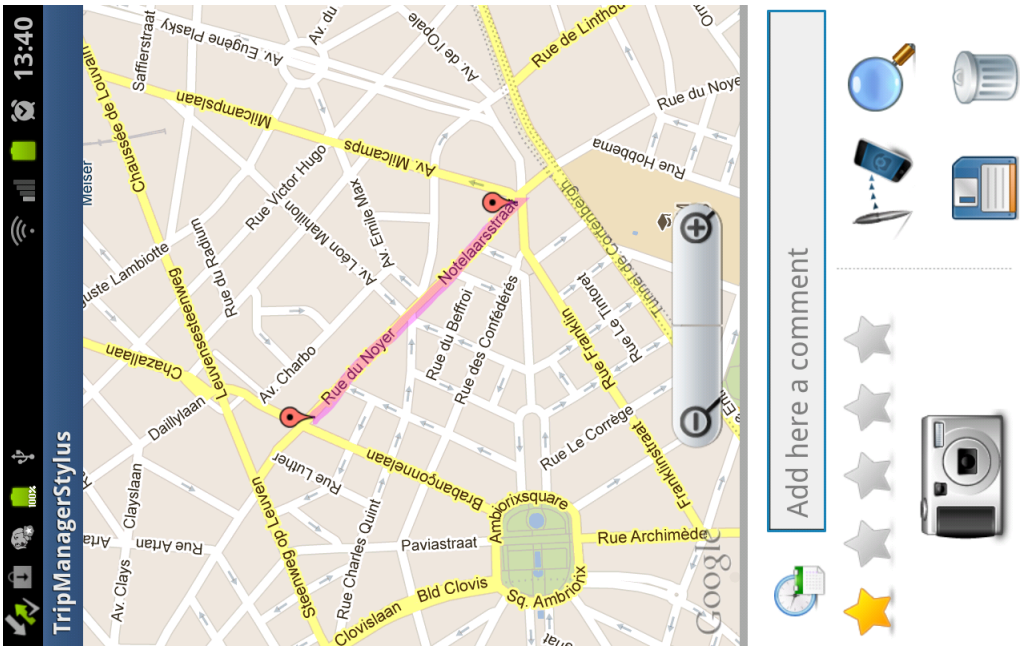

Figure 32: Path with complexity level 1: The most simple path, which can be annotated with only a simple straight line.

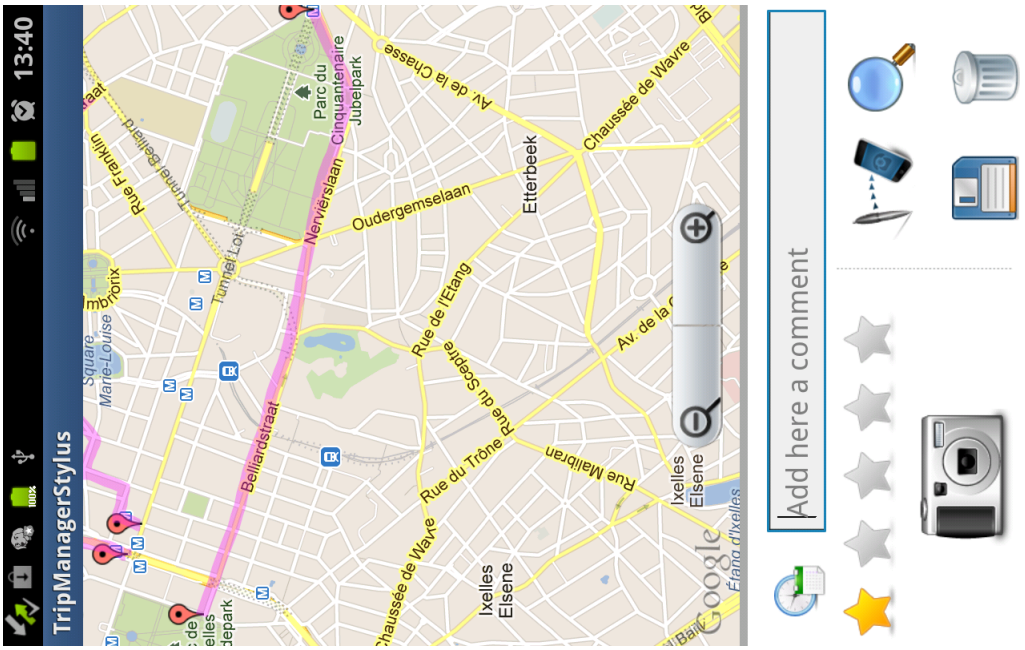

Figure 33: Path with complexity level 2: Still a relatively simple line. An addition level of complexity comes from it's size, forcing the participants to zoom in/out on the smartphone.

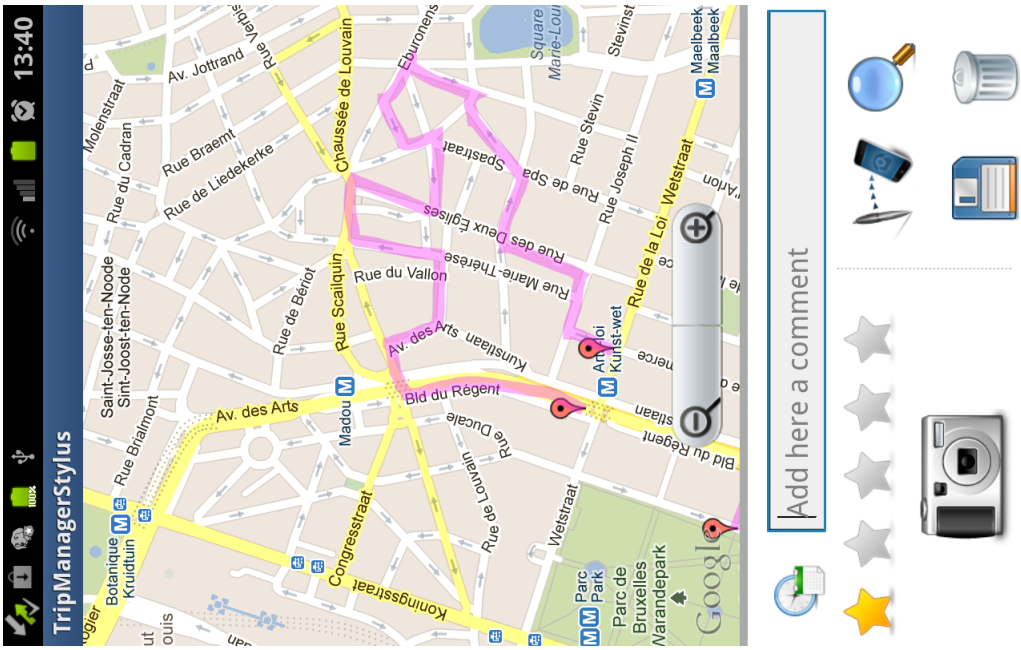

Figure 34: Path with complexity level 3: Here we were interested in the precision of the tool, and of course the time needed to draw, so we selected a path which high complexity but still we were not forcing the users zoom in/out.

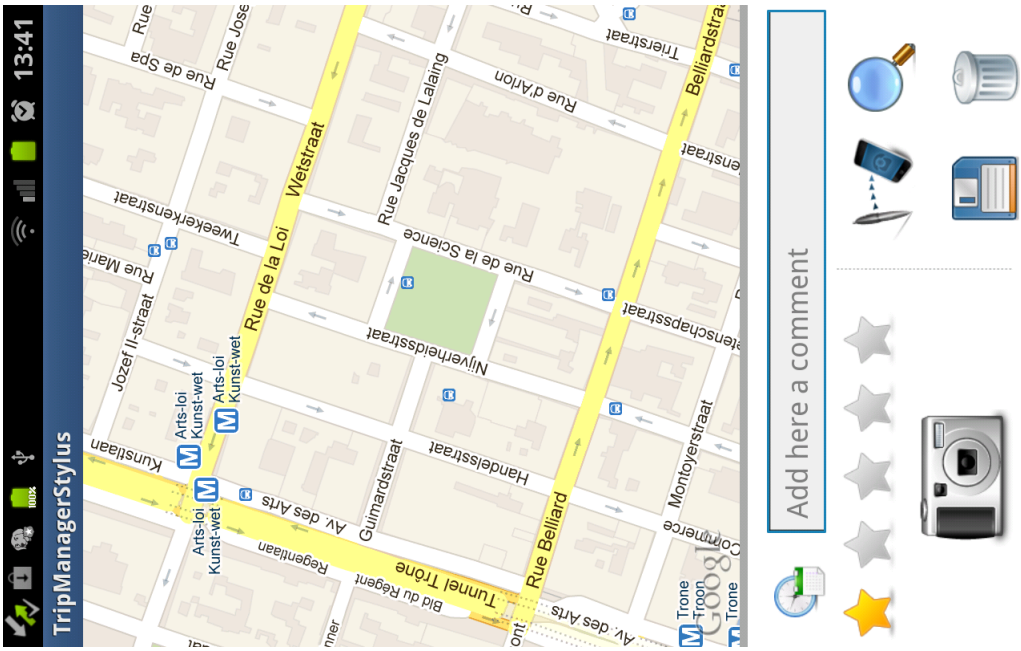

Figure 35: Area with complexity level 1: Simple area, which can be easily annotated with each condition.

Figure 36: Area with complexity level 2: Still an easy are to annotate, but with an additional level of complexity brought in by the rounded corners. If the Participant wanted to annotate it very accurately than he/she had to zoom in spending more time with the annotation.

Figure 37: Area with complexity level 3: The most complex park, which has even a concave shape.

Calculating the precision:

We also had to prepare the system to be able to calculate the precision of an annotation. Initially we were searching for an efficient algorithm which can compare two polygons and tell us how similar are they. There were one famous algorithm called $1, which is used as a symbol recognizer, and gives a score between 0 and 1 (0 means not match has found, 1 perfectly similar) to the similarity of two polygons. Since it is designed for recognizing symbols regardless of their size, position, and angle, it has to make some geometrical transformations both on the sample data and on the templates too. These transformations are not useful for us, since we already know that position, angle and size will and should be the same. Finally we have decided to rewrite the algorithm, such as it skips the geometrical transformations and it gives simply a score grater than 0 (in our case 0 means the perfect match), to be able to calculate the smallest differences between the template and the annotation made by the participant. In the table below you can find some example values of precision.

Accuracy	Description
0-1	The sample almost perfectly matches. This is very rare, it appears only at very simple shapes only
1-2	Very precise.
2-3	Precise
3-4	Acceptable
4-6	Quite inaccurate, but still recognizable
6-10	Inaccurate, maybe the annotation contains some points which were added by accident.
10+	Annotation error.

Table 1: Since we have rewritten the $1 algorithm to our needs, we also changed the scale on which the accuracy was measured. Here we can see some examples of the measured accuracy and the meaning of this score.

You can find more details about the modified $1 algorithm in the section 5.6.2.

6.1.4 Evaluation run

We took care of each participants to get the same information about the system and the same training, so we prepared the exact steps of an evaluation run. You can read this document in the annex F.

Note: An evaluation took on average about 45-60 minutes.

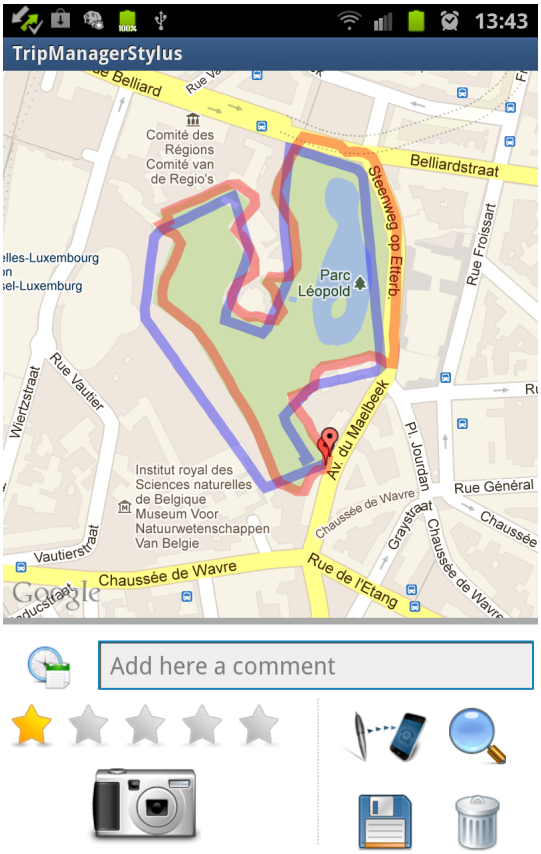

Figure 38: Red marker indicates the template, while the blue one the annotated area with touchpad.

In the previous section we discussed about how we prepared the evaluation by defining what and how to measure, we also discussed about the selection of participants and how we orgranized them into different control groups. We also have discussed about how we prepared the system for the evaluation to be able to analyze the data. Now in this section first we will talk about how we prepared for the analysis, than later on in a different section about the analysis itself, where we will speak about the concrete results.

6.2 Preparation of data

After finishing the evaluation with every participants, in the database we had more than 600 annotations saved. This is because during the training we asked the users to save some annotations (path between train stations, an example are etc. see section 6.1.1), and 6 more annotation / user with each condition. Clearly in this data we have some annotations which shouldn't

be taken into consideration, and we expect that we will have some outliers which will have negative effect on the precision of the measurement. Let's see in more details how we manipulated the raw data gathered from the participants.

Calculating the precision: As we have seen already we used a modified

version of the $1 algorithm to rate the accuracy of an annotation. We had iterate through all the annotations trying to recognize the path or area drawn, and give a score to their accuracy. We saved this data to database next to the annotations.

Automatic and manual error detection:

We had at the beginning more than 600 annotations, from which only 378 valid annotations are expected (3 paths + 3 areas * all the 3 condition * 21 participants). We had to delete those annotations which had been done during the training, and some annotations which were saved mistakenly. Furthermore we had a couple of annotations which were so inaccurate that we considered them as outliers (it occured only a few times). These annotation had been scored with more than 10.

After filtering out every annotations we made a last manual check (one by one we displayed them to see if the array of points contained some points which shouldn't be there). After the automatic and manual filtering we remained with 341 valid annotations.

Delivering everything to database:

During the evaluation we had not just saved the strokes but measured the time needed to perform the tasks, and we had an evaluation form collecting the participant's subjective experience during the evaluation. For this we had created two more database tables.

6.3 Analysis

This section is the focal point of the whole thesis, here we will discuss about the results of the measurements and the evaluation forms. First we will start to analyze the time needed to perform the tasks, then we will analyze the accuracy of the strokes, finally we will see the results about the user evaluation.

6.3.1 Time

For each of the conditions we asked the users to draw 6 strokes (3 paths and 3 areas) with increasing level of difficulty. For each of the strokes we

measured the time elapsed between the moment when they started the stroke and when they finished it. For this we used a chronometer with an accuracy of a hundredth of a second.

Significance:

Before we analyze the different user groups, and draw some conclusions, let's see if the data is significant or not. On the Fig. 39 we can see the boxplot of the three conditions. By experience we saw that the most complex path was a very good indicator to compare the accuracy of each conditions, hence we will use the level 3 times to remain consistent.

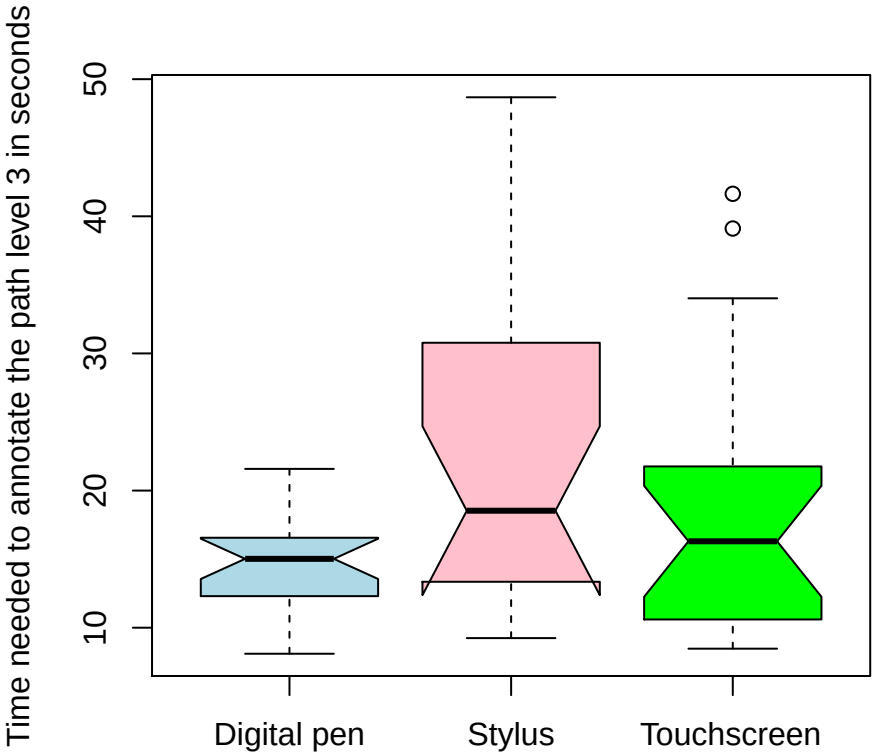

Figure 39: Boxplot of the level 3 times in terms of each condition

We run a paired t-test to see if there are any significant difference between the samples, and we found as we expected the digital pen and stylus inputs are significantly different (p is 0.01), but the digital pen and touchscreen pair and stylus and touchscreen pairs are not (p is 0.11 and 0.26).

User groups:

Let's see the time results of each conditions in terms of experience with computers (see Fig. 40). The first thing what we can observe is that the time needed for the different groups relates to our expectations: the beginners were slower with each condition than the experts and intermediates. In case of the digital pen this time difference was not that huge since the digital pen highly resembles to the ordinary pen.

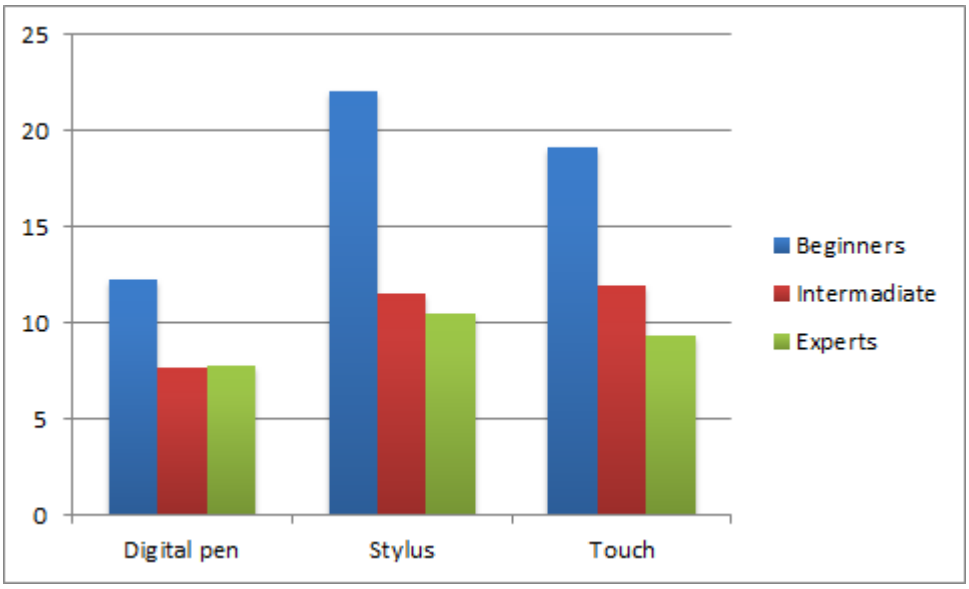

Figure 40: Compare the time needed to perform the tasks, in case of each condition in terms of computer experience.

Some other interesting results can be observed if we select our user groups based on age. Let's see the results in terms of age on the Fig. 41. In case of each condition we can observe that the older generation were much slower then the younger one. This difference was more accentuated in case of the stylus input and less in case of the digital pen.

We also grouped the users based on their experience with smartphones. On the Fig. 42 that indeed those who claimed that they have experience with smartphone were faster. And again in case of digital pen we have less difference, while in case of stylus it's more accentuated. We must also note, that this result is very similar with the previous grouping based on age, since the younger generation was who had more experience with smartphones.

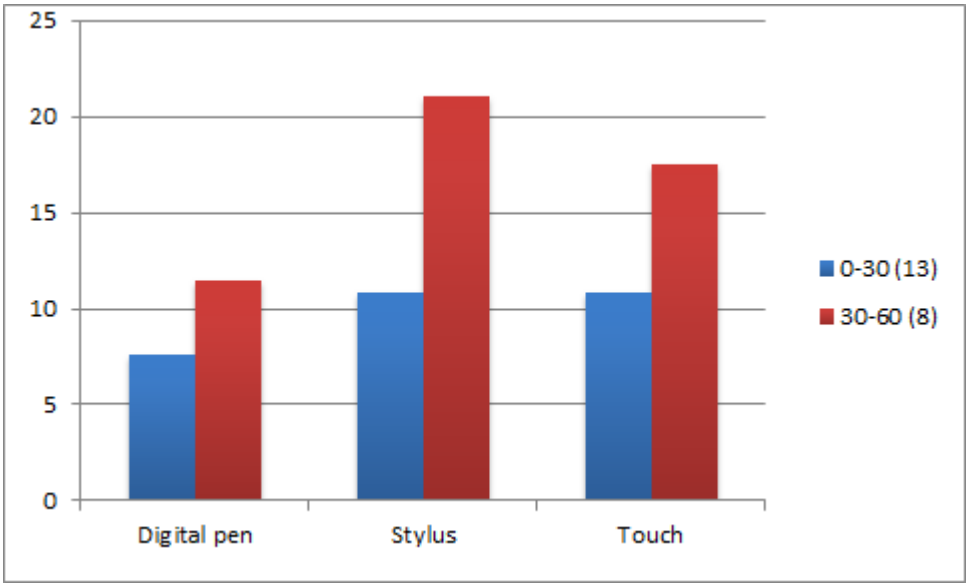

Figure 41: Compare the time needed to perform the tasks, in case of each condition in terms of the user age.

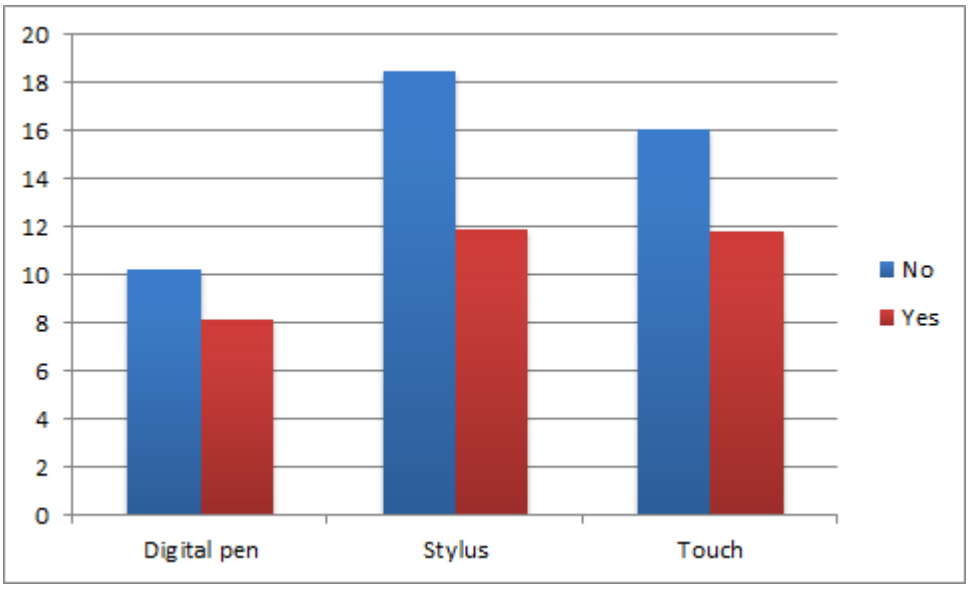

Figure 42: Compare the time needed to perform the tasks, in case of each condition in terms of the participants experience with smartphones.

6.3.2 Accuracy

We saw how much time it took the annotation, now let's see the same statistics about the accuracy of the annotation.

Significance:

First let's see the accuracy on the boxplot below (Fig. 43). As we can expect from the boxplot there is a significant difference between the accuracy of digital pen and stylus (p = 0.0002) and between digital pen and touchscreen inputs (p = 0.0005).

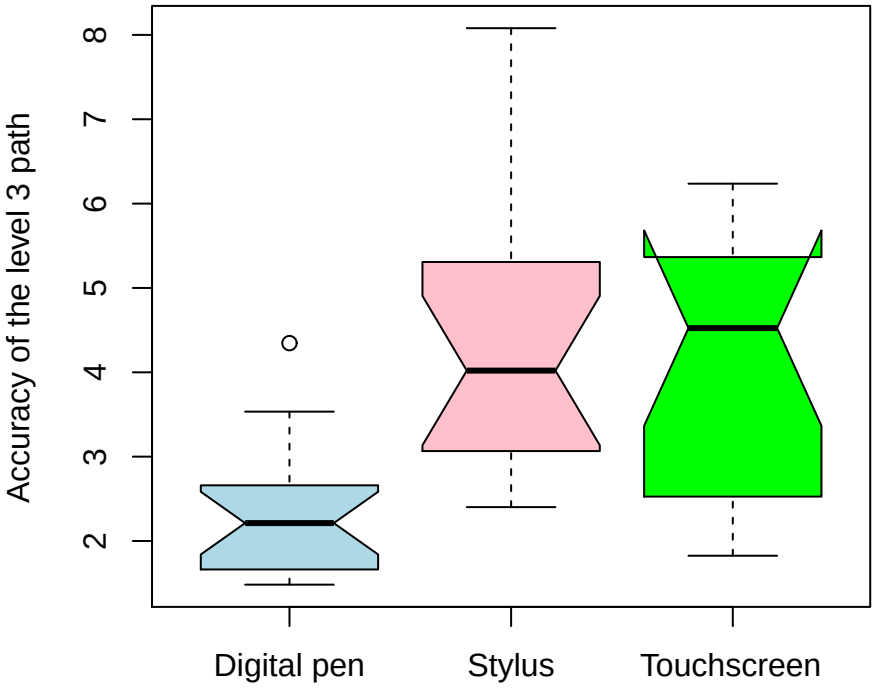

Figure 43: Boxplot of the level 3 times in terms of each condition

User groups:

Now let's see the results based on user group experience. On the barchart below you can see the accuracy of each conditions in terms of the computer experience(Fig. 44). Before we analyze the data we have to note that based on the average accuracy, although digital pen's accuracy is in general better than the other two conditions accuracy, in general we found

that each conditions accuracy is acceptable. From the diagram we can conclude the same results, what we have accepted from the boxplot. The digital pen itself was a much more accurate input device than the other two. There is no significant difference between user groups with different computer experience. Although the beginners seem a little bit more accurate we cannot generalize anything from this, since on one hand they were much slower than the intermediates and experts, and on the other hand the sample size was not as large (only 7 beginners were evaluated).

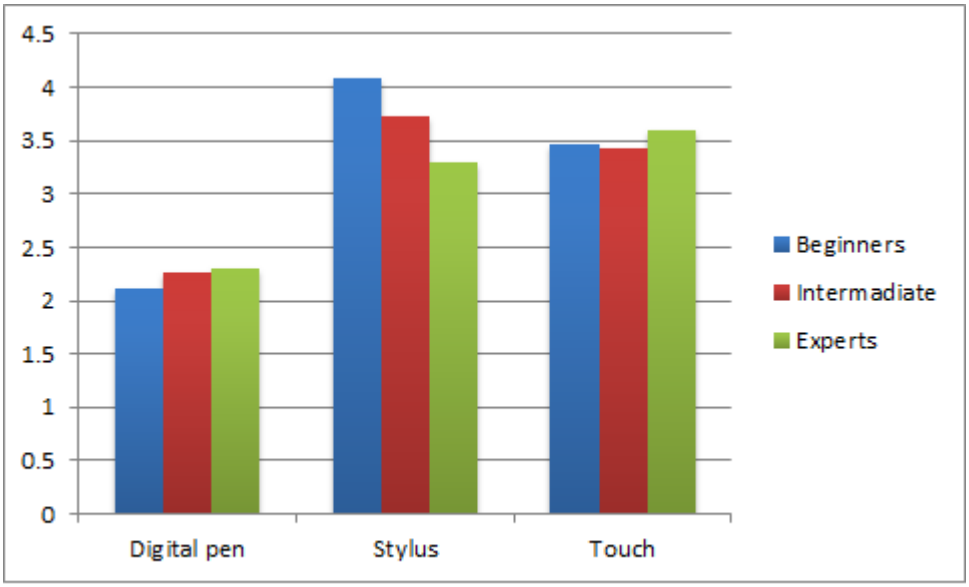

Figure 44: Accuracy of each conditions in terms of the computer experience.

There was no significant difference between stylus and touchscreen inputs in general, but we experienced in general that the beginner participants were using the stylus with difficulty. They didn't know how to keep the stylus properly, and they made more mistakes too.

As we mentioned in the previous measurement, the path level 3 was a good example to measure the accuracy of conditions. Let's see the mean of each conditions (table 2), and relate with the sample values mentioned in the table 1. We can conclude that the accuracy of the digital pen is very good, while touchscreen and stylus were still acceptable, but less accurate.

Just as in case of the time, we also analyzed the data in terms of participants' age and experience with smartphones (see Fig. 45 and Fig. 46). Surprisingly these values were very close to each other, we cannot observe any significant difference. We expected that the younger generation will be

Condition	Average accuracy
Digital pen	2.35
Stylus	4.37
Touchscreen	4.14

Table 2: The average accuracy of the conditions in case of the path level 3 annotation.

more precise than the older one. It seems that the older generation although was slower but, they had the same accuracy as the younger generation.

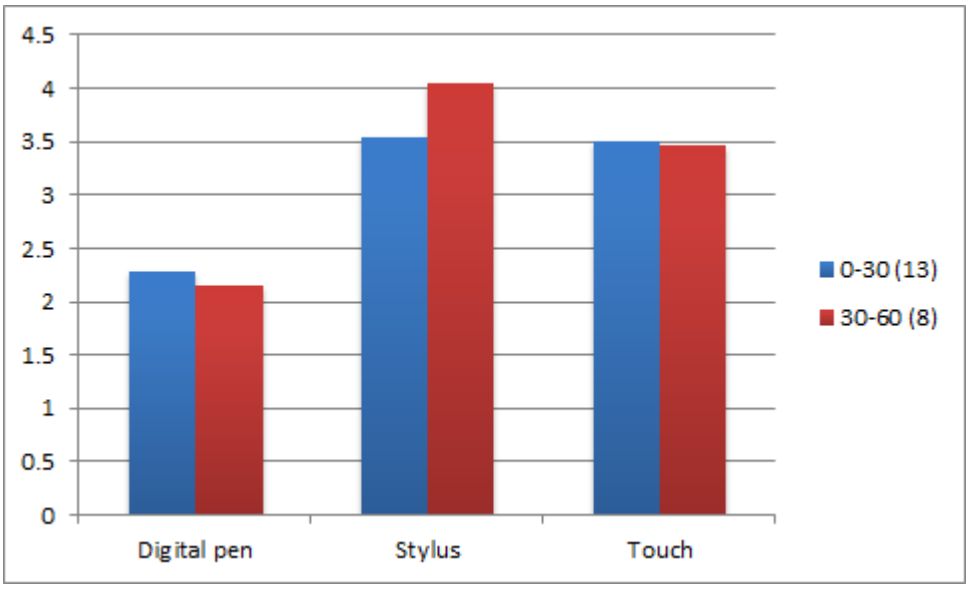

Figure 45: Accuracy of each conditions in terms of the participants age.

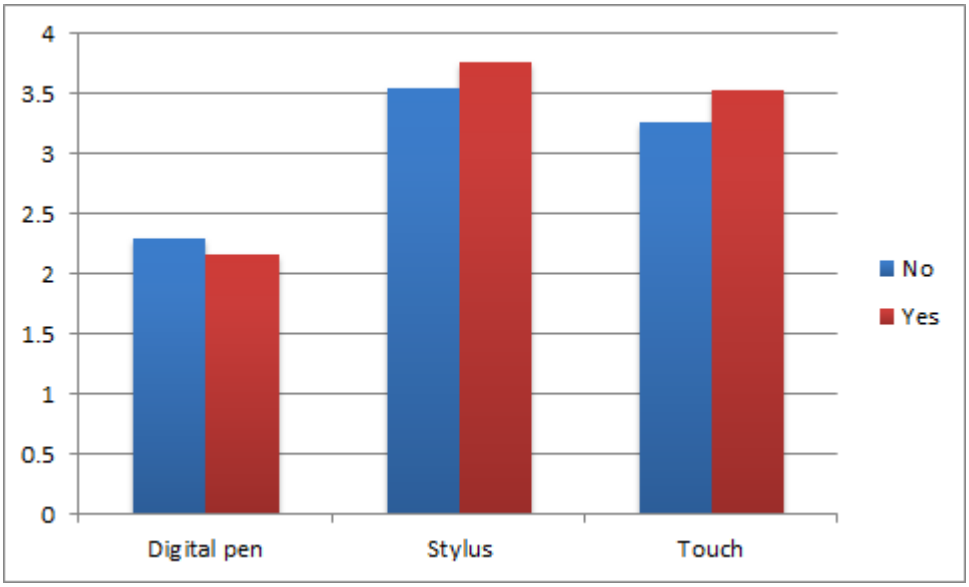

Figure 46: Accuracy of each conditions in terms of the smartphone experience.

6.3.3 Evaluation forms

We also were interested about the experience and opinion of the participants. Let's see the results of these evaluations. We asked the participants to rank the following non-functional requirements from 1-5 : Usability, Responsiveness, Precision, Understandability, Quickness, let's see the result on the Fig. 47. We also asked the users to select that condition which they think it's the most Usable, Responsive etc... Let's see these results too (the Fig. 48). If we consider the first diagram we can conclude that except the responsiveness and understandability which were not significantly different, the digital pen was slightly more popular than the other two conditions. This is clearly reflected in the second diagram, where the participants were allowed to choose only one condition. We also must note here that in the second diagram the differences are more accentuated. We explain this by on one hand with the true performance of the digital pen, and a bias in the usability of the stylus (See section 6.4).

6.3.4 Relation between measured results and evaluation forms

Two of the non-functional requirements, the precision and quickness were both measured (accuracy and time results in the section 6.3.2 and 6.3.1) and asked from the participants through an evaluation form. Let's see if the measured values relates with the evaluations.

We concluded form the measurements that the accuracy of the digital pen was significantly better than the other two conditions' accuracy. Based on the results of the evaluation forms we can conclude the same: 18 participants were selecting digital pen as the most accurate device, in contrary

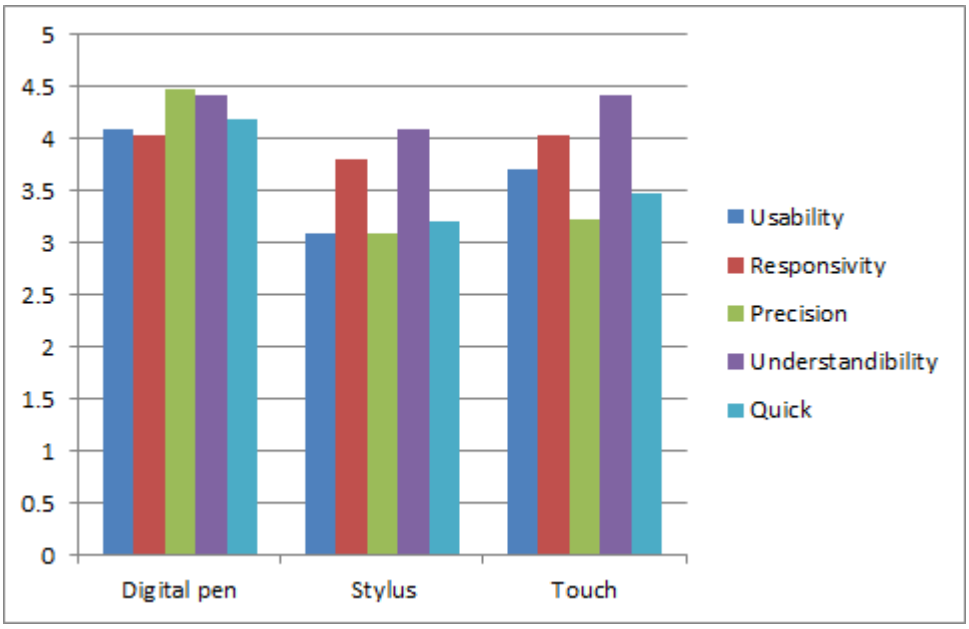

Figure 47: The rate from 1-5 to each non-functional requirements.

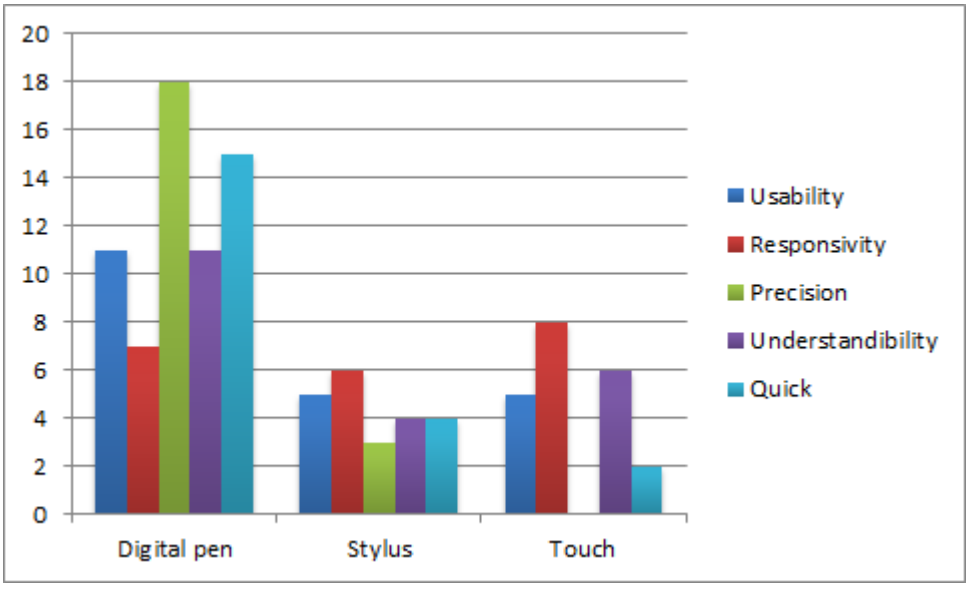

Figure 48: The rate from 1-5 to each non-functional requirements.

with 3 to stylus and nobody voted for touchscreen.

The same observations can be made on quickness of conditions. Again in case of digital pen we have shown that it was significantly quicker than the other two. This is also reflected by the evaluation forms results: 17 participants voted for the digital pen as the quickest input device for stoke data, while only 4 of them voted for stylus and only two for touchscreen.

6.3.5 Mixing paper and digital user interface

During the evaluation we observed an interesting phenomenon. In case of many participants we observed that sometimes they mixed up the paper interface with the digital one. Sometimes they pushed the paper with finger, trying to rate a path, and sometimes they wanted to push the buttons on the touchscreen with the digital pen. Since it was not part of the planned measurement we didn't pay attention to it, just later on we realize the importance of this phenomenon, and another research can be done on how the different environments can be combined together.

6.4 Biases in the measurement

Bias introduced by the system:

Although we put huge efforts to make the system as usable as possible we still had to make some compromises like for example the saving of an annotation using digital pen. Since the annotation first must have been transfered to the phone, and then could be saved we had to decide whether this should be included or not. Since the way of sending data to the phone from digital pen is a clear bias of the system, (maybe in a couple of months a new series of digital pen would came which can stream it's data a with a wireless technology) we decided to not to measure it, and focus the measurement on the annotation task only (we stopped the chronometer when the user finished the drawing).

Another bias what we experienced is that since only one of the participants had previous experience with the digital pen, it was a very new technology what amazed almost everybody, especially the precision of the pen. Our experience was that it influenced their subjective opinion about for example the responsiveness. And although many of them have mentioned that the digital pen seemed highly usable, and they still would prefer using the stylus or touchscreen (simply because there the map is dynamic, even if these conditions are not that precise), they voted with higher mark for the digital pen.

Another bias which also influenced the scores is that, it was not as easy to learn how to use the stylus of the Galaxy Note. None of the have used Galaxy note before, and only 2-3 of them has used stylus before, they didn't knew in what angle they should keep in their hands, how to double click with them, how to avoid clicking with it using a function button (the stylus has another button which enables the users to access some other functionalities of the phone. In this evaluation we didn't use any of them), and finally how hard they should press. This also affected heavily not just the measured values (accuracy, time, nr. of mistakes), but their subjective opinion. We experienced that when the participants had to select the most usable condition, they tended to not to select the stylus.

Learning effect:

So far we have defined the concrete steps of the measurement and what we measured exactly at these elementary tasks. Since the tasks were very easy a strong learning effect was present. We tried to overcome on this effect by randomizing the order of the conditions of which the participants made the evaluation. We also had to make sure that the participants are isolated from each other, so no participants could see the tasks, and learn them by watching other participants. As the coordinators, we also had to be sure that each participants gets the same amount of explanations and time to for training.

Language:

The language was a bias also, since some of the participants was Hungarian, without any English knowledge. We overcome on this bias, by inventing very similar sentences at the writing comment part (”Two train stations” was translated to ”Ket vonat megallo”, and ”This is my favorite park” to ”Ez az en kedvenc parkom”)

6.5 Summary

In this section we have analyzed both measured data and the data from the evaluation forms. As we expected the digital pen was dominating the other two conditions in terms of accuracy and speed, but in contrary with our expectations we found that the stylus was not that accurate. I fact the touchscreen seemed a little bit more accurate, which was reflected both the measured values and the participants' opinion.

We also expected that there would be a significant difference in accuracy of the touchscreen / stylus and digital pen in terms of the participants' experience with smartphones or in age, but it turned out that there is no. This can be explained with the significantly higher time needed to accomplish

the task with the same level of accuracy, so the user has chosen accuracy over time.

Observations made during the evaluation process:

After the evaluation based on the participants feedback some possible improvements can be noted. Fortunatley the system was carefully developed, modularized and the source code was well documented in order to allow further modifications ether by our team, or by a different team.

- Option to delete the last point from the path: the system currently has only one option to delete, but it deletes every point from the map. This option was the most disturbing for most of the participants, since when they made a mistake they had to start everything from the beginning.
- One source of the inaccuracy of the digital pen and the stylus was that different participants held the pen and the stylus in different angles. From the evaluation point of view it would be better if before the start of the evaluation there would be a small calibration, where the system learns the users preference to hold the stylus.
- Now on the smartphone the areas can be selected if the user double-taps on the screen (the path is automatically closed), but it was mandatory that the point where the user makes a double-tap must be in the edge of the area. An improvement would be that let the user to double-tap freely for example inside the area. We also observed that some of the participants were slower at double tap, or the second tap was slightly misplaced. This can be modified too in the future.
- The area selection with digital pen was more obvious for the users than with stylus or touchscreen, but sometimes it happened that they picked up the pen for a short period of time, and the system considered it as two independent strokes were created. The usability of the system would improve if it would consider these strokes as one single continuous stroke.
- The performance of the pen is not that good, here we can think about further optimizations.
- We run the accuracy after the evaluation. Since the algorithm is very quick, the annotations can be check in runtime giving a better feedback to the user.

7 Conclusion and future work

7.1 Project summary

We have seen how we documented ourselves about the state of the art, to find the research area which is not yet explored. We found that although many project has been done with digital pen, and there were a couple of them with map interaction none of the compared and evaluated with stylus and touchscreen inputs. We have decide to make this evaluation on a digital and printed map, allowing the participants to annotate places, paths and areas. They could add comments to it, rate, attach an image, and save them on a server, to allow the searching among them.

After we defined both the functional and non-functional requirements of the system, we could make the system architecture and system design. In meantime we obtained a Livescribe Exho Smartpen, and a Samsung Galaxy Note, and we started the initial tests, and the learning of the Livescribe and Android SDKs. During development we faced with many issues which could be solved partly (for example the data streaming from digital pen to smartphone), but finally in two three iterations we could finish the implementation part. After that we identified those aspects of the project which can be compared with each condition (for example although we implemented the option to take an image it was not part of the evaluation), we prepared the evaluation itself, creating a measurement part and asking the users to fill out an evaluation form too.

We estimated that 21 participants would be enough to evaluate the conditions, so we selected 21 persons taking into consideration their computer experience (7 beginners, 7 intermediates and 7 experts). Finally when we finished the evaluations we introduced everything to a database, and we implemented also an algorithm to calculate the accuracy of each stroke created by the users, preparing the system to analyze the data easily.

7.2 Evaluation results

One part of the results met with our expectations. The digital pen was way more precise and faster than the other two conditions, and the acceptance from the users were very good too. Even the older generation could use it without problems (this is reflected by their accuracy too), even though their average annotation time was much higher.

Stylus input however was a surprise for us. We expected that it will be much more accurate and faster than the touch input, but this was not the case. In the section 6.4 we listed those properties of the stylus which we think that it influenced the users. We also expected that those participants

who had previous experience with smartphones will be more accurate than the others, but again this was not reflected from the measurements.

7.3 Further development

- We mentioned those technical issues what we faced with during the intergeneration of the Livescibe digital pen with the smartphone. Another digital pen which has wireless access to other resources, would eliminate the need of the static USB connection, and would allow the data streaming which would eliminate the additional synchronization step, what we needed to include.
- We see a good potential in the crowdsourcing aspects of the project. We prepared the system to integrate with other crowdsourced system by using inkML standard. We look forward to see improvements here.
- Since we wanted to give the participants a very practical scenario (not just simple annotations on a map), we implemented additional functionalities, which were not evaluated directly, but improved the usability of the system since they found it useful what they would use in their daily life. These functionalities were the possibility to attach an image, to search among the annotations, to attach a timestamp and rate the annotation. These were only a small fraction of those functionalities can be integrated with the system.
- The built in hand writing recognition was very poor in both speed and quality. This also should be improved in the future, maybe by selecting another pen, or using another software.
- The user management can also be improved, allowing the users to limit the search results based on users who made them.

Future improvements in evaluation:

Another improvement can be the evaluation of the system with more participants, maybe involving more complex scenarios with more functionalities, and measuring more attributes, like the number of errors made during a specific tasks.

8 Appendix

A executeCommand() and getTextContent() functions

The **executeCommand()** is a predefined penlet function which is being called on desktop application request. **getTextConten()** is an auxiliary function which reads the content of a file (permanent storage). This file contains the serialized **inkML** annotations.

```
/**
 * Handles the infromation tranfer from pen to an external application
 * Usually a desktop application calls it
 */
public String executeCommand(Command arg0, DataSender arg1) {
  // reset last result
  String result = getTextContent();
  //play click sound
  SoundResource sound =
  context.getResourceBundle().getSoundResource("SL_Next");
  this.player.play(sound);
  return result;
}

/**
 * Reads the data of the file test.txt, and returns it as
 * a String object. This wile contains the inkML representation
 * of the annotations.
 * @return - inkML representation of the annotations
 */
private String getTextContent()
{
  String lastResult = "";
  try {
    PenletStorage store = this.context.getInternalPenletStorage();
    is = new DataInputStream(store.openInputStream("test.txt"));
    lastResult = is.readUTF();
    is.close();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    this.logger.debug("xception" + e.toString());
  }
  return lastResult;
```

```
}
```

Listing 3: The modified recognizer function, of the $1 implementation.

B Code samples which handles penlet events

```
/**
 * Called when a new stroke is created on the pen.
 * We make a distinction between search mode and annotation mode.
 * In search mode only an are is allowed (only one stroke, a cosed path)
 * in annotation mode we allow many strokes.
 *
 * We also have to recognize wether a path or an area was created
 * The stroke information is added to the ICRContext
 */
public void strokeCreated(long time, Region regionId, PageInstance page)

  if (MapUtils.isValidStroke(annotationUtils.temporaryStroke))
  {
    switch (annotationUtils.mode) {
        case MapConstants.ANNOTATION_MODE:
          if (MapUtils.isPath(annotationUtils.temporaryStroke))
          {
            this.label.draw("Path␣added(␣" +
               annotationUtils.temporaryStroke.size() + "␣points)",true);
            this.logger.debug("Path␣stored");
            //play click sound
              SoundResource sound = c
               ontext.getResourceBundle().getSoundResource("SL_CA");
          this.player.play(sound);
          }
          else
          {
            this.label.draw("Area␣added(␣" +
                annotationUtils.temporaryStroke.size() + "␣points)",true)
            this.logger.debug("Area␣stored");
            //play click sound
              SoundResource sound =
                context.getResourceBundle().getSoundResource("SL_CA");
          this.player.play(sound);
          }
```

```
            break;
        case MapConstants.SEARCH_MODE:
          if (MapUtils.isPath(annotationUtils.temporaryStroke))
          {
            this.label.draw("You must select an area",true);
            //play click sound
                SoundResource sound =
                   context.getResourceBundle().getSoundResource("SL_Error")
            this.player.play(sound);
          }
          else
          {
            //play click sound
            SoundResource sound =
                 context.getResourceBundle().getSoundResource("SL_CA");
            this.player.play(sound);
            this.label.draw("Search area recorded( " +
                 annotationUtils.temporaryStroke.size() + " points)",true);
            this.logger.debug("Area stored");
            // remove all the other strokes
            annotationUtils.removeAllPoints();
          }
          break;

        default:
          break;
      }

      // we have most probably an annotation
      // so we will add to the annotation list
      annotationUtils.applyTemporaryStroke();
      reserializeAnnotation("stroke created");
    }
    else
    {
      this.logger.debug("Stroke sent to text recognizer");
      this.icrContext.addStroke(page, time);
    }
    strokeCreatedEventDelegator(time, regionId, page);
  }

  /**
   * When the user pauses (pause time specified by the wizard),
   * all strokes in the ICRContext are cleared
```

```
 */
public void hwrUserPause(long time, String result) {
    this.icrContext.clearStrokes();

    // log the recognized word
    this.logger.debug("Text␣recognized␣on␣Pause:␣" + result);
    // the point can be recognized as a single
    // tap, so we will not take into consideration
    if (!result.equals("i") && !result.equals("I"))
    {
      //play click sound
      SoundResource sound =
           context.getResourceBundle().getSoundResource("SL_CA");
      this.player.play(sound);
      this.logger.debug("");
      this.logger.debug("");
      this.logger.debug("");
      this.logger.debug("Text␣recognized:␣" + result);
      // append teh word to the model
      annotationUtils.appendComment(result);
      //display it on the pen
      this.label.draw(annotationUtils.getComment(),true);
      // reserialize everything
      reserializeAnnotation("User␣pause");
    }
}


/**
 * A single Tap on the map?
 */
public void singleTap(long time, int x, int y) {

  // add the point
  this.logger.debug("Single␣tap:␣X␣" + x + "␣Y:␣" + y);
  //  we are sure that a stroke is not created
  annotationUtils.flushTempraryCoordinateStorage();

  // if the tap is outside of the map area, than we will have to stop it
  if (x < MapConstants.PAPER_MAX_Y)
    return;
  if (y > MapConstants.PAPER_MAX_X)
    return;
  //play click sound
```

```
    SoundResource sound =
         context.getResourceBundle().getSoundResource("SL_Click1");
    this.player.play(sound);

    annotationUtils.addPoint(new Coordinate(x, y));
    this.label.draw("Point␣added␣(␣" +
         annotationUtils.getNrOfPoints() + "␣)");
    reserializeAnnotation("Single␣tap");

  }

  /**
   * A double tap on the map
   */
  public void doubleTap(long time, int x, int y) {
    //  we are sure that a stroke is not created
    annotationUtils.flushTempraryCoordinateStorage();

    // remove the last two points because of teh double tap
    annotationUtils.removeLastStroke();
    annotationUtils.removeLastStroke();

    // display the sumary
    this.logger.debug(annotationUtils.getSummary());
    this.label.draw(annotationUtils.getSummary(), true);

    // we have to specially refresh here,
    // because the user pause comes before the double tap
    reserializeAnnotation("Double␣tap");
  }


  /**
   * Example pen down event handler on the rate 5 star
   * @param time - timestamp when the pen event was generated
   * @param region - region id of the rate 5 ergion
   * @param page - on which page the event occured
   * @return
   */
  protected boolean onRate5PenDown(long time, Region region, PageInstance p
    this.label.draw("Rate:␣5");
    // set the rate
    annotationUtils.setRate(5);
    // reserialize the annotation
```

```
  reserializeAnnotation("rate");
  //play click sound
  SoundResource sound =
       context.getResourceBundle().getSoundResource("SL_IncSpeed");
  this.player.play(sound);
  return true;
}
```

Listing 4: The modified recognizer function, of the $1 implementation.

C ApplicationModel

```
/**
 * Singleton model implementation for  holding the application wide varai
 * @author Becze
 *
 */
public class ApplicationModel {


        /**********************
         *  Attributes
         **********************/
        // user contants
        public int userId;
        public String condition;


        // tabhost (to control the tabs form the mdoel
        public TabHost tabHost;
        public int viewMode = 0;


        // sample paths
        public Annotation samplePaths;

        // display a list of paths, here temporary a list of paths are a
        // this was done only to test the $1 algorithm , and has no use
        public ArrayList<Vector<Point>> helperDisplay;

        // new annotation is what the user creates
        public Annotation newUserAnnotation;
```

```
// based on this annotation searches the user
public Annotation searchByAnnotation;
// the last search results are stored here
public ArrayList<Annotation> searchResults;

// we will save the temprary stroke representation of the annotat
public String tmpXML;

/**********************
 *  Singleton implementation
 **********************/
private static ApplicationModel _instance;


/**
 * singleton class initialization
 * @return
 */
public static ApplicationModel get_instance() {
        if(_instance == null)
                _instance = new ApplicationModel();
        return _instance;
}

/**
 * Initialize the model
 */
public ApplicationModel() {
        // intiialize the annotations
        searchByAnnotation = new Annotation();
        newUserAnnotation = new Annotation();
        samplePaths = new Annotation();
        searchResults =  new ArrayList<Annotation>();
        helperDisplay = new ArrayList<Vector<Point>>();

        // reset the annotation
        searchByAnnotation.reset();
        newUserAnnotation.reset();
}
```

```
}
```

Listing 5: The modified recognizer function, of the $1 implementation.

D Recognizer class

This class is responsible for recognizing and scoring the strokes created by the participants.

```
package be.ac.vub.recognizer.interaction.dollar;

import java.util.*;

import be.ac.vub.model.ApplicationModel;

public class Recognizer
{
  //
  // Recognizer class constants
  //
  ApplicationModel model = ApplicationModel.get_instance();
  int NumTemplates = 16;
  public static int NumPoints = 128;
  public static double SquareSize = 250.0;
  double HalfDiagonal = 0.5 * Math.sqrt(250.0 * 250.0 + 250.0 * 250.0);
  double AngleRange = 45.0;
  double AnglePrecision = 2.0;
  public static double Phi = 0.5 * (-1.0 + Math.sqrt(5.0)); // Golden Rat

  public Point centroid = new Point(0, 0);
  public Rectangle boundingBox = new Rectangle(0, 0, 0, 0);
  int bounds[] = { 0, 0, 0, 0 };

  Vector<Template> Templates = new Vector(NumTemplates);

  public static final int GESTURES_DEFAULT = 1;
  public static final int GESTURES_SIMPLE = 2;
  public static final int GESTURES_CIRCLES = 3;

  public Recognizer()
  {
    this(GESTURES_DEFAULT);
  }
```

```
public Recognizer(int gestureSet)
{
  switch(gestureSet)
  {
    case GESTURES_DEFAULT:
      loadTemplatesDefault(); break;

  }
}

void loadTemplatesDefault()
{
  Templates.addElement(loadTemplate("Path␣Level␣1", TemplateData.p_l1,
  Templates.addElement(loadTemplate("Path␣Level␣2", TemplateData.p_l2,
  Templates.addElement(loadTemplate("Path␣Level␣3", TemplateData.p_l3,
  Templates.addElement(loadTemplate("Path␣Level␣1␣r", TemplateData.copy
  Templates.addElement(loadTemplate("Path␣Level␣2␣r", TemplateData.copy
  Templates.addElement(loadTemplate("Path␣Level␣3␣r", TemplateData.copy
  Templates.addElement(loadTemplate("Area␣Level␣1", TemplateData.a_l1,
  Templates.addElement(loadTemplate("Area␣Level␣2", TemplateData.a_l2,
  Templates.addElement(loadTemplate("Area␣Level␣3", TemplateData.a_l3,
  Templates.addElement(loadTemplate("Area␣Level␣1␣r", TemplateData.copy
  Templates.addElement(loadTemplate("Area␣Level␣2␣r", TemplateData.copy
  Templates.addElement(loadTemplate("Area␣Level␣3␣r", TemplateData.copy
}


Template loadTemplate(String name, int[] array, boolean display)
{
  return new Template(name, loadArray(array, display));
}

Vector loadArray(int[] array, boolean display)
{
  Vector v = new Vector(array.length/2);
  for (int i = 0; i < array.length; i+= 2)
  {
    Point p = new Point(array[i], array[i+1]);
    v.addElement(p);
  }

//  System.out.println(v.size() + " " + array.length);
  if(display)
    model.helperDisplay.add(v);
```

```
    return Utils.Resample(v, NumPoints);
  }



public Result Recognize(Vector points)
{
  // Forcing the number of point in the sample to be constant
  points = Utils.Resample(points, NumPoints);

  // display the sample
  model.helperDisplay.add(points);

  // the template id of the best match
  int templateId = 0;
  // distance between the template and the sample
  double distance = Double.MAX_VALUE;
  for (int i = 0; i < Templates.size(); i++)
  {
    Template template = ((Template)Templates.elementAt(i));
    double d1 = Utils.rotatedPathDistance(points, template.Points);
    double d2 = Utils.rotatedPathDistance(template.Points, points);
    // get the min of the two distances
    double d = (d1 < d2)? d1:d2;
    if (d < distance)
    {
      distance = d;
      templateId = i;
    }
  }
  // we just devide it with 100 to fit into the interval [0-10]
  double score = distance / 100;

  //display the template
  model.helperDisplay.add(((Template)Templates.elementAt(templateId)).Poi

  return new Result(((Template)Templates.elementAt(templateId)).Name, sco
};

  int AddTemplate(String name, Vector points)
  {
    Templates.addElement(new Template(name, points));
    return Templates.size();
```

```
  }

  int DeleteUserTemplates()
  {
    for (int i = Templates.size()-NumTemplates; i > 0; i--)
    {
      Templates.removeElementAt(Templates.size()-1);
    }

    return Templates.size();
  }

}
```

Listing 6: The modified recognizer function, of the $1 implementation.

E PHP scripts

synchronizePenlet.php:

```
<?php

  // we will open the synchronization file, and based in it's content we
  // will return it
  $file=fopen("penletData.txt","w+");

  // we will initialize the file status
  fwrite($file, "Status: dataRequested");
  fclose($file);
  /*
   *  Now we will have to wait until the data is written into the file, s
   *  we will loop for 5 seconds in each 100ms we read the content of the
   *  if the status has changed to ready than we will return it to the cl
   */
  $data = "noData";
  // check the file content
  for ($i = 0; $i < 50; $i++)
  {
    usleep(100000);
    // we will initialize the file status
    $statusLine = file_get_contents("penletData.txt");
```

```
    if(substr_count($statusLine, "</ink>") != 0)
    {
      $data = $statusLine;
      break;
    }
  }

  print(json_encode($data));


?>
```

Listing 7: The modified recognizer function, of the $1 implementation.

saveannotationHandler.php:

```
<?php
/*
 * We will save the annotation to the database
 */
mysql_connect("127.0.0.1","root","");
mysql_select_db("trip_manager");
$query = 'INSERT INTO annotations (points, comment, rate, user_id, `condi
                                   "'.$_POST['comment'].'",
                                   '.$_POST['rating'].',
                                   '.$_POST['user_id'].',
                                   "'.$_POST['condition'].'"
                                    )';
$sql=mysql_query($query);

print $query;
mysql_close();
```

Listing 8: The modified recognizer function, of the $1 implementation.

getAnnotation.php:

```
<?php
  mysql_connect("127.0.0.1","root","");
  mysql_select_db("trip_manager");

  try{
```

```
        $query = 'SELECT *
              FROM annotations
              WHERE id ='. $_GET['id'];
    }
    catch (Exception $e)
    {
        $query = 'SELECT *
              FROM annotations';
    }

    $sql=mysql_query($query);

    while($row=mysql_fetch_assoc($sql))
      $output[]=$row;

    print(json_encode($output));

    mysql_close();
?>
```

Listing 9: The modified recognizer function, of the $1 implementation.

deleteAnnotation.php:

```
<?php
/*
 * We will save the annotation to the database
 */
mysql_connect("127.0.0.1","root","");
mysql_select_db("trip_manager");
$query = 'UPDATE annotations SET comment="invalidated",accuracy=9999 WHER
$sql=mysql_query($query);

print $query;
mysql_close();
```

Listing 10: The modified recognizer function, of the $1 implementation.

saveAnnotationAccuracy.php:

```
<?php
/*
```

```
 * We will save the annotation to the database
 */
mysql_connect("127.0.0.1","root","");
mysql_select_db("trip_manager");
$query = 'UPDATE annotations SET shape="'. $_POST['name'] .'",accuracy='.
$sql=mysql_query($query);

print $query;
mysql_close();
```

Listing 11: The modified recognizer function, of the $1 implementation.

F Evaluation form

form.pdf

Evaluation proposal

Szabolcs Becze

June 7, 2012

This is a summative(after development), cooperative evaluation to measure the usability of the digital pen and paper, stylus and touch pad input methods. Each participant will get a unique identifier and each annotation will be saved into the database, so we can evaluate them later on.

1 Introduction

Purpose: The introduction of both the paper and smart-phone interfaces to the participant. To show them the basic functionalities of the digital pen and show them a simple demonstration of the data transfer between the pen and smart-phone. It will also be highlighted that the purpose of this evalutation is to measure the usability of the paper and smart-phone interfaces and to compare the different input methods, and not to evaluate their performance.
Description: Three simple scenarios will be shown: A simple annotation a path, an area and simple search scenario.
Duration: 3-4 minutes.

2 First part of the evaluation

Purpose: To evaluate the UI and train the participants, by giving them simple tasks. We are noting whether the task has been accomplished or not, and we are also counting the number of errors they made.
Description: The following three scenarios are considered:

1. **simple scenario:**
   - Create a path between two train stations (Station Brussel-Schuman and Station Brussel-Luxemburg) in the city center.
   - Attach the following comment: *"Path between two train stations."*
   - Save the annotation.

form.pdf

2. **Complex scenario:**
   - Annotate the area of the park de Brussels(Warandepark).
   - Attach the following comment: *"This is the metro station Merode"*.
   - Rate it with 5 stars.
   - Delete the previous comment and attach a new one: *"This is a park"*.
   - Save the annotation.
3. **Search scenario:**
   - Select the search mode.
   - Select an area which contains the previous two annotations.
   - Filter the search such that the search results must contain the word *"park"*.
   - Submit the search.

Duration: Each participant will do each scenarios from above with all the three different conditions in a randomized order(digital pen and paper, stylus and touch-pad). The following two tables are filled:

Table 1: Was the scenario accomplished? (Yes-No)

Scenario	Digital pen and paper	Stylus	Touch-pad
Simple scenario			
Complex scenario			
Search scenario			

Table 2: How many errors made the participant?

Scenario	Digital pen and paper	Stylus	Touch-pad
Simple scenario			
Complex scenario			
Search scenario			

form.pdf

3 Second part of the evaluation

Here we would like find out how efficient are the different conditions at map interactions. We will ask each participant to perform scenarios with increasing difficulty using each of the input methods, and we will measure the time needed to perform the task, the number of mistakes what they made during the task, and using a heuristic function we will calculate a relative accuracy of the drawn shapes.

Tasks		Digital pen and paper			Stylus			Touch-pad		
		Time	Mistakes	Accuracy	Time	Mistakes	Accuracy	Time	Mistakes	Accuracy
Marking a place on the map	Level 1									
	Level 2									
	Level 3									
Marking an area on the map	Level 1									
	Level 2									
	Level 3									
Writing a comment on the map										

Table 3: We are measuring the time needed to perform the task, the nr. of mistakes, and the accuracy of the annotation.

G Evaluation questionnaire

questionnaire.pdf

Application evaluation form

Participant ID: _____

With this questionnaire we would like to get feedback about the usability of the application with each of the three different input methods.

About you

1. **Your age**: _____
2. **Your gender:** ☐ Male ☐ Female
3. **I am:** ☐ Left handed ☐ Right handed
4. **My computer experience level is:**
   - ☐ Beginner
   - ☐ Intermediate
   - ☐ Expert
5. **I have experience with smart-phones (ex. touch, multi-touch):** ☐ Yes ☐ No
6. **I have experience with digital pens:** ☐ Yes ☐ No

About the application

7. **How would you rate the application from 1-5 (1 - Strongly disagree, 5 - strongly agree).**

Digital pen and paper

	Strongly disagree				**Strongly agree**
1. **I found the system usable:**	1	2	3	4	5
2. **I found the system responsive:**	1	2	3	4	5
3. **I found the system precise:**	1	2	3	4	5
4. **I quickly understood the system:**	1	2	3	4	5
5. **I found the system quick:**	1	2	3	4	5

Stylus

	Strongly disagree				**Strongly agree**
1. **I found the system usable:**	1	2	3	4	5
2. **I found the system responsive:**	1	2	3	4	5
3. **I found the system precise:**	1	2	3	4	5
4. **I quickly understood the system:**	1	2	3	4	5
5. **I found the system quick:**	1	2	3	4	5

Touch-pad

	Strongly disagree				**Strongly agree**
1. **I found the system usable:**	1	2	3	4	5
2. **I found the system responsive:**	1	2	3	4	5
3. **I found the system precise:**	1	2	3	4	5

uestionnaire.pdf

4. **I quickly understood the system:**

1	2	3	4	5

5. **I found the system quick:**

1	2	3	4	5

8. Which system was the most easy to use?

- ☐ Digital pen and paper
- ☐ Stylus
- ☐ Touch-pad

9. Which system was the most responsive?

- ☐ Digital pen and paper
- ☐ Stylus
- ☐ Touch-pad

10. Which system was the most precise?

- ☐ Digital pen and paper
- ☐ Stylus
- ☐ Touch-pad

11. Which system was the easiest to understood?

- ☐ Digital pen and paper
- ☐ Stylus
- ☐ Touch-pad

12. Which system was the quickest?

- ☐ Digital pen and paper
- ☐ Stylus
- ☐ Touch-pad

12b. Write here you observations and suggestions:

uestionnaire.pdf

References

[1] Ink markup language (inkml), w3c recommendation 20 september 2011. http://www.w3.org/TR/InkML/, 2011.

[2] $1unistroke recognizer in javascript. http://depts.washington.edu/aimgroup/proj/dollar/, 2012.

[3] Anoto. http://en.wikipedia.org/wiki/Anoto, May 2012.

[4] Anoto official website. http://www.anoto.com, May 2012.

[5] Echo smart pen. http://www.livescribe.com/en-us/smartpen/echo/, May 2012.

[6] Livescribe official website. http://www.livescribe.com/en-us/, May 2012.

[7] Non-functional requirement. http://en.wikipedia.org/wiki/Non-functional_requirement, 2012.

[8] Point in polygon. http://en.wikipedia.org/wiki/Point_in_polygon, 2012.

[9] Touchscreen. http://en.wikipedia.org/wiki/Touchscreen, 2012.

[10] Tripadvisor's official website. http://www.tripadvisor.nl/, 2012.

[11] Joerg Arzt-Mergemeier, Willi Beiss, and Thomas Steffens. The digital voting pen at the hamburg elections 2008: electronic voting closest to conventional voting. In *Proceedings of the 1st international conference on E-voting and identity*, VOTE-ID'07, pages 88–98, Berlin, Heidelberg, 2007. Springer-Verlag.

[12] Naser Avesta. Performance evaluation of digital pen for capturing data in land information systems. 2011.

[13] Michael Grossniklaus Rudi Belotti Corsin Decurtins Beat Signer, Moira C. Norrie and Nadir Weibel. Paperbased mobile access to databases. 2005.

[14] Chih-Kai Chang. Augmenting wiki system for collaborative efl reading by digital pen annotations. In *Proceedings of the 2009 International Symposium on Ubiquitous Virtual Reality*, ISUVR '09, pages 43–46, Washington, DC, USA, 2009. IEEE Computer Society.

[15] Chia-Hao Chuang, Po-Yao Chao, Hsiao-Kuang Wu, and Gwo-Dong Chen. Integrated textbook: Augmenting paper textbooks with digital learning support using digital pens. In *Proceedings of the Sixth IEEE*

International Conference on Advanced Learning Technologies, ICALT '06, pages 613–617, Washington, DC, USA, 2006. IEEE Computer Society.

[16] Raimund Dachselt, Mathias Frisch, and Eike Decker. Enhancing uml sketch tools with digital pens and paper. In *Proceedings of the 4th ACM symposium on Software visualization*, SoftVis '08, pages 203–204, New York, NY, USA, 2008. ACM.

[17] Florian Geyer, Daniel Klinkhammer, and Harald Reiterer. Supporting creativity workshops with interactive tabletops and digital pen and paper. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 261–262, New York, NY, USA, 2010. ACM.

[18] François Guimbretière. Paper augmented digital documents. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, pages 51–60, New York, NY, USA, 2003. ACM.

[19] Jan Hess, Lin Wan, Volkmar Pipek, and Guy Kuestermann. Using paper and pen to control home-it: lessons learned by hands-on experience. In *Proceddings of the 9th international interactive conference on Interactive television*, EuroITV '11, pages 203–212, New York, NY, USA, 2011. ACM.

[20] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. Pen + touch = new tools. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 27–36, New York, NY, USA, 2010. ACM.

[21] Motoki Miura, Susumu Kunifuji, and Yasuyuki Sakamoto2. Practical environment for realizing augmented classroom with wireless digital pens. In *Proceedings of the 11th international conference, KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems: Part III*, KES'07/WIRN'07, pages 777–785, Berlin, Heidelberg, 2007. Springer-Verlag.

[22] Moira C. Norrie, Beat Signer, and Nadir Weibel. Print-n-link: weaving the paper web. In *Proceedings of the 2006 ACM symposium on Document engineering*, DocEng '06, pages 34–43, New York, NY, USA, 2006. ACM.

[23] Hendro Prastowo. "digital pen mobile mapping", mobile mapping using digital pen & paper technology meets tradition. 2006.

[24] Beat Signer. Interactive paper: Past, present and future. 2008.

[25] Beat Signer, Michael Grossniklaus, and Moira C. Norrie. Interactive paper as a mobile client for a multi-channel web information system. *World Wide Web*, 10(4):529–556, December 2007.

[26] Jürgen Steimle. Designing pen-and-paper user interfaces for interaction with documents. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, TEI '09, pages 197–204, New York, NY, USA, 2009. ACM.

[27] Yu Suzuki, Kazuo Misue, and Jiro Tanaka. Stylus enhancement to enrich interaction with computers. In *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, HCI'07, pages 133–142, Berlin, Heidelberg, 2007. Springer-Verlag.

[28] Brecht Van Laethem. Paper-digital maps. 2011.

[29] Sarah Van Wart, K. Joyce Tsai, and Tapan Parikh. Local ground: a paper-based toolkit for documenting local geo-spatial knowledge. In *Proceedings of the First ACM Symposium on Computing for Development*, ACM DEV '10, pages 11:1–11:10, New York, NY, USA, 2010. ACM.

[30] Brett Wilkinson and Paul Calder. Investigating touch interactions for an augmented world. In *Proceedings of the 20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat*, OZCHI '08, pages 25–32, New York, NY, USA, 2008. ACM.

[31] Ron B. Yeh, Andreas Paepcke, and Scott R. Klemmer. Iterative design and evaluation of an event architecture for pen-and-paper interfaces. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 111–120, New York, NY, USA, 2008. ACM.