Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Conceptual Modelling with Smart Paper

Master thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Van Denhouwe Tania

Promoter: Prof. Dr. Olga De Troyer
Advisor: Dieter Van Thienen

Academic year 20014-2015

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Conceptueel modelleren met behulp van intelligent papier

Masterproef ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Van Denhouwe Tania

Promotor: Prof. Dr. Olga De Troyer
Begeleider: Dieter Van Thienen

Academiejaar 2014-2015

Abstract

Conceptual modeling is an important part of the development process. It is used for modeling the database, the interaction in user interfaces and the different objects involved. The last decade more and more tools were developed to assist with the modeling process, going from desktop to cloud applications. However, even if there are plenty of digital tools some modelers still prefer to model on paper, as it is faster and more flexible. In this thesis we looked for an appropriate bridge between paper-based modeling and digital modeling tools. We opted for an approach where we used existing soft and hardware, i.e. a digital (smart)pen, interactive paper, a handwriting recognition engine (MyScript) and an existing drawing tool (draw.io), and integrated them through the Draw2Model application. We also have tested how feasible this approach is. The Draw2Model application runs on a C# server. The data from the model created on the interactive paper is collected by a penlet of the smartpen and retrieved by the server application, which will send the data to the MyScript recognition engine. This engine will send the recognition results back to the sever, which will then analyze and convert this information into an XML format that fits the draw.io XML format.

Keywords: Handwriting recognition, Conceptual Modeling, MyScript, Echo Smartpen, ORM, Draw2Model, Digital-paper interaction

Samenvatting

Conceptueel modelleren is een belangrijk onderdeel van het software ontwikkelingsproces. Het wordt gebruikt voor het modelleren van de gegevensbanken, het modelleren van de interactie in gebruikersinterfaces en de verschillende betrokken objecten. In de afgelopen tien jaar werden steeds meer tools ontwikkeld om te helpen bij het modelleer proces gaande van desktopgebaseerde tools tot Cloud-applicaties. Ondanks de vele digitale tools die ter beschikking staan van de modelbouwer zal deze nog steeds gemakkelijk naar pen en papier grijpen om te modelleren gezien dit vaak sneller en flexibeler is. In deze thesis gaan we op zoek naar een geschikte methode om een brug te vormen tussen papier-gebaseerd modelleren en de digitale tools. Daarnaast onderzoeken we hoe haalbaar het is om zulke bruggen te creëren. De uiteindelijke aanpak maakt gebruik van bestaande software en hardware, nl. een digitale pen, interactief papier, een handschrift erkenning motor en een bestaande teken tool, zijnde draw.io en verbinden deze door middel van de Draw2Model applicatie. Deze applicatie draait op een C# server. De data van het getekend model zal door middel van een penlet voor de digitale pen opgeslagen worden en kan door de server applicatie opgevraagd worden en naar de recognition engine van MyScript gestuurd worden die de resultaten van de herkenning terug naar de server applicatie zal sturen. Deze gegevens worden dan verwerkt en in een XML formaat geschikt voor draw.io omgezet.

Trefwoorden: Handschrift herkenning, Conceptueel modelleren, MyScript, Echo Smartpen, ORM, Draw2Model, Digital-paper interactie

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Acknowledgments

This thesis was supported by the WISE research group of the Vrije Universiteit Brussel, which provided me with the necessary tools. I thank Prof. dr. O. De Troyer and D. Van Thienen who provided me insight and expertise that greatly assisted the research.

I thank my family and friends for their moral support and willingness to serve as test subjects, and my boyfriend F. Van Ham for his assistance with setting up the C# server and comments that greatly improved the thesis.

And finally I would like to show my gratitude to my grandmother M. Hulet, to have taught me so much and gave me the opportunity to pursue higher education.

Contents

List of Figures

List of Tables

List of Abbreviations

AFD	Anoto Functionality Document
API	Application Programming Interface
ATK	Application Toolkit
CDK	Cloud Development Kit
CSDP	Conceptual Schema Design Procedure
ER	Entity Relationship
HTTP	Hypertext Transfer Protocol
InkML	Ink Markup Language
JSON	JavaScript Object Notation
ORM	Object Role Modelling
OS	Operating System
SDK	Software Development Kit
UML	Unified Modeling Language
XML	Extensible Markup Language
REST	Representational State Transfer

1 Introduction

1.1 Context

Conceptual modeling is an important part of a development process (Figure 1.1). It is used during the software development process as well as for modeling databases and for many other applications. One can describe a conceptual model best as the blueprint of a domain or concept that should ensure that everyone has the same idea of how the domain/concept should be interpreted, what the different aspects/parts are and how they relate to each other.

1.2 Problem

Nowadays there are many conceptual modeling languages and methods that enable us to bring concepts to paper; each of them has its own specific advantages and disadvantages depending on the purpose for which they are used.

Conceptual modeling is often a collaborative effort. During such a process, one easily grasps for a pen and paper to quickly explain to others the notion of the concept by means of a sketch, to which others can add suggestions or improvements. These sketches are often sloppy and are later on

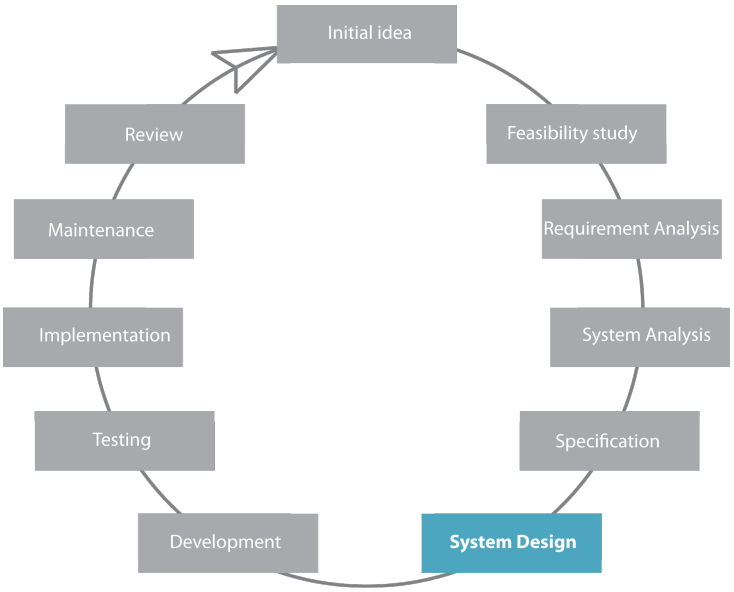


Figure 1.1: The development cycle

manually entered into a modeling tool for further processing. This creates a kind of duplication of effort and can be rather time consuming depending on the ease of use of the tool and the experience of the modeler with the tool.

1.3 Justification

In the last three decades the paperless office is a buzzword in many companies and institutions. Archives are being digitized, forms filled out via computers or tablets and correspondence and billing is done via e-mail. However, even with the strong rise of digitization, paper is still omnipresent in the office (Sellen & Harper, 2002).

This is due to the affordances of paper versus digitized documents. Paper is thin, light, porous, and flexible, making it permanent, portable, and contextualized. Digital documentation on the other hand is dynamic, immaterial, and agentive but depends on its carrier. Even if digital devices are becoming thinner, more flexible and require less battery power they still don't have the affordances of paper. As paper does not seem to be easily abandoned from the office environment, a bridge between the digital and the physical world seems to be the ideal solution in order to have both the affordances of paper and as the advantages of digital documentation (Dymetman

& Copperman, 1998).

Intelligent paper is a proposal for such a bridge. Intelligent paper consists of three elements: a support; being the sheet of paper that can uniquely be identified; an input device called a pointer that can record the coordinates on the paper, and a communication infrastructure, which sends the page id and pointer location pairs to a recognition program.

1.4 Objectives & Research Questions

The main goal of this thesis is to support a modeler in creating conceptual models with pen and paper while avoiding the additional effort of entering them manually into a modeling tool. Therefore, we propose to use intelligent paper for sketching the model followed by transforming this conceptual model in digital ink into a true digital model, recognizable by a modeling tool. Because of the wide array of conceptual modeling languages and the limited time available for a Master thesis, this thesis will focus on just one: the Object Role Modelling (ORM) language. This language was chosen because it is used a lot in the WISE research lab of our university for research purpose and teaching.

To achieve this goal, we have to overcome different obstacles. We first question which digital pen is most suitable. Once a pen has been chosen, a way to extract the writing/drawing and apply shape and text recognition has to be found. After the shapes and text have been extracted they have to be grouped into the objects of the ORM language, -so a way to group objects has to be developed. Finally, a format to export the components of the model has to be decided on and this format has to be converted into a format suitable for a modeling tool.

During this process, we ask ourselves the following questions:

- What are the differences between the available smart pens and what are their limitations?
- What smart pen is the best choice for this application?
- How are the strokes recorded and what is the most suitable digital ink format?
- How can the digital ink be extracted on different Operating System (OS)?

- What software is available for text and shape recognition?
- Should text and shape recognition happen separately or at the same time?
- How well does the recognition software work?
- What are the limitations of the recognition software?
- How can we group the recognized elements and translate them to the elements of the ORM language?
- Do we need to limit or adapt the current graphical representation of the ORM language in order to enable the grouping of elements? If so, how?
- Which is a suitable export format?
- Which tools are available to visualize the model?

1.5 Research Method

We used a research methodology based on Design Science (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007) to achieve the research goal. An overview of the methodology is given in Figure 1.2. We shortly explain the different steps taken.

1.5.1 Awareness of the Problem

In order to create a bridge between paper and digital world, the option of a digital pen for conceptual modeling is explored. A literature study has been performed about similar projects as well as about the software and hardware relevant to the thesis.

1.5.2 Solution

The Echo smartpen of Livescribe will be used in combination with the handwriting recognition software of MyScript to create an ORM model on paper and translate it to an ORM model for the modeling tool Draw.io. Along the way, problems and their solutions are explored. This should create an insight into the feasibility of such a bridge and how to achieve this with other modeling languages.

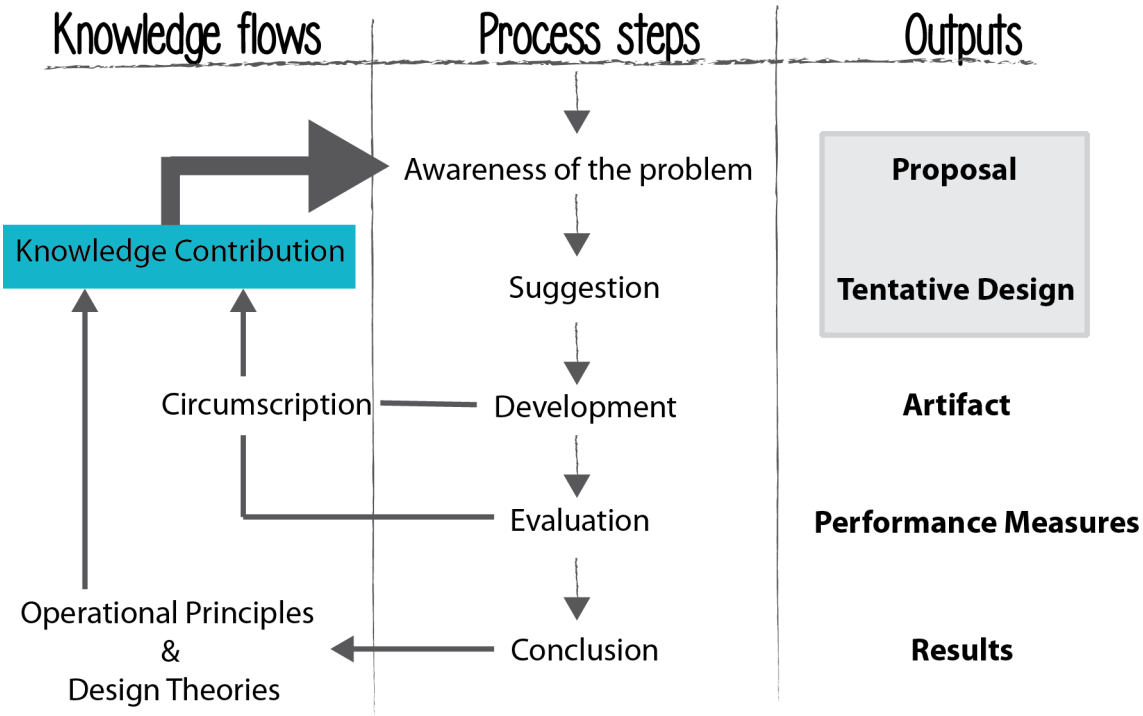


Figure 1.2: DSR process model

1.5.3 Development

The development process has been divided in the following steps:

1. The drawing of different ORM elements with the smart pen.
2. The extraction of the digital ink.
3. The conversion of the digital ink into a format suitable for recognition.
4. The grouping of elements into ORM objects.
5. Translating the objects into a suitable Extensible Markup Language (XML) format.
6. Compare the original drawing with the result.

This process is repeated in an iterative way in order to improve the result (see Figure 2.2).

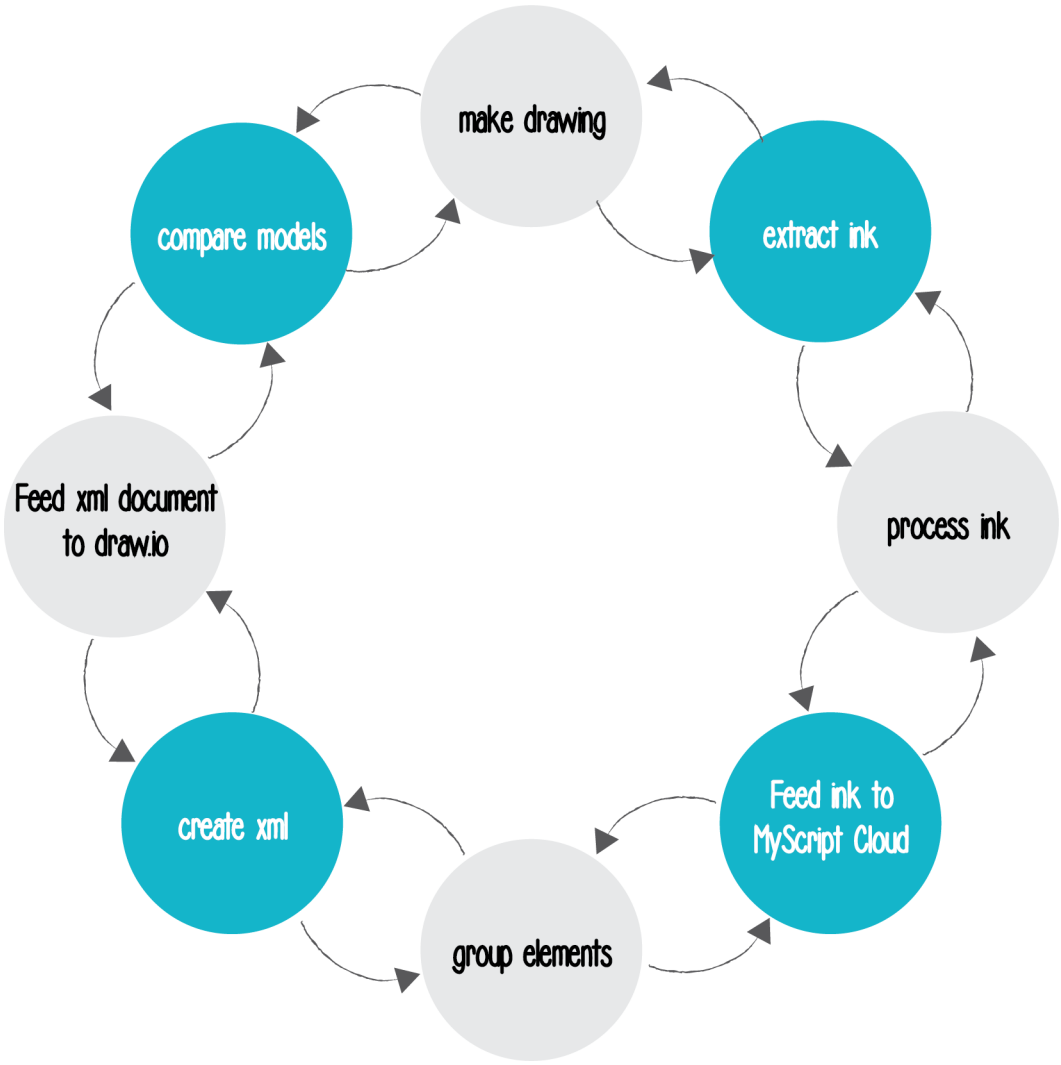


Figure 1.3: The different stages of the development process

1.5.4 Evaluation

The development phase is followed by an evaluation to investigate how the obtained models deviate from the paper models and to elicit the option of end-users.

1.5.5 Conclusion

Finally, a conclusion regarding the proposed solution is formulated based on the findings obtained in the evaluation phase, a global overview of the

accomplishments, limitations, and possible solutions are formulated. From this conclusion some possibilities for future work are formulated.

1.6 Structure of the Thesis

The thesis is structured as follows. In Chapter 2, related work is discussed to show what has already been done in the context of the topics of the thesis. Chapter 3 provides the background about the different components involved for creating a solution: ORM, draw.io, digital pen, Anoto paper, digital ink, XML, and existing conceptual modeling tools.

In Chapter 4 the different technologies which are used in the development process are discussed. This chapter is followed by Chapter 5 which describes the development process itself.

The last chapters, Chapter 6 and Chapter 7, provide the evaluation of the solution, a reflection on the results, soft- and hardware used, and the process used and possible improvements.

2 Related work

This chapter discusses previous work that is relevant to this thesis. We First provide an overview of some applications of intelligent paper. This is followed by a brief overview about existing work for enhancing UML diagramming with digital pens and paper, sketch-based diagramming applications, paper-based digital interactions and go into some detail about Tahuti, which is a sketch recognition system for UML Diagrams, and the LADDER sketching language. Finally, we discuss Smart Study, since it uses the Echo smartpen for paper-digital interaction in an educational context and 53's Paper, which simulate paper interaction for creating diagrams and visualizations.

2.1 Applications of Intelligent Paper

As intelligent paper does not only enable the capture of digital handwriting, but can also be used as an interactive medium, some of its applications are described in this chapter in order to provide an overview of the possibilities and added values of using intelligent paper.

One of the applications is PaperPoint (Signer & Norrie, 2007): a paper based presentation tool. This tool doesn't only enable to use paper as a medium to navigate trough slides but also enables users to interact with the content of the slides in real time and encourages collaborative presentations

or brainstorming.

A second application is the use of a smartpen and interactive paper as an interactive guide during the EdFest international arts festival (Signer et al., 2006) illustrated in Figure 2.1. This application demonstrates how intelligent paper can provide users with information or can assist them, when combined with a database.

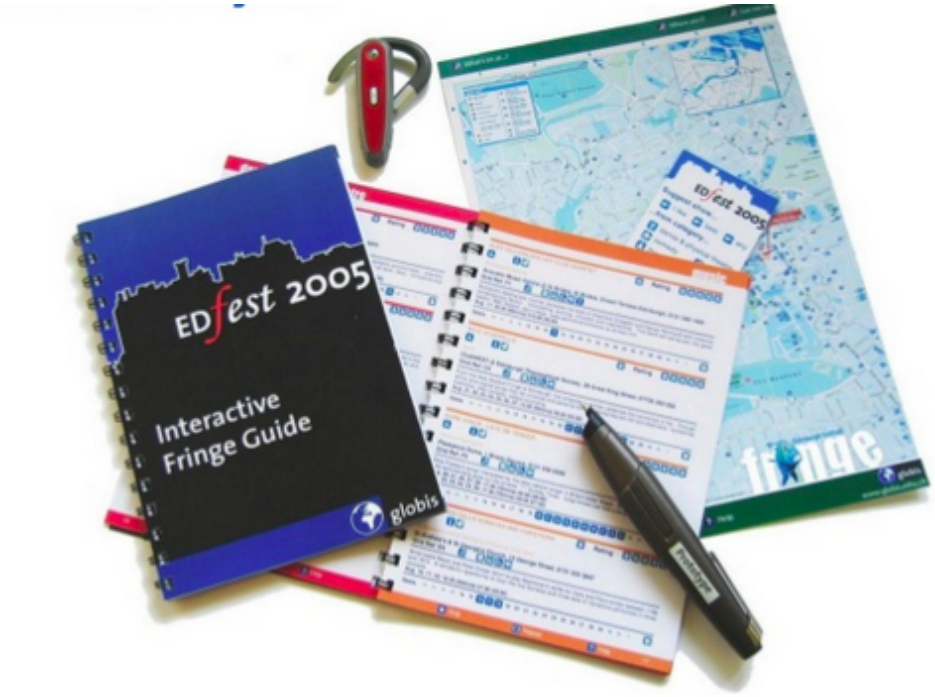


Figure 2.1: The EdFest Interactive Guide

Source: (Signer, 2013)

Another application is for the medical sector, where handwriting needs to be digitized. As nowadays doctors and other medical personnel have to deal with a lot of paperwork they often spend more time on the paperwork than on the patients themselves. Having to first write everything down when seeing a patient and then having to input it manually into the computer is very time consuming (Yen & Gorman, 2005). Therefore some hospitals have equipped their medical staff with intelligent paper (REPORTER, 2010). This and data acquisition (Rose & Bezjak, 2009) are only some of the challenges

that hospitals have to deal with that could be supported by using intelligent paper.

The last application we will discuss is a learning application. Smart study (Van Thienen, 2014a) and the Lernstift/vibewrite (Lee Garber, 2013) demonstrate the use of intelligent paper in learning by correcting their mistakes and providing learners with feedback.

2.2 Enhancing UML Sketch Tools with Digital Pens and Paper

Over the last two decades Sketch-based diagram applications have been developed to support the software development process. Examples of these are the Tahuti (Hammond & Davis, 2006), which uses multi-layered recognition to recognize the hand drawn shapes of an UML diagram, and the E-whiteboard tool of (Chen, Grundy, & Hosking, 2003), which encourages collaborative sketching. However these tools still lacked the flexibility of paper.

There have also been some developments on systems that integrate paper based and digital interaction by creating areas on the pages that have interactions assigned to them. These interactions can be invoked by tapping on the designated area (Yeh, Klemmer, & Paepcke, 2007) and (Brandl, Haller, Oberngruber, & Schafleitner, 2008).

(Dachselt, Frisch, & Decker, 2008) try to combine both a tool to support Conceptual diagramming and paper based-digital interaction. They propose a tool where the different shapes of an UML diagram can be selected on a paper page (= UML palettes) to make the recognition easier.

Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams

This paper explains a multi-layered (shape) recognition method. The method consist out off four stages which are depicted in Figure 2.2.

1. Pre-processing: corners are detected by means of stroke timing data (a user slows down when reaching a corner)

2. Selection: strokes are grouped to form a collection

3. Recognition: the groups are matched with the possible shapes

4. Identification: the most likely match is selected

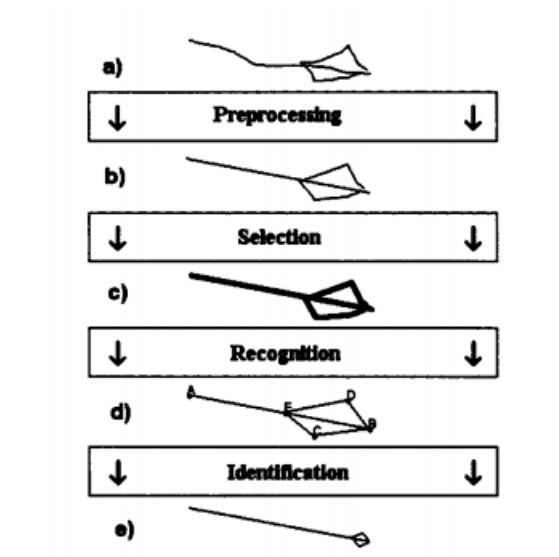


Figure 2.2: The Stages of Recognition

Source: (Hammond & Davis, 2006)

2.3 LADDER, a Sketching Language for User Interface Developers

LADDER (Hammond & Davis, 2005) is a language that enables one to describe the characteristics of a shape by defining hard and soft constraints. The hard constraints need to be satisfied in order for the shape to be recognized whiles the soft constraints could aid the recognition by specifying relationships that usually occur. For example if a line is drawn followed by the drawing of a triangle this could mean that an arrow was drawn therefore aiding the recognition. Note however that even if this constraint is not met the arrow should still be recognized.

The authors performed a study to identify the natural vocabulary for describing shapes in order to ensure a more intuitive syntax and vocabulary.

This resulted in the conclusion that a hierarchical symbolic shape based language (new shapes are defined in terms of previously described shapes) is more intuitive as well as that shaped-based geometrical properties are more intuitive than feature-based properties.

An important notion is the notion of a shape group. It states that shapes within a certain domain are often grouped. In the case of ORM, a group of two lines and boxes is used grouped to create roles. This paper states that in case of such groups the movements of one shape influences the movement of another.

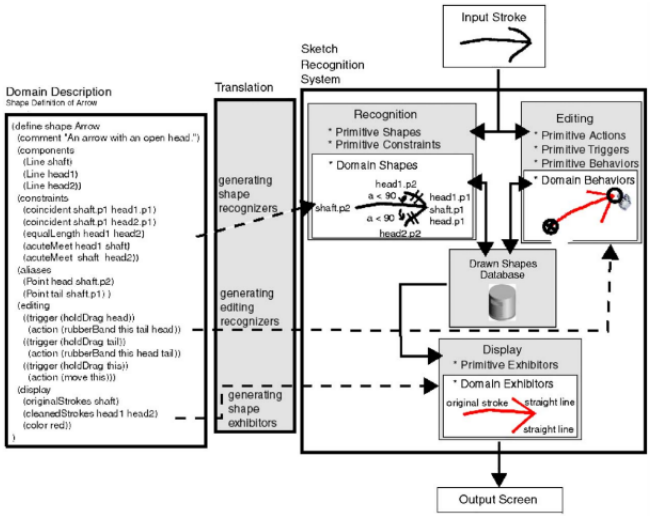


Figure 2.3: The LADDER System Framework

Source: (Hammond & Davis, 2005)

2.4 Smart Study

Smart Study (Van Thienen, 2014a) is an educational platform created by Dieter Van Thienen as part of his Master's thesis as a means to facilitate the learning process of primary school children using a smart pen and tablet. The Echo smartpen is used in combination with specially designed paper and an Android application (see Figure 2.4). The smartpen is used to capture the answers of a student. An intermediary program then extracts the handwritten data and stores the automatically corrected answers and other processed data into a database, where it can be fetched by the Android application. The application itself provides feedback about the correctness of the answers and gives an overview of the student's filled out exercise answers, solutions and overall score.

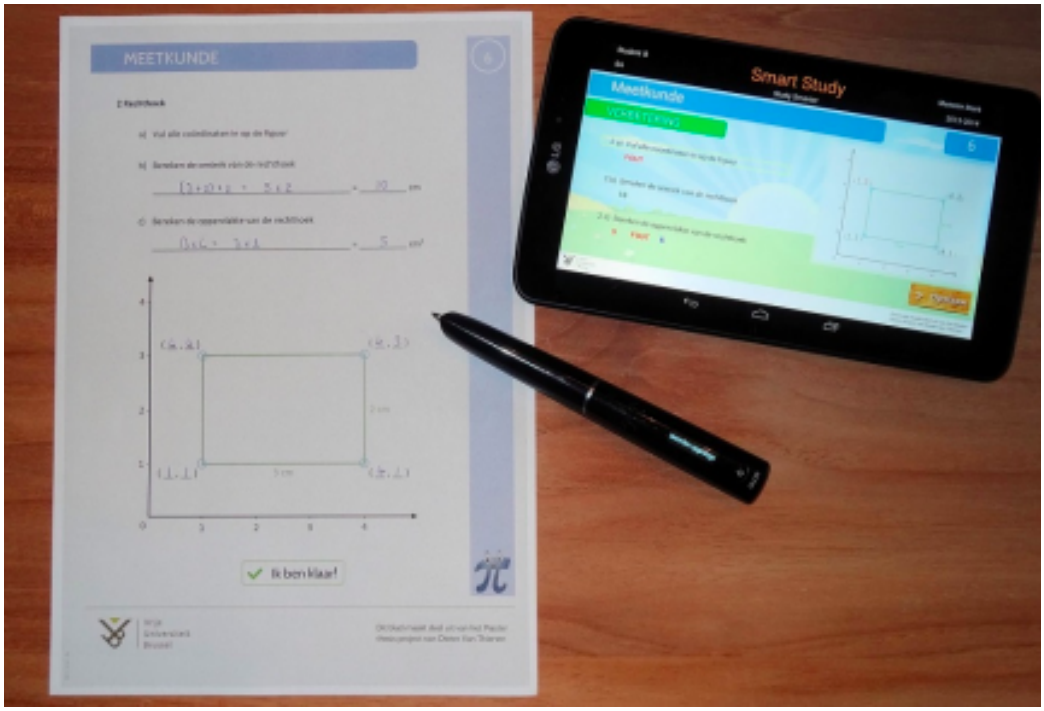
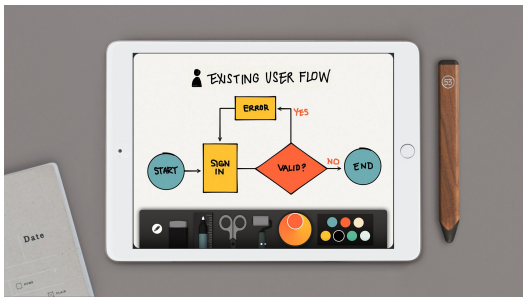


Figure 2.4: The Smart Study Components

Source: (Van Thienen, 2014a)

2.5 Paper by 53

Paper by 53 ™(FiftyThree, 2015) is a mobile application for iPad, combining their THINK Kit with their Pencil and Paper application. The THINK Kit is an Intention Engine which recognizes and corrects objects in real time. It enables a user to create objects, fill them, cut them and move them freely over the digital page. The other element is 53's pencil which can be seen in Figure 2.5b. This pencil can be used like an actual pencil to draw on a (iPad) tablet. The system recognizes surface pressure and is able to detect and ignore hand palm interactions, by flipping the pencil it works as an eraser, and you can use your finger to smooth rough edges and blend colors. The pen uses Bluetooth to interact with the tablet as well as a sensor to detect contact with the surface.



(a) Think By 53

(b) Pencil By 53

Figure 2.5: 53's products

Source: (FiftyThree, 2015)

3 Background

In this chapter we will discuss the different elements used in the development process in order to provide the reader with the background necessary for understanding the problem we formulated in the introduction and the proposed solution. We discuss existing modeling tools for ORM, the ORM language itself, the different kinds of digital pens available, and XML as well as InkML, which is an XML format for digital ink.

3.1 Conceptual Modelling Tools

Conceptual modeling is an important part of the software development process. Over the years different techniques and tools have been developed in order to easily create conceptual models. The tools can easily be divided into two categories. The ones that assist in creating one specific kind of model and those that provide an easy way to draw any kind of model without any restriction. The first category has as the advantage that it discourages/forbids wrong models and enables the further processing of the models. The second category on the other hand has the advantage that it enables the user to create many different kinds of models with the same tool and can even be used to mix concepts of different languages within one model. These kinds of tools usually offer a simple canvas with the possibility to link different shapes. In the following sub-sections we describe an example tool from each

category. The tools are well known and widely used.

3.1.1 Visio

An example of a tool of the first kind is Visio (Microsoft, 2015). This software was created for modeling databases and software. For databases, it provides support for ORM, Entity Relationship (ER), and relationship modeling for respectively the conceptual analysis and logical design of Relational Model tables. Modeling for software in general is supported by UML (Evans & Hallock, 2003).

This tool makes it easy to create an ORM model and can convert the model into Relational Model tables. However this conversion is not always errorless. Unfortunately Visio can only be used within a Microsoft Windows environment. Figure 3.1 shows a screenshot of the application.

Figure 3.1: Screenshot of visio

Source: (Microsoft, 2015)

3.1.2 Draw.io

Draw.io (Draw.io, n.d.) is a diagram editor that falls into the second category. This editor is a web-based application, meaning that it is platform indepen-

dent and that schemas can be saved to the cloud and accessed from any PC with an internet connection and a browser. Figure 3.2 shows a screenshot of the application. It is an open source application that is easy to use and contains many shapes as well as the possibility to create your own shapes or search for images across the Web to use them within the tool. Models created with this tool can easily be exported to different formats and shared via different media. It is also integrated with Google Drive. Another tool that could be placed in this second category, but will not be discussed, is Lucidchart (Inc, 2015).

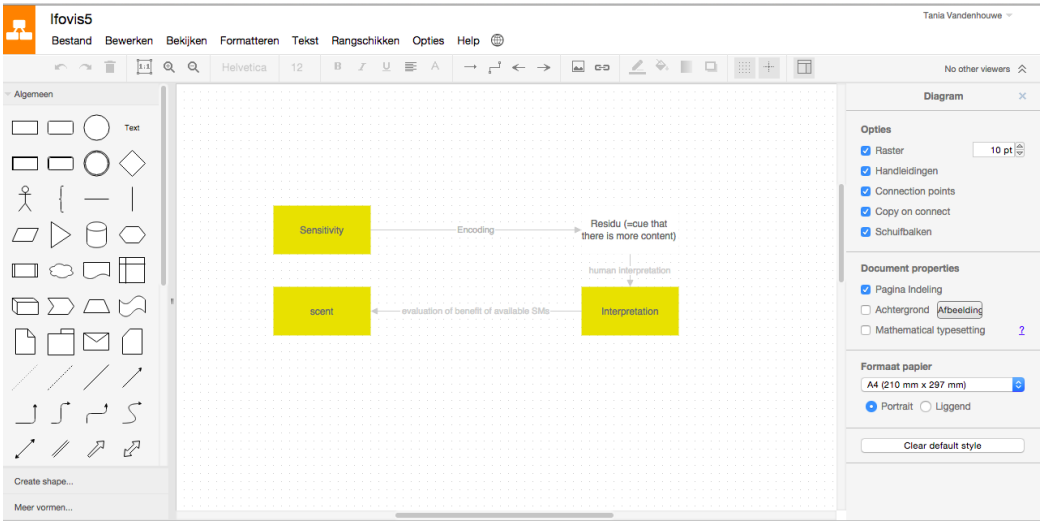


Figure 3.2: Screenshot of draw.io

3.2 ORM

ORM or fact oriented modeling is a conceptual modeling method, that is designed to create and query database models at the conceptual level. Its modeling language has a rich graphical notation that contains both text and graphical symbols making it easy to express constraints. ORM also includes procedures for designing and transforming ORM schemas into relational database schemas.

ORM's core components are *fact types* (relationship types). These can be unary, binary, ternary, or n-ary. Some n-ary fact types can be decomposed in two subsets of its roles. If an n-ary fact type is elementary (non-decomposable), each uniqueness constraint spans over at least n-1 roles. This

is Nijssen's "n-1 rule". As this condition is necessary but not sufficient for a fact type to be elementary an n-airy fact type where the uniqueness constraint spans over less then n-1 facts is decomposable. However this does not guarantee that a fact type where the uniqueness constraint spans over at least n-1 fact types is elementary (Meersman, 2000).

Note that attributes are not used in the ORM language. Instead, a distinction between lexical or value types and non-lexical object types are made. This has several advantages:

- Semantic stability, meaning that ORM schemas are less sensitive to changes.
- Natural verbalization, meaning that ORM schemas can easily be translated into natural language sentences to explain them to customers.
- It enables population, meaning that example population can be given for fact types.
- Null avoidance; meaning that null values are not needed.

3.2.1 Conceptual Schema Design Procedure

As mentioned in the previous section, ORM includes an effective modeling method (Meersman, 2000). This method provides a step-wise procedure to construct and validate models and is referred to as the Conceptual Schema Design Procedure (CSDP).

An overview of the CSDP is given below:

1. Deduce elementary facts from examples (informal verbalization by domain experts and formal rephrasing by the modeler)
2. Create/draw fact types and "test-populate" with sample instances
3. Combine object types and identify derived facts
4. Add the constraints
    - Uniqueness constraint
    - Mandatory role constraint
    - Set and sub-type constraints
    - Other constraints
5. Use positive population to do final checkings

3.2.2 ORM(2) Notation

The main graphical symbols for the ORM 2 notation are depicted in figure 3.3 (the notation used by ORM 2 is slightly different from the original notation used for ORM). Object Types and Value Types are depicted respectively by a soft rectangle (symbol 1) and a soft dashed rectangle (symbol 2). Each Object Type has a reference scheme, which indicates how each instance can be mapped to a combination of one or more values. This mapping may be abbreviated by displaying the reference mode in parentheses (symbol 3).

There are two kinds of relationships; existential facts, which are used for preferred relationships, and elementary facts. The exclamation mark (symbol 4) declares that an object type is independent (instances may exist without participating in any elementary facts).

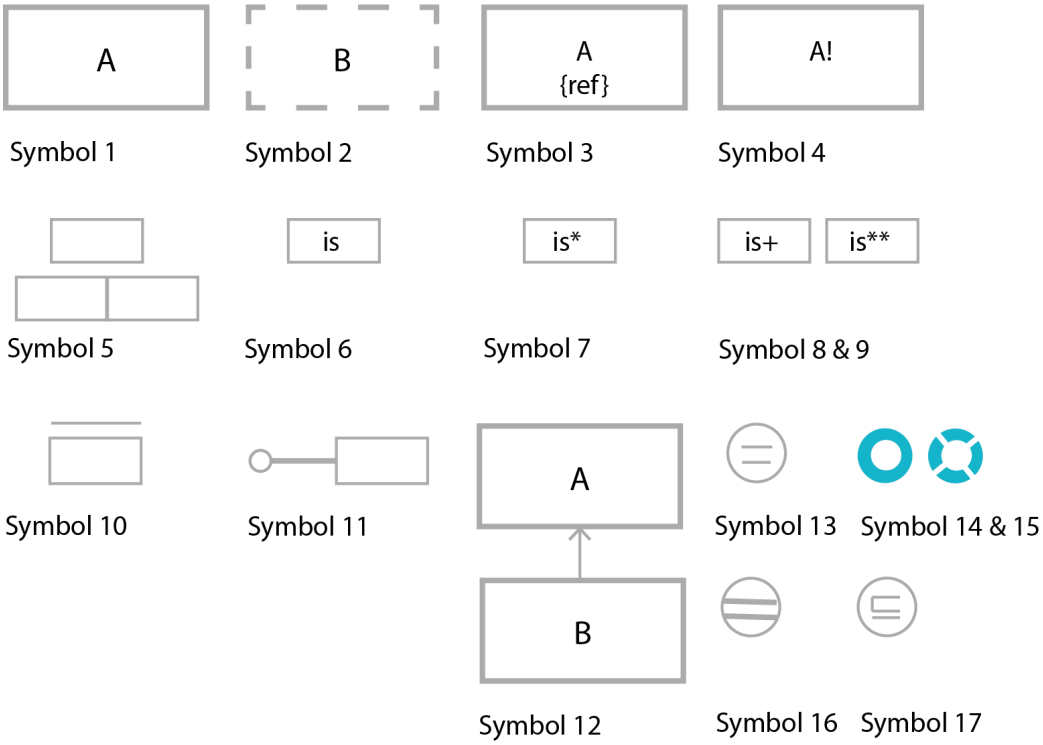


Figure 3.3: ORM graphical notations

A fact type is a relationship between one or more Object Types or at most one Value Type. Each role of a fact type is depicted by a box (symbol 5), which is linked by means of a straight line to one Object Type or Value

Type that plays the role in the relationship. Roles may be given role names (symbol 6). If the fact type is a derived fact type, this is indicated by an asterisk "*" (symbol 7). If a fact type is a semi derivation or a derivation which is stored this is respectively represented by a "+" and a double asterisk "**"(symbol 8 and 9).

The uniqueness of the value of a role(-combination) is indicated by putting a line above the role(s) involved (symbol 10). A mandatory role is indicated by a large dot placed at the object type's end of the connection line(symbol 11). Sub-typing between object types is indicated with an arrow pointing from the sub-type to the super-type (symbol12).

There are also symbols for constraints,e.g. for indicating equality between roles or fact types (symbol 13), choice (inclusive and exclusive or) between roles, fact types and sub-types (symbol 14 and 15), preference (symbol 16)(Halpin, 2009) (Halpin, 2009),(Halpin, 2005), (Halpin, 2006), (Halpin, 2001).

3.3 Digital Pen

There are different kinds of digital pens, using different methods to capture the digital ink. Each one has its own pros and cons. We briefly review them, a more profound study of digital pens can be found in (Van Thienen, 2014b).

3.3.1 Motion based

This type of digital pen is equipped with different motion sensors (gyroscopes, accelerometers and/or magnetometer). As this pen is able to determine its position based on the 3D-movements, the pen makes it possible to write on any kind of surfaces (Wang & Chuang, 2012).

An actual example of such a smart pen is the Lernstift (Lee Garber, 2013)/VibeWrite which can be seen in Figure 3.4. This pen is a stand-alone device, meaning that it does not need a support (paper) or a communication structure, as the recognition happens within the pen itself. As the pen can communicate via Wi-Fi, it can retrieve what has been written and share or edit it.

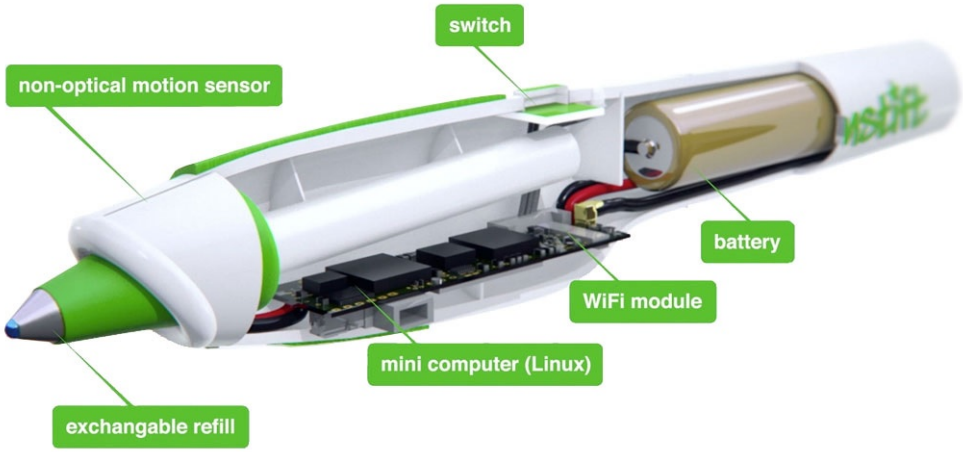


Figure 3.4: The VibeWrite pen

Source: (Postscapes, n.d.)

3.3.2 Positional

This kind of pen functions by detecting the position of the pen with the help of an external device. This external device can be a graphics tablet or a small device that is clipped on a paper. We discuss three of these pens.

Wacom Tablet

A Wacom Bamboo (Wacom, n.d.-a) tablet is an example of such an external device as can be seen in Figure 3.5. As paper is not required, the user needs to get used to write on the tablet while having the visualization on the screen. The user has to learn not to look at what its hand is doing but what is actually being written or drawn.

Wacom Inkling

An example of a positional pen that does use paper is the Wacom Inkling (Wacom, n.d.-b). It captures the position of the pen with an external device that is clipped to a page. See Figure 3.6. As it works with any kind of paper it is possible to switch easily between different paper formats and types, and everything drawn and written is captured digitally as well as analogously.

Figure 3.5: A Wacom tablet

Source: (Missfeldt, 2015)

Figure 3.6: The Wacom Inkling Device

Source: (CMedia, 2015)

Touch Devices

Some examples of positional capturing devices are devices like smart-phones and tablets. These detect the position of a pen or finger based on touch sensors (capacitors or heat sensors). This obviously has the advantage that no actual pen is needed, but has the disadvantage that the capturing of strokes is not that accurate. See figure 3.7 for an example of such a touch based device.

Figure 3.7: An Apple iPad

Source: (Personalized, 2014)

3.3.3 Camera Based

The last type of digital pen that will be discussed is one with an embedded camera. This camera interacts with special paper, which contains a dotted pattern. When the pen touches the paper, the pressure sensor is activated, enabling the image recording of the camera. This camera will recognize the dotted pattern and knows its position on the paper and sends the coordinates of this position. This is the kind of pen that will be used in this thesis.

Figure 3.8: The Echo smartpen

Source: (Matt, 2015)

3.4 Interactive Paper

As mentioned before in Chapter 1.3, intelligent paper consists of three elements: a pen, a support, and a communication infrastructure. For camera based pens such as the Livescribe Echo Smartpen the support is paper with the Anoto dot pattern (DigitaalSchrijven, 2011) printed on it.

The dots themselves create a grid of XY-coordinates that the pen is able to "read" i.e. making it possible for the pen to uniquely identify its position on the page as well as identifying the page on which it writes, enabling the pen to capture handwriting and detect pen events. Figure 3.9 provides an overview of the working of Anoto paper and its structure.

3.5 Digital Ink

We can speak of a pen-based interface when the movement of a pen is captured as digital ink. This digital ink can be captured in different kind of ways, using radio frequency, physical pressure, optical tracing and so on. This digital ink can then be passed to recognition software that will convert the pen input into the appropriate computer actions or store the pen strokes in a document for later use.

3.6 XML

XML is a language that was developed for structuring data with the main goal to be human readable as well as machine readable. An XML document follows a tree like structure where each branch is enclosed by tags.

There are three kinds of tags:

- a start-tag
- an end-tag
- an empty-element tag

Each XML element begins with a start-tag, which can contain element attributes, and ends with an end-tag. Between this two tags content or child elements can be added.

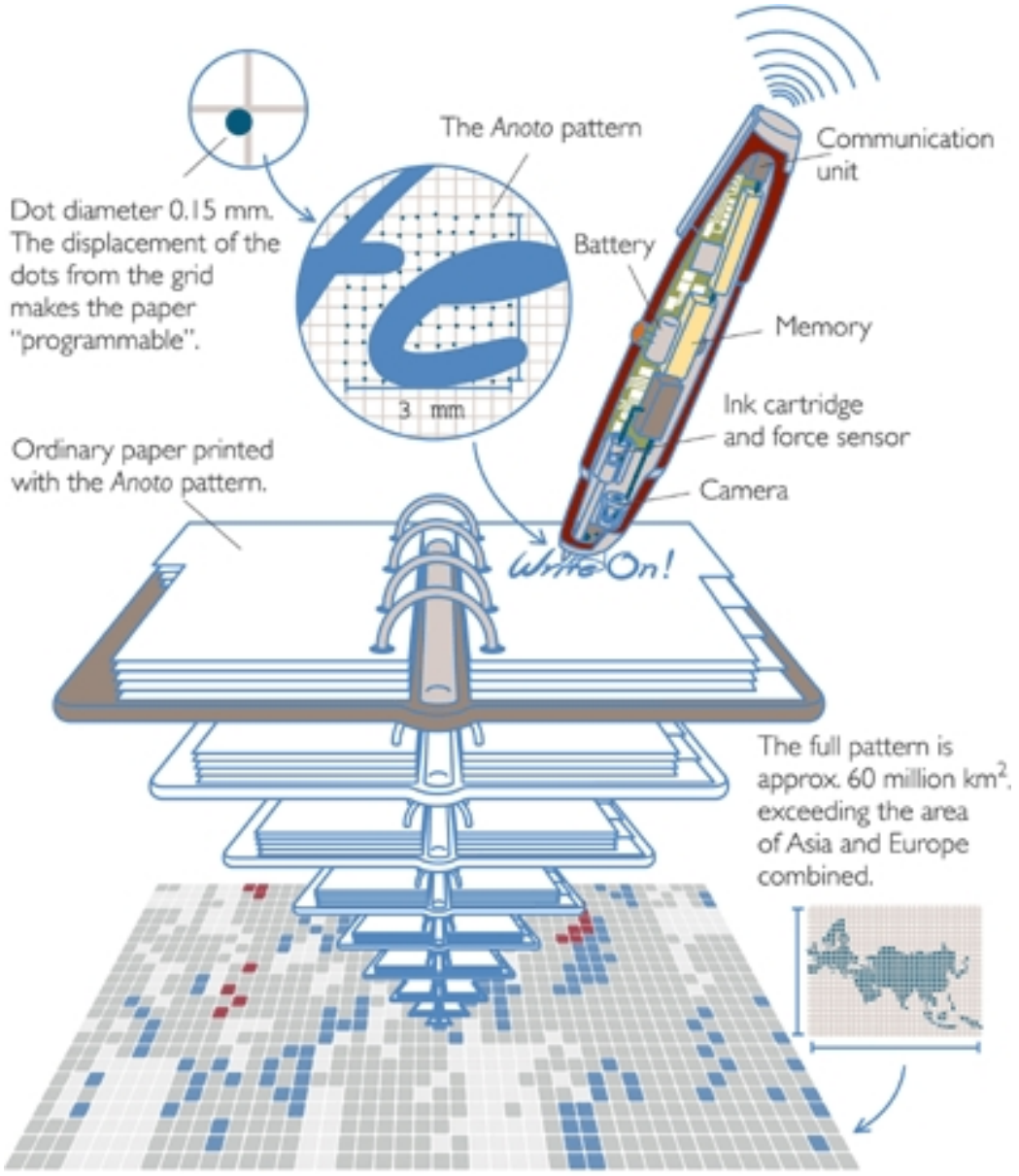


Figure 3.9: Anoto explained

Source: (Media, 2015)

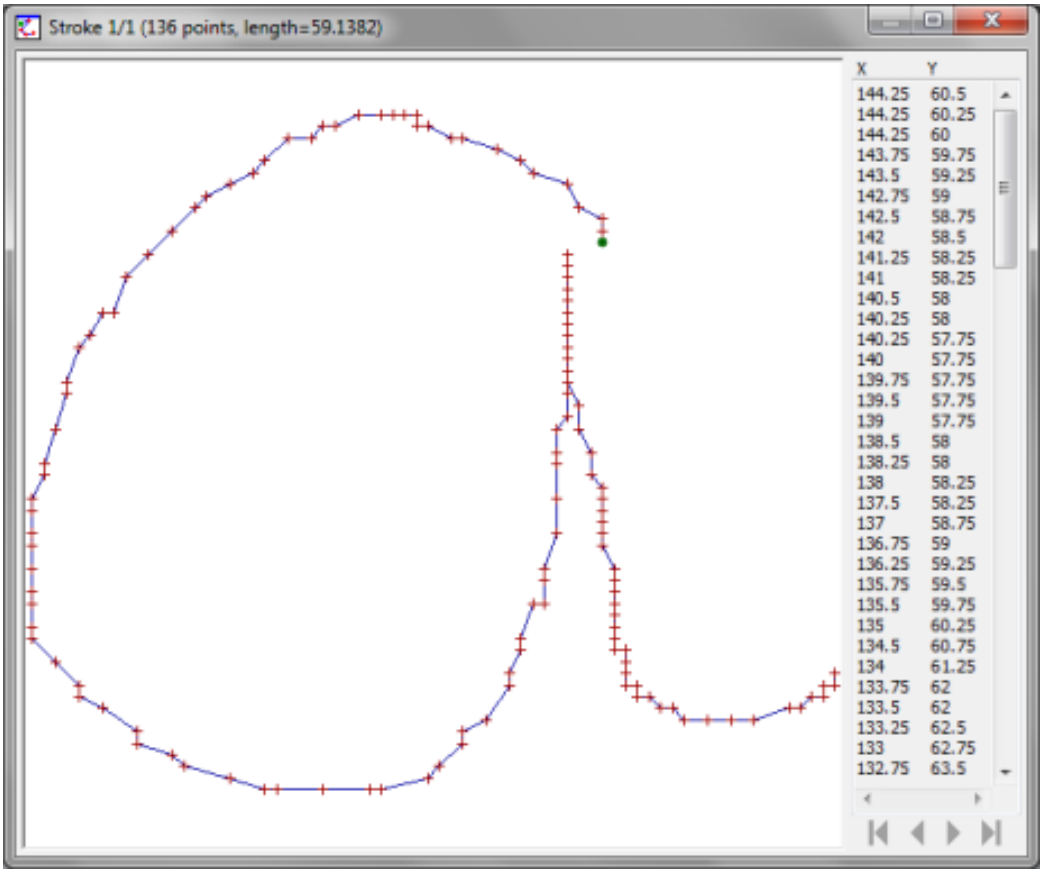


Figure 3.10: An example of a trace/stroke in digital ink

Source: (MyScript, 2015d)

3.6.1 Graphs

Graphs basically represent networks where nodes are connected to each other via edges. Graphs can be used to visualize networks, interact with networks, perform an analysis on the network, or apply algorithms such as Dijkstra's algorithm for finding the shortest path between nodes.

Note however that a graph can contain a circular structure whilst XML trees follow a linear pattern, so a system to encode a loop inside a tree has to be found.

3.6.2 InkML

In order to create a uniform format for digital ink that can easily be exchanged and understood, a mark-up language format for storing the pen strokes was created. This format is called Ink Markup Language (InkML) (Chee, Froumentin, & Watt, 2006). This language basically captures the strokes of a digital pen as different coordinates along the strokes. The pen records its position at different time intervals while the pen tip is touching the surface. As long as the pen is down all the different points recorded count as a stroke.

It is important to note that InkML was not designed to store semantic information, such as the fact that the ink should be recognized as handwriting or as a shape, or to store contextual information about the ink, such as in what field of a form the ink was written. But InkML can be extended by including XML from other schemas at specific locations in a file or stream. Additionally, it can also be embedded within another XML document.

Elements

The current InkML specification defines the following set of primitive elements.

- <ink>: the root element of the file in which all the different strokes are stored
- <trace>: this is equivalent to a stroke and represents a sequence of contiguous ink points captures as the X -and Y- coordinates.
- <traceFormat>: contains information about what the coordinates represent; values for pen position, angle, tip force button states and so on.
- <inkSource>: contains information about device that was used; sampling rate and resolution
- <brush>: contains information about the attributes of ink such as color, width pen modes (eraser or writing) and writer identification.
- <traceGroup>: if traces share the same characteristics they can be grouped together by this element so that one same brush element can be used on the whole group
- <traceView>: groups traces by reference.

- <context>
- <annotation>: labels trace with attributes

InkML traces are limited to Cartesian coordinates and do not support non-ink events (although this can be handled via annotations).

4 Used Technologies

In this chapter the technologies, which were used in the development process, are discussed in some detail.

4.1 Livescribe

The Livescribe company is the global leader in the design and manufacturing of smartpens (Livescribe, 2015) . Their pens make use of the camera technology and specially designed paper with an embedded dotted pattern on it. They currently offer three different kinds of smartpens; the Livescribe 3 smartpen, the Livescribe Wi-Fi and the Echo smartpen. The main difference between these smartpens is the transition mode they use. The Livescribe 3 smartpen (I. Livescribe, 2015b) uses Bluetooth technology, the Livescribe Wi-Fi smartpen (I. Livescribe, 2015c) uses Wi-Fi or USB, and the Echo (I. Livescribe, 2015a) smartpen only transfers data via USB. Other differences are that the Livescribe 3 smartpen requires IOS 7 or newer and convert handwriting to text automatically whilst the echo smartpen works on both Windows and Mac but requires MyScript for handwriting recognition. All the pens record what is being written down and store it in their local memory. They also have an OLED display screen (except for the Livescribe 3 smartpen) and the possibility to record and replay audio.

4.1.1 Requisites

In order to use the Livescribe smartpens you need to acquire the pen as well as some writing medium (Anoto dotted paper). This paper can be bought or printed at home if the printer is a color laser printer that is Adobe PostScript compatible and can print at 600dpi or higher.

4.1.2 Development Kit

When developing a custom application for Livescribe smartpen three elements are involved.

1. A paper design, which contains all the necessary fields, buttons, text, and images.

2. A penlet, which is an application that runs on the smartpen and tells what should happen when what action is done (pen down, pen up, stroke drawn, ...) on a field.

3. An actual application, which extracts the data from the pen and does the necessary computations and transformations.

The paper design is rather straightforward but seems to require an active area to activate the penlet. Each paper has one or more penlets attached to it, meaning that it is the paper that decides which penlet is activated and not the penlet itself. The coupling between paper and penlet can be done by right clicking on the paper design, Select Properties, Select the Document tab, and then clicking the Edit Application List to open a dialog as illustrated in Figure 4.1. In this dialog we can add the penlets which to couple to this paper.

More information about creating a customized paper and penlet can be found in the (I. Livescribe, 2007-2009) document.

4.2 MyScript

MyScript is a company that develops software for shape, text, math, and multi-content recognition. They offer three types of developer kits: one for desktop application development, the Software Development Kit (SDK), one for cloud driven application development, the Cloud Development Kit (CDK) and one for tablet application development the Application Toolkit (ATK).

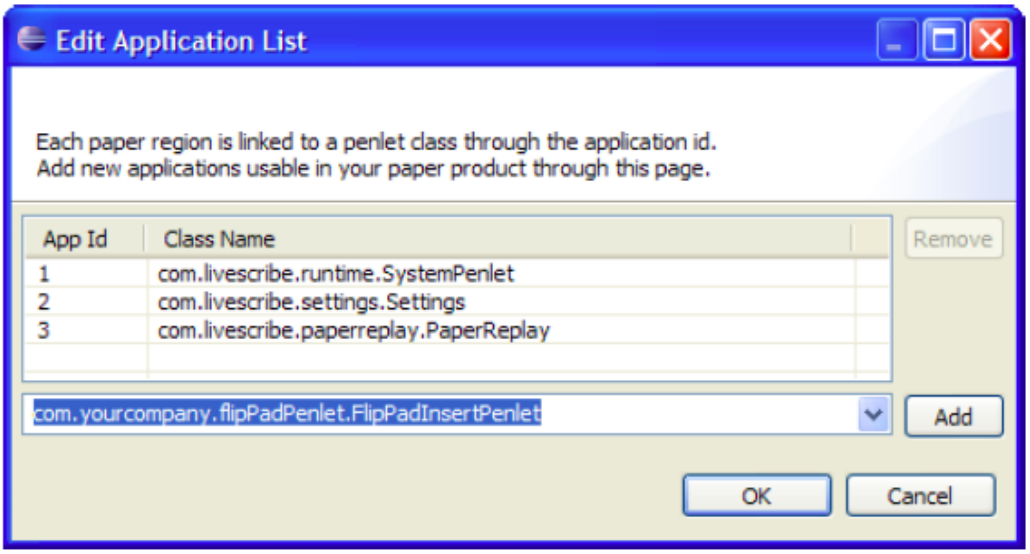


Figure 4.1: The dialog window for adding penlets to specific paper designs

Source: (I. Livescribe, 2007-2009)

MyScript exclusively works with digital ink as input. This ink is defined as a series of strokes where each stroke is defined by multiple x and y coordinates ordered in time. A stroke is recorded from the moment that the pen touches the smart paper surface until the moment the pen is lifted from the surface.

Once a stroke is captured this stroke is send to the MyScript recognizer in the form of a floating integer x and y coordinates. The recognizer will then process the digital ink input and match it with the most likely form, letter, music-note, or mathematical expression. As handwriting recognition is a tricky business (a letter O could be recognized as the number 0 or as an oval shape) there are various recognizers specialized in different kinds of recognition. These are text, math, shape, music, mixed-content and doc model recognizers.

Every recognizer relies on resources which are binary assets containing a priori knowledge that can be related to the symbols to recognize or other rules depending on the form of input. These resources provide clues for the recognition and need to be configured before the recognizer is applied to a stroke. It is important to select the proper resources for a certain task in

order to achieve optimal recognition.

In the case of an ORM where both text and shape recognition is necessary it is thus recommended to create a switching system between both recognition modes. However this would undermine the usability of the pen as a modeling tool.

After the recognizer has processed the input, the result is delivered in the form of a hierarchical tree. This tree contains a list of candidates for the recognition and is ranked on the probability of the candidate. This ranking depends on metrics, after which the recognizer will select the candidate with the highest likelihood.

4.2.1 Cloud Development Kit (CDK)

The CDK supports different recognition options based on what code you wish to use. This kit provides means for text, shape, math, and multi-content recognition. In order to do the recognition the digital ink has to be converted into a specific format depending on the programming language you wish to use.

The cloud application is a Hypertext Transfer Protocol (HTTP)-based set of services based on a Representational State Transfer (REST) architecture and uses the MyScript recognition engine (MyScript, 2015b) in a client-server configuration. Therefore the use of the MyScript cloud application requires a HTTP client, a JavaScript Object Notation (JSON) parser/writer and an Internet connection. An application key is also required. As a developer it is possible to get a key for a trial period of 90 days by registering on `https://dev.myscript.com/`.

Once registered, multiple samples for the use of the application are provided in different programming languages. For Java there is only a sample for handwriting recognition. For C and JavaScript there are samples for equitation, handwriting, shapes, shapes and text, and music recognition. For JavaScript there is also the possibility of handwritten recognition in a boxed field format for individual letters and number recognition.

All recognition requests require the strokes and are made via a POST method to the server. This request has to be made to the right kind of server, therefore there are different recognition URLs.

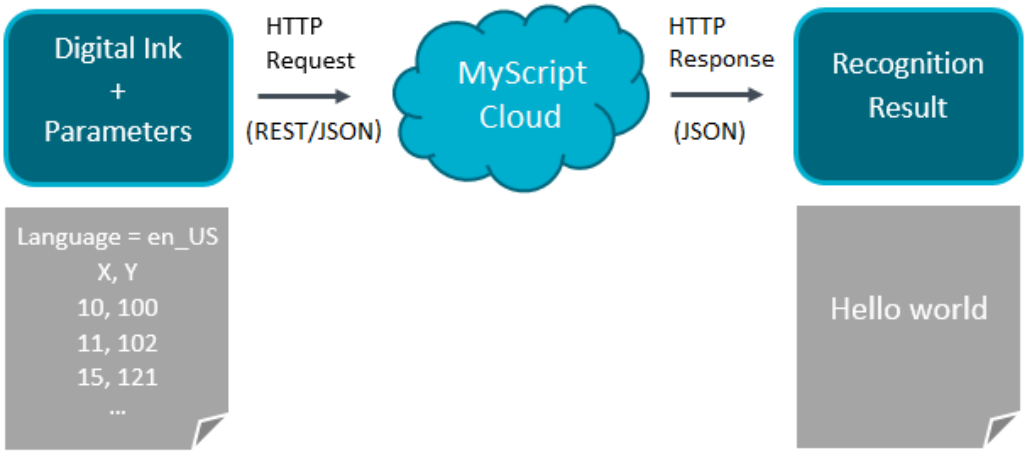


Figure 4.2: The MyScript Cloud Architecture

Source: (MyScript, 2015c)

The JavaScript code requires the ink strokes to be structured in an array with JSON objects of the form {type:stroke, x:[array of x values of that stroke], y:[array of y values of that stroke]. This requires the JQuery library `http://jquery.com/`

For the C code each stroke has to be represented by two rows. The first row contains the abscissa set (x-coordinates) and the second contains the ordinates set (y-coordinates) this can be parsed to a JSON flow with the help of cJSON `http://cjson.sourceforge.net/`

For Java the external library Jackson JSON processor is recommended to create the required JSON flow`http://jackson.codehaus.org/`

4.2.2 Text Recognition

Text recognition request are made to the `https://cloud.myscript.com/api/myscript/v2.0/hwr/doSimpleRecognition.json` URL and requires some extra information like text language, how the digital ink is structured and if the result should contain more details. The possible parameters that can be added to the Text recognition request are given in Table 4.1 .

The handwriting text recognition does not simply guess one letter at a time but recognizes whole sentences. This makes it possible to recognize

Table 4.1: possible parameters for a text recognition request

Structure	Parameter	Mandatory
apiKey		YES
instanceID		NO
hwrInput	hwrParameter	YES
hwrInput	inputUnits	YES
inputUnits	hwrInputType	NO
inputUnits	components	YES
hwrParameter	language	YES
hwrParameter	hwrInputMode	INO
hwrParameter	contentTypes	NO
hwrParameter	subsetKnowledges	NO
hwrParameter	useRecources	NO
hwrParameter	userLkWords	NO
hwrParameter	resultDetail	NO
hwrParameter	hwrProperties	NO
hwrParameter	textCandidateListSize	NO
hwrParameter	wordCandidateListSize	NO
hwrParameter	characterCandidateListSize	NO
hwrParameter	wordPredictionListSize	NO
hwrParameter	wordCompletitionListSize	NO
switchToChildren		NO

complex handwriting. The guessing process uses lexicons and language models in order to understand how parts of a language come tighter and interact. The possible results for a character or sentence are returned in a tree like structure according to their probability and are given a score from 0 to 1 to reflect the level of confidence in the given result.

4.2.3 Shape Recognition

For this kind of recognition the URL for the request is `https://cloud.myscript.com/api/myscript/v2.0/shape/doSimpleRecognition.json` and the possible parameters are:

Table 4.2: possible parameters for a shape recognition request

Structure	Parameter	Mandatory
apiKey		YES
instanceID		NO
shapeInput	components	YES
shapeInput	userResources	NO

The shape recognizer engine uses resources, which describe a set of shapes and components and will send back the components in a beautified form (if requested) and give the shapes information in order to enable the reconstruction of the shape as a vector object. The shapes that this engine is able to recognize are listed in Figure 4.3.

4.2.4 Analyzer Recognition

This recognition mode can, for now, only recognize a mix of text and shapes. The result sent by this recognizer will describe the following elements:

1. Text candidates with information about positioning, orientation, justification, bounding box and font size

2. Shape candidates

3. Structural elements such as groups, tables and underlines with the relation between the different objects.

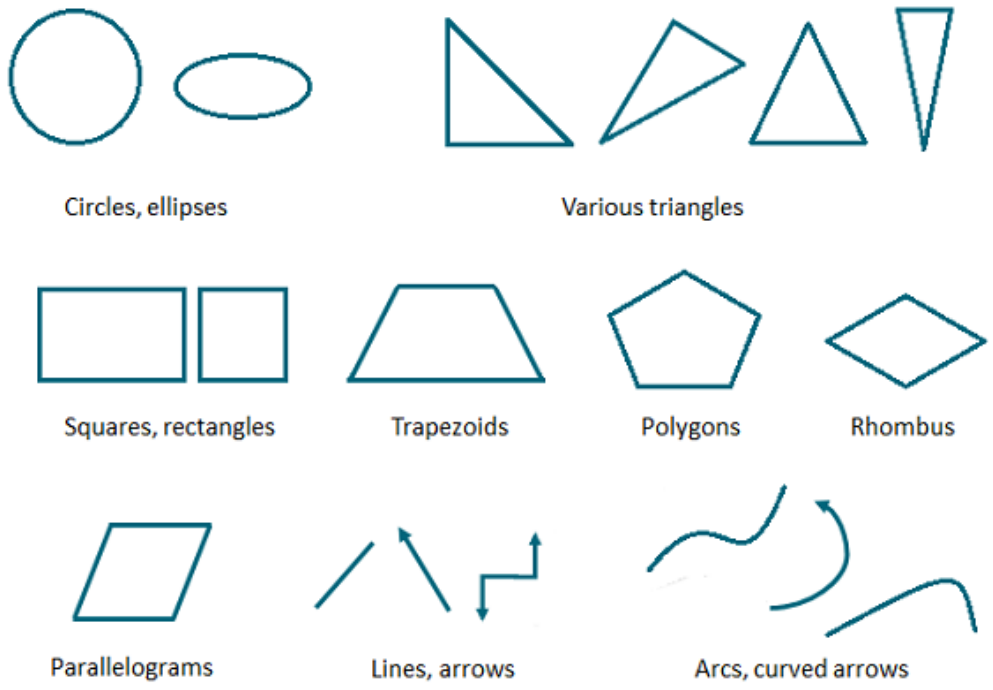


Figure 4.3: The different shapes recognized by MyScript

Source: (MyScript, 2015e)

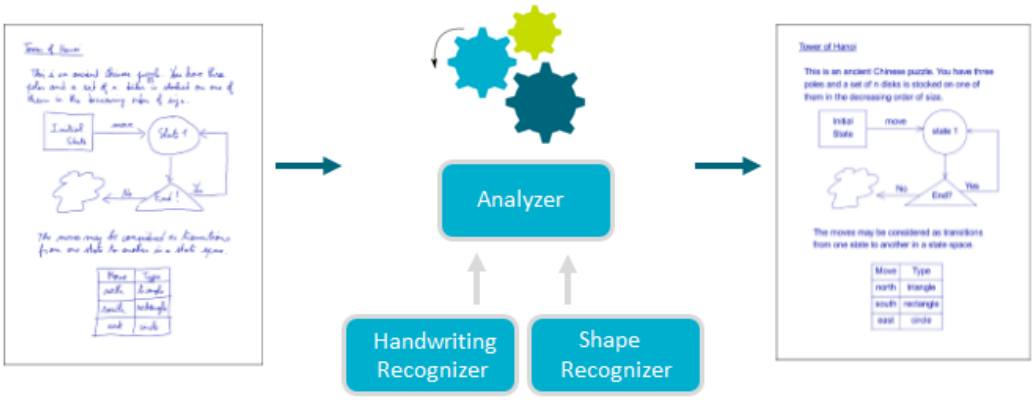


Figure 4.4: Analyzer Workflow

Source: (MyScript, 2015a)

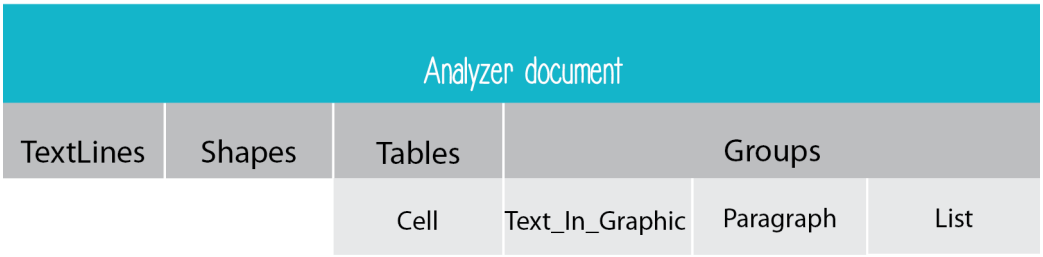


Figure 4.5: Analyzer result structure

Table 4.3: possible parameters for a mixed-content recognition request

Structure	Parameter	Mandatory
apiKey		YES
instanceID		NO
analyzerInput	components	YES
analyzerInput	parameter	YES
analyzerInput	userResources	NO
switchToChildren		NO

4.2.5 Limitations and Potential Errors

Unfortunately there are some limitations to the recognition engine:

1. Only digital ink can be sent.
2. The service is limited to one single type of input for Text, Shape, or Equitation
3. The analyzer service is limited to the text and shape input type.

A full overview of the limitations and possible errors can be found at `http://doc.myscript.com/MyScriptCloud/2.2.0/html/appendices/limitations.html`

4.3 mxGraph

MxGraph is a JavaScript diagramming component that works on all major browsers, including touch devices on which draw.io was built. It is a market

leader for JavaScript Graph visualization. It is a library that enables interactive graph and charting applications to be quickly created. MxGraph is a derivate from JGraph which was developed for the Java language.

`https://www.jgraph.com/`

MxGraph Model and Cells

The mxGraph model describes the structure of a graph. It is possible to make additions, changes, and removals within a graph structure as well as to create structure or set visual states such as visibility grouping and style.

However even if the mxGraphModel is the underlying structure that stores the graph, all changes happen throughout the mxGraph class. The key Application Programming Interface (API) methods are the following:

- mxGraphModel.beginUpdate()
- mxGraphModel.endUpdate()
- mxGraphModel.addVertex()
- mxGraphModel.addEdge()

Two components can be distinguished: a vertex and an edge. Both require a parent, id, value, and style as parameters. Vertexes also require x-, y- values and a width and height. Edges require the additional source and target parameters.

In the mxGraph model each element of a graph (edges, vertices/nodes or groups) is called a cell. When a new cell is created, a value, user object, a geometry and as style are required for the constructor. The style follows the concept of a (CSS) style sheet, the geometry provides additional information about the position of a cell and the user object provide the context and stores the business logic associated with the cell.

`https://jgraph.github.io/mxgraph/docs/manual.html` 2006-2014 by JGraph Ltd.

5 Development

In this chapter, the different stages of the development process will be explained in detail. Basically, the development of the software is divided in five stages each with their own challenges.

The first stage is the extraction of the digital ink. The second stage is recognition of the handwritten shapes and text. The third stage is the conversion to a format that can be visualized. The fourth stage is the selection of an appropriate Modeling tool for displaying the result, and the fifth and final stage of the development process is the automation of the whole process from the extraction of the ink to automatically opening the created document in the selected modeling tool. The final product developed as a result of all these steps is the Draw2Model application.

In Figure **??** an overview of the different options for each step is given as well as their pro's and contra's.

5.1 Extraction of the Digital Ink

In order to achieve the handwriting recognition, the digital ink first needs to be extracted. As the SDK was no longer supported we searched for a way to extract the ink data with the available echo desktop software. Because

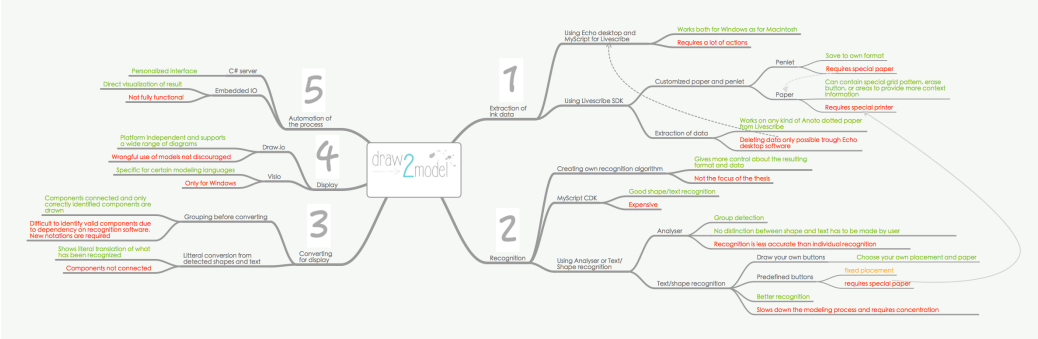


Figure 5.1: An overview of the considered options for each stage of the process

extracting the data in this way is not straightforward we decided to use the Livescribe SDK (although it was no longer supported) to explore the options of a customized penlet and the option to use the SDK for extracting the data from the standard Anoto papers provided by Livescribe.

5.1.1 On a Macintosh Computer

As the development kit for the Echo smartpen was only available for Windows, an alternative way to extract the digital ink had to be found in order to also enable Macintosh users to use the application. It is possible to access the ink files by installing the Echo desktop software and activating the MyScript license. Once this is done the document of which the ink file has to be accessed has to be opened with the MyScript software included in the Livescribe application. This will allow the user to export the file to mail, word, or as an image. When selecting the export to mail option a ".notes" file can be found in the attachment of the mail. This is the ink file and can be downloaded from mail to the computer. Note that the extraction of the .notes file requires a lot of steps and is not very user friendly.

5.1.2 On a Windows Computer

The same process as with a Macintosh computer is possible in Windows, however since we still had a Livescribe SDK for windows at our disposition we explored the option of creating our own penlet application and paper design and looked for a way to access the data of standard paper with the help of the Livescribe SDK.

We explored the added value of a customized penlet and paper design.

The customized paper and penlet would enable us to create a tool similar to the one described by (Dachselt et al., 2008) and it would also enable us to convert the strokes directly to an appropriate format similar to the one which can be extracted from the Echo desktop program.

The creation of a customized paper design and penlet similar to the UML palettes approach was discarded after some discussion about the usability of such tool. Even if this kind of approach would provide us with a better context (i.e. we know in advance the concept that the user is drawing), and thus a better recognition, it would require the user to select the appropriate shape from the palette before actually drawing it on the sheet of paper. This is an additional step that could impact the usability; also this approach could result in serious discrepancies between what is drawn and what is actually recorded.

Therefore we preferred to explore how feasible it would be to only use what a user draws, without requiring any actions from the user's part to provide us with the information about the shape he is drawing. This approach would also enable users to use the whole surface of the paper (as no interactive fields are required) and to use any kind of Anoto dotted paper without requiring any specially designed paper (which requires the user to have access to a color laser printer that is Adobe PostScript compatible and can print at 600dpi or higher).

However this decision still required us to explore the different options for the extraction of the ink data. There were two ways to go. The first one was to create a customized penlet which handles the saving of the data in the required format and the second one was to explore the process of Livescribe itself for the extraction of the stroke coordinates.

The customized penlet and paper would consist of one penlet for multiple pages, which together form a notebook. Each page of this notebook would be completely blank, with the exception of a logo, used to indicate the top and bottom of the page. In this way, the space could be used maximally for modeling.

Our initial plan for the penlet was to save the data directly in the appropriate data format as depicted by Figure 5.3. We however noticed that the pen could not handle the file size that such format would generate and we thus had to change the penlet to save into a lighter format. Unfortunately

even a lighter format seems to require too much data and thus the option of a customized penlet was abandoned in favor of the extraction of the default data from any kind of paper.

The Original Penlet

The Draw2Model penlet has 3 files. One that manages the events (fantastico), one that handles the input and output of the strokes (StrokesIO) and one for the storage of the strokes (StrokeList).

The StrokeList is our own implementation of an ArrayList, as the LivescribeSDK does not support a complete version of Java and thus does not support generics. This implementation generates, by default, an array of size 10 which can grow when necessary and stores Stroke-Objects. This structure is used to store the different strokes of the current page.

The StrokeIO file contains all functions regarding storing and retrieving StrokeLists to and from ink files. The export function adds the necessary headers and stroke markers in between strokes and flushes everything to the according file. The load function retrieves the file associated with the current page and convert the content into a StrokeList.

The last file, fantastico, contains all the event logic. This includes the logic for saving and the logic behind the display. We decided that the strokes have to be saved to a file every time the pen changes pages, the penlet is destroyed, or when the pen is connected with the computer through the USB. Each time a stroke is created we check if the current page is still the page the user was writing on. If this is not the case the current StrokeList is exported to the ink file associated with the current page, the current page is changed to the page on which the stroke was created, the StrokeList of that page is loaded from the according ink file and the stroke is saved in the StrokeList.

The Optimized Data Format

As saving the coordinates directly in a .notes file on the pen proved to be too heavy for the pen (it could not manage to read/write files of the size that is required to store coordinates as a string), an alternative way of saving the digital ink was explored. We tried to save strokes in the following format: the first four bytes would contain the data size described by an Integer (so the data file can contain up to 2^{31} bytes), the data, following, will be "shorts" (sixteen bit integers) containing the number of strokes followed by the number

Bytes	0	1	2	3	4	5	6	7
0	Datasize [43]				#Strokes [3]		#Coordinates s0[2]	
8	s0 x0		s0 y0		s0 x1		s0 y1	
16	#Coordinates s1[3]		s1 x0		s1 y0		s1 x1	
32	s1 y1		s1 x2		s1 y2		#Coordinates s2 [2]	
40	s2 x0		s2 y0		s2 x1		s2 y1	

Figure 5.2: Saving strokes in a byte-wise manner

of coordinates and coordinates themselves in the alternating way of two bytes, containing the number of coordinates for the strokes, and two bytes for each coordinate (thus eight bytes for each coordinate pair). This way we can retrieve strokes and coordinates by doing mathematical calculations (the first stroke's coordinates start at the 4+2+2 = 8th byte) and the original file size is compressed to 2 à 3 times it size. In figure 5.2 a visual representation of the compressed file format is shown. Unfortunately, this compression was still not enough to handle the saving of strokes, so an alternative with different files for one document was envisioned. Actually, this is the same way the pen handles the saving of the strokes.

Using the Default Retrieving Mechanism

Because our penlet should do exactly the same as the default penlet of the pen, i.e. recording strokes, it could be questioned why we are not using the default penlets. The problem with this was how to retrieve the data once it is recorded by a default penlet, as with a penlet we would achieve the data retrieval with an internal function of the penlet which is called outside the penlet.

Therefore we did some research. First of all we sniffed the packages that were sent between the pen and each desktop application. This provided us some insight into the default penlets, which seem to have one individual penlet for each type of page. This unfortunately did not give us a way to access the data, as we have to know the name of the internal function that is in charge of the data retrieval (if there even is such function). Fortunately, we stumbled upon some existing code which enables the retrieval of all Anoto Functionality Document (AFD) files stored on the pen. This code seems to collect all different sub files of one document from the pen and put them all together. Because of this we can retrieve digital ink from any kind of AFD paper and provide the user with a tree-like structure of his documents.

5.2 Communication with MyScript Cloud Recognizer

The recognition of the handwriting is handled by a MyScript cloud service. In order to use this service the coordinates have to be transformed into JSON. As we want the conversion to work both on pages extracted from a Macintosh computer as well as on the files extracted from the special penlet application, both files need to have the same format (see Figure 5.3). As due to the file constraints the penlet could not directly save the information in this format, an additional transformation must be done to create a .notes file or to transform the created file to the required JSON notation.

```
1  2590 1787 //maximal x and y value
2  3
3  1
4  5//number of strokes
5  77688 56 68 // id that indicates that a new stoke will follow
6  4//number of coordinates in this stroke
7  3 4 //first coordinate of stroke (x, y)
8  ... //3 more coordinates
9  77688 56 68  //id that indicates the new stroke
10 ... //repeat line 5 till 8 until all the strokes are done
11 2 1
12 0
13 70
14 0
15 0
```

Figure 5.3: Structure of a .note file

We also explored the option of individual recognition of text and shape by adding some buttons to the paper, which indicated if the recorded ink belonged to a text stroke or shape stroke. This would make the recognition more accurate as the drawn ink would be matched to specific resources (text or shape) and avoid erroneous recognition of a letter O as a circle or of a rectangle's side as a letter "L". However we noted that this approach required the user to constantly keep in mind to switch between shape and text while drawing and to move his/her hand toward the switching buttons, which made the modeling on paper slower. Another downside to this approach was also that the individual recognition does not enable the recognition of groups.

5.2.1 Individual Text/Shape Recognition

For this approach again two possibilities were investigated: Adding buttons to a specific paper design or letting the user draw his own buttons where he/she wants and on whatever Anoto dotted paper he/she wants. This last approach was inspired by the Livescribe piano application where the user draws it's own piano keys.

For this approach to work we let the user draw 3 lines and write a T between the two first lines and an S between the second and last line (|T|S|) before starting to model. The coordinates of these lines are extracted and the boundaries of the buttons are recorded. The only thing left to do is to designate a stroke to the shape strokes or to the text strokes. This is done by iterating over the strokes and each time a coordinate is detected which lies within the boundaries of a button the following strokes are attributed to the strokes of the corresponding kind. Once all strokes are attributed to their right kind the text strokes are fed to the text recognition engine and the shape strokes to the shape recognition engine.

5.2.2 Conversion from .notes Format

The conversion to the JSON format from a .notes format consists of a number of string transformations. First we get the content of the file and split it in an array containing all the different lines of the file. As the file always follows the same structure the identifier of a new stroke can be found on line 5. We use this identifier to split the original file content in the different strokes. At last we iterate over each stroke and push each x coordinate on an array containing the x coordinates of the stroke and each y coordinate to an array containing all the y coordinates of the stroke. These arrays are then returned within a stroke object of the format { "type": "stroke", x: x-coords, y: y-coords };.

```
1 function process(contents) {
      // trim away unnessecary information
3     var c = contents.toString();
      // get each newline
5     var newlines = c.split("\n");
      //get the code that anounces new stroke
7     code = newlines[4];
      //divide the document in strokes
9     theStks = getStrokes(c, code).strokes;
      //create a stroke object
11    processStrokes(theStks);
      return {
```

```
13          strokes: strokes,
            shapes: shapeStrokes,
15          text: textStrokes
        };
17  }
```

These different strokes are saved within an array that is fed to the MyScript recognition engine together with a call-back function, which describes what needs to happen with the data once it is received, the API key and the recognition URL.

```
1   recognitionCall = function(strokes, apiKey, url, callback) {
        if (!url) url = "https://cloud.myscript.com/api/myscript/v2
    .0/analyzer/doSimpleRecognition.json";
3       var jsonPost = {
            "parameter": {
5               "hwrParameter": {
                    /** Language is a mandatory parameter. */
7                   "language": "nl_NL"
                }
9           },
            "components": strokes
11      };
        /** Send data to POST. Give your API key as supplied on
    registration, or the
13       * server will not recognize you as a valid user. */
        var data = {
15          "apiKey": apiKey,
            "analyzerInput": JSON.stringify(jsonPost)
17      };
        /** Post request.  Careful! If there are no candidates, the
    sample may crash. */
19      $.post(url, data, function(jsonResult) {
            callback(jsonResult);
21      }, "json").error(function(XMLHttpRequest, textStatus) {});
    };
```

5.3 Conversion to ORM Objects

This step was the most challenging part of the work. In this step, the different recognized components must be combined to create the elements of an ORM scheme. As the available components are shapes, groups, and text, a grouping mechanism based on these components had to be created.

5.3.1 Restricting the Notation

Because the MyScript recognition is still not optimal (at time of writing), we decided to restrict the ORM notation to come to better recognized models. As the MyScript software can have some difficulties to distinguish an ellipse from a circle, a rectangle from any other kind of quadrangle (rhombus, parallelogram, square or trapezoids) and does not recognize composed notations, we decided to only use ellipses, rectangles, lines and arrows as shapes. In addition, we restricted the use of composite notations. In Figure 5.4 the notations that are not supported are listed.

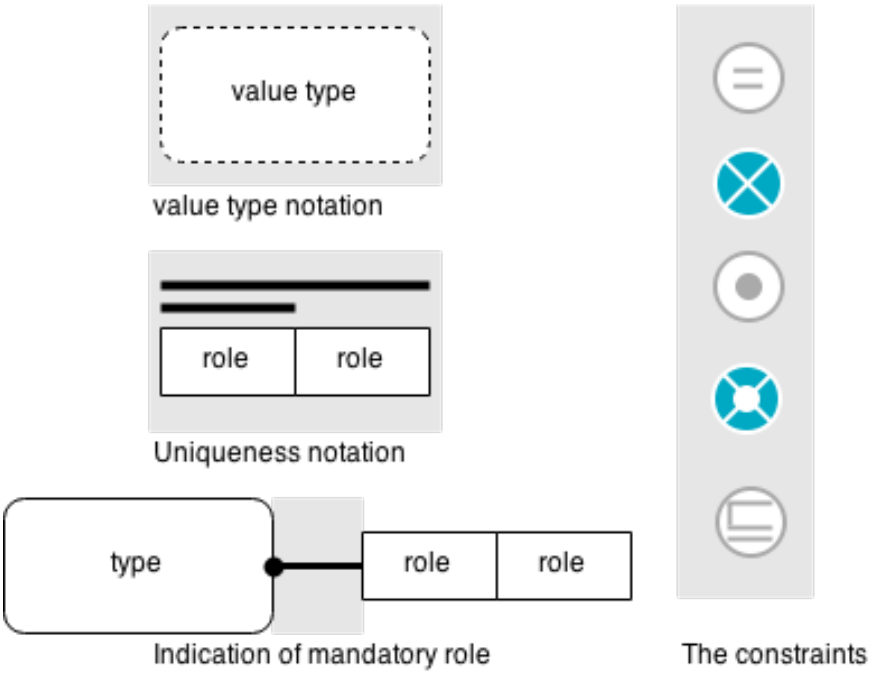


Figure 5.4: Notations that can not be detected by the MyScript recognizer

Due to the restriction on the used shapes we decided to base the distinction between Object Types and Value Types, the attribution of uniqueness and role requirement, and the recognition of constraints on text. Therefore, the restricted notation requires the text of a Value Type and the text of a role to contain one or more distinctive characters, as well as the constraint notation to be textual as shown in Figure 5.5. We opted to use the same shape (ellipse) for both Object Types and Value types but to enclose the name of the Value Type between '( )' brackets. We require the text of a role to contain a '@' character if it is not a unary role, and a '!' character if it

is a mandatory role. Constraints are indicated with an abbreviation of their meaning ('ex' for exclusion, 'ior' for inclusive or, 'eor' for exclusive or, 'eq' for equality and 'sub' for the subset constraint). Both capital or/and small letters can be used for the constraints.

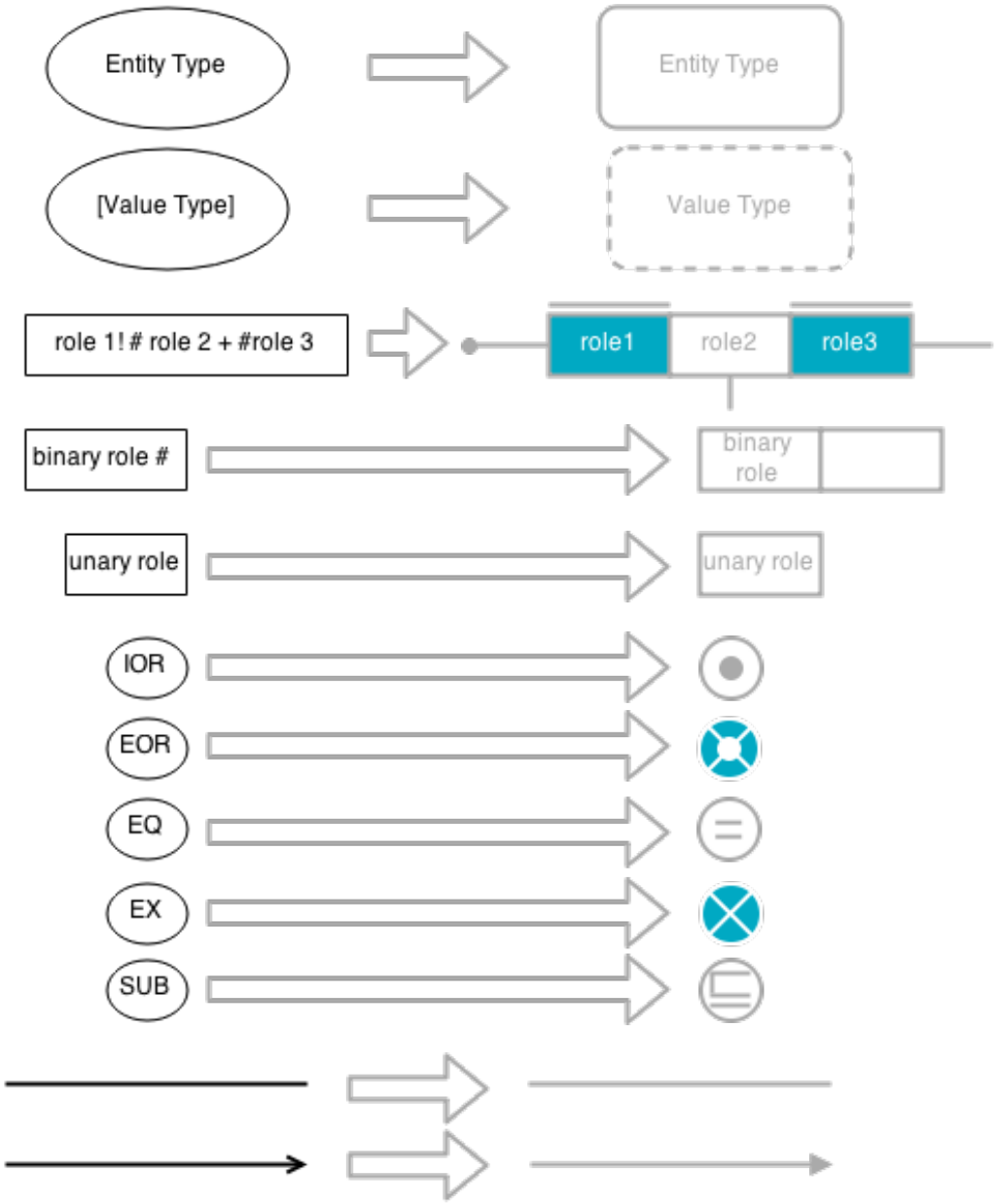


Figure 5.5: Restricted notations and their ORM2 equivalents

The last restriction is one that affects both shape and text. Because roles in the original notations are sometimes recognized as cells and sometimes the separation between the roles is recognized as a part of the names of the roles (i.e. the pipe symbol "|" is sometimes recognized as the number 1 or as the letter l) the option of representing the roles as individual (read disconnected) rectangles was considered. However this raised a problem for grouping them (it was not always clear when the disconnected roles had to be grouped). Therefore we opted to apply a textual solution to represent different roles by separating the different roles by a distinguishable character, in this case the '#' symbol. The Figure 5.5 shows an overview of the new notations used (left hand side) and their equivalent in original ORM 2 notation (right hand side).

Some last limitations have to be mentioned as well. All textual elements have to be written horizontally as the MyScript recognition engine does not recognize vertically written text. Roles also need to be placed horizontally; this is due to the algorithm that will create the bounding boxes of the individual roles to enable the connections to be between the Object Types and the individual roles and not between the Object Types and role group.

5.3.2 The Identification Algorithm

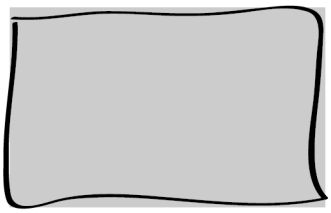
This part of the conversion relies mainly on the restrictions introduced for the notations. This process is done by a waterfall like structure of conditions describing the unique properties of the different objects in an ORM model. Due to the fact that there are only two shapes (ellipses and quadrangles), a line, and an arrow, the distinction between the shapes can be used as a condition/property of an object. The other conditions/properties have to be based on textual properties.

This process is done in 3 steps:

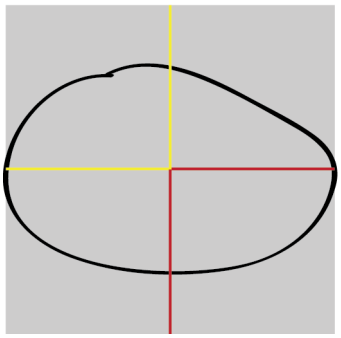
1. Iterating over the groups with the type "TEXT_INSIDE_GRAPHICS", these contain the references to the shapes which contain text/paragraphs which are basically our objects.

2. Decide what type of object it is based on the shape and text. This decision process is depicted in Figure 5.7.

3. Based on the type the different objects are created. This step includes defining a boundary box for the objects. This box delimits the area of the shape and is depicted as the grey areas in Figure 5.6 . These limits are necessary to create the connections between objects.



The bounding box of a rectangle is definde
by the corners of the quadrangles. The result is a
rectangular area with the coordinates:
(minimal x of all corners , minimal y of all corners)
(minimal x of all corners , maximal y of all corners)
(maximal x of all corners , maximal y of all corners)
(maximal x of all corners , minimal y of all corners)



The bounding box for a elliptical shape is defined
by it's center and maximal radius.
This is a square with coordinates:
(center x -max radius , center y - max radius)
(center x - max radius,center y + max radius)
(center x +max radius , center y - max radius)
(center x + max radius,center y + max radius)

Figure 5.6: Bounding box for ellipse and quadrangles

```
function createObjects() {
    for (var a = 0; a < groups.length; a++) {
        var group = groups[a]
        var textAndShape = getTextAndShape(group); //get the
shape and the text from the group
        if (textAndShape != undefined) {
            var text = textAndShape.theText;
            var shape = textAndShape.theShape;
            var kind = defineObjectsKind(shape, text); //
define the kind of object this makes
            makeTheObjects(kind, shape, text); //create the
objects
        }
    }
```

The most difficult part in step 3 is to define the bounding boxes for non-unary roles. Because of the notation used for such roles (one rectangle encompassing different roles which are divided by a "#" symbol), the limits of one role have to be determined.

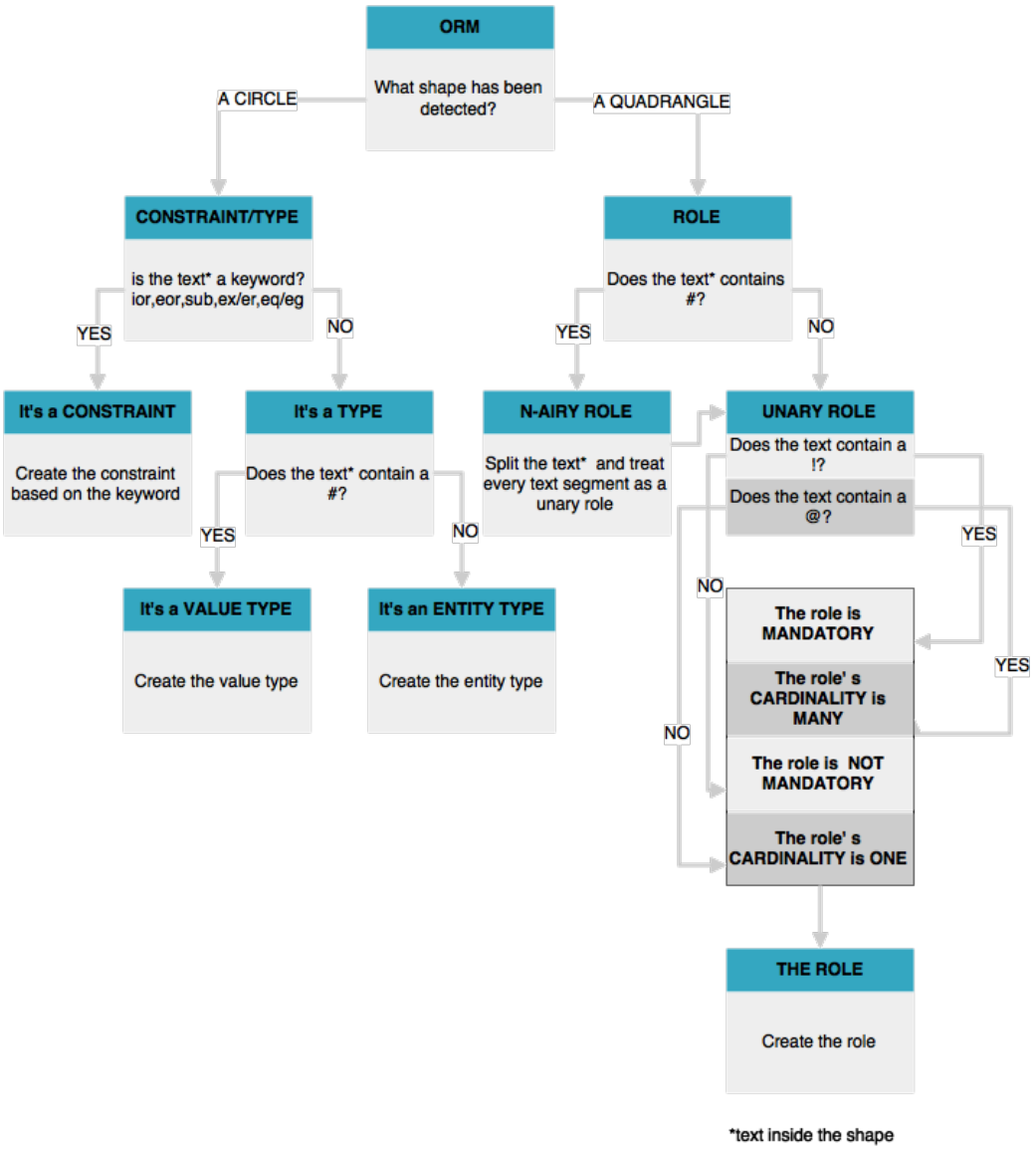


Figure 5.7: Flowchart for determination of object kind

One option was to delimit the role by finding the coordinates of the "#" symbol. However, this was not possible as we cannot count on the MyScript recognizer to give us the coordinates of the recognized symbols. Therefore we decided to create bounding boxes of the same length (total length of the box divided by the number of roles). Note that this method is not really accurate as some roles contain more text and are therefore longer but it enables one

to create connections by drawing the connection on the designated area of a role. Figure 5.8 depicts the difference between the actual role boundaries and the attributed boundaries. Note that a user has to envision the attributed boundaries to make the correct connections.

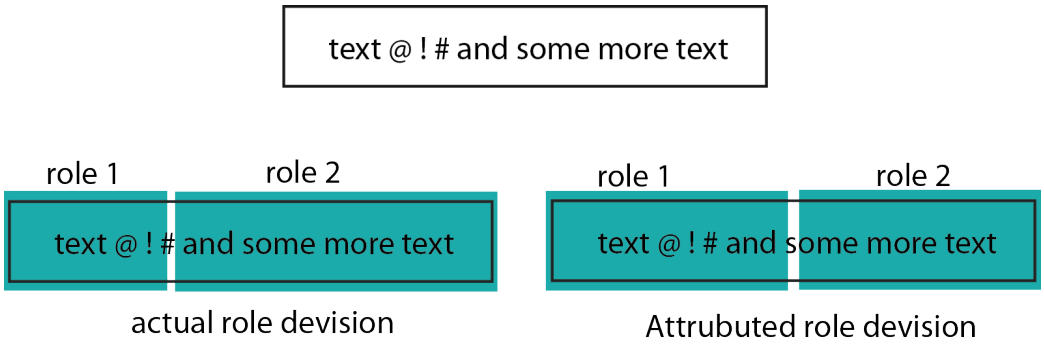


Figure 5.8: The difference between actual boundaries and attributed boundaries

Explored Alternatives

In an earlier version of the implementation we experimented with other ways to distinguish the different elements of the model. The initial idea was to distinguish roles from objects through their textual elements. Objects would contain text and roles would contain numbers, this would make it easy for the user to add the role names afterwards outside the role by writing down the number of the role followed by the roleś name. The number inside the role would then be mapped to the corresponding textual element. The problem with this approach, however was that the vertices of a role box sometimes were recognized as numbers, creating false identifications of a role. The initial idea, as well as why this didn't work, is illustrated in Figure 5.9.

We also explored a way to distinguish a normal connection and a mandatory connection. We tried to distinguish it by verifying if a circular object was attached to the connection line. The problem with this was that the circular object attached to the connection could be a constraint so we changed the notation to a triangle to indicate that the relationship is mandatory. This idea also did not work out as the MyScript recognizer identified the line connected to the triangle as a poly-line instead of a line and a triangle.

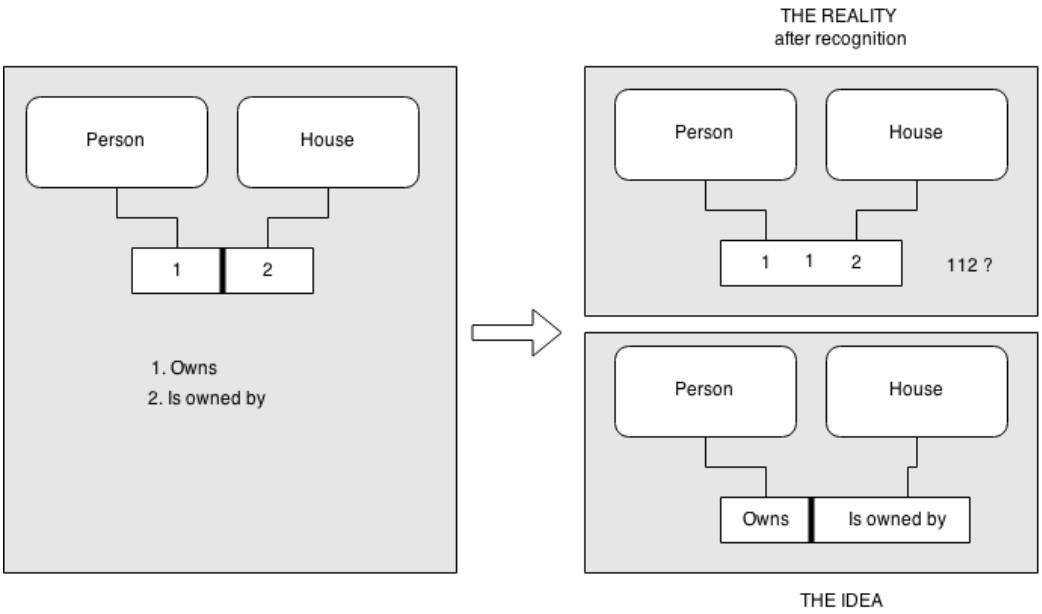


Figure 5.9: The initial idea and why it did not work out

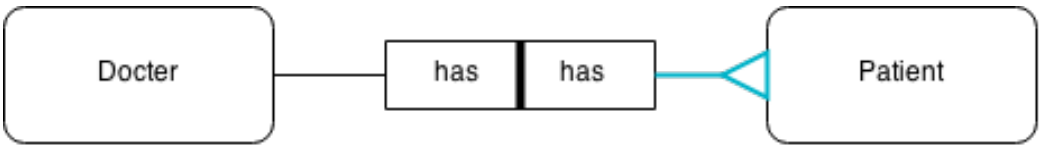


Figure 5.10: Experience with the notation for a mandatory relation

5.3.3 The Connection Algorithm

To connect two objects we use the bounding boxes of the objects. We select all the lines and arrows and see whether their begin or end coordinate lie within the bounding box of an object. Once both begin and end coordinate have been matched to an object the connection between the objects can be made.

For the distinction between a line and an arrow, we can rely on the MyScript recognition. However, to be sure we should test whether the line is connected to two Object Types (inheritance relation) or an Object Type and a role (simple connection). The only difficulty there is that if an arrow is misinterpreted as a line by the MyScript recognizer there is no way to know the direction of the arrow, however we can notify the user of a possible mistake in the model by changing the color of this connection in the resulting

diagram.

5.4 Extraction to XML Format

As it should be possible to edit and share the drawn model digitally, the model should be exported to a XML format that can be opened in a model editor. We opted to make it possible to visualize and edit the model in Draw.io. The first step was to analyze draw.io's XML format, as no definition of this format was available. Therefore different models were drawn using the draw.io tool, and afterwards saved to their XML format. The different models were analyzed and the different components were extracted.

The draw.io XML format is based on xGraph. Objects are independent, linked by referring to the id of the source and the target and grouped by creating a group and using this group id as a parent. Other elements like decorations and colors are given as layout attributes.

Creating the XML Document

To enable the conversion to XML, the shapes are first mapped onto ORM objects and their connections. Each ORM Object has a function *addToXML*, which will create the XML version of the objects.

Once the objects have been created they are saved in an array. To create the XML file for draw.io it is only necessary to iterate over this array and apply the addToXML function to each object. This is done by the *encloseXML* function which will inject the XML description of the different objects between the start and end tags of the XML document.

```
function encloseXML(objects, filename) {
    var doc = new XMLWriter('UTF-8', '1.0');
    doc.writeStartDocument(false);
    // create root doc
    doc.writeStartElement("mxGraphModel");
    doc.writeAttributeString("dx", "1");
    doc.writeAttributeString("dy", "461");
    doc.writeAttributeString("grid", "1");
    doc.writeAttributeString("gridSize", "10");
    doc.writeAttributeString("guides", "1");
    doc.writeAttributeString("tooltips", "1");
    doc.writeAttributeString("connect", "1");
    doc.writeAttributeString("fold", "1");
    doc.writeAttributeString("page", "1");
    doc.writeAttributeString("pageScale", "1");
```

```
        doc.writeAttributeString("pageWidth", "826");
17      doc.writeAttributeString("pageHeight", "1169");
        doc.writeAttributeString("style", "default-style2");
19      doc.writeAttributeString("math", "0");
        doc.writeStartElement("root");
21      doc.writeStartElement("mxCell");
        doc.writeAttributeString("id", "0");
23      doc.writeEndElement();
        doc.writeStartElement("mxCell");
25      doc.writeAttributeString("id", "1");
        doc.writeAttributeString("parent", "0");
27      doc.writeEndElement();
        //HERE ALL THE OBJECT'S XML CODE IS INJECTED
29      for (var i = 0; i < objects.length; i++) {
            objects[i].addXML(doc);
31      };
        //TILL HERE. FILE IS CLOSED AND FLUSHED HERE
33      doc.writeEndElement();
        doc.writeEndElement();
35      doc.writeEndDocument();
        console.log(doc.flush());
37      sendData(doc.flush());
}
```

Note, however that we adjusted the visual ORM2 notation (Moody, 2009). The mandatory constraint is translated visually by giving a color to the required role, instead of the original notation with a dot on the side of the mandatory relation. We made this change because it would still provide the information about the mandatory-ness of a role even if the connection itself has not been recognized and a color accent would attract the attention (Spence, 2001) of the user to the roles that are mandatory.

Some other changes we made to the final notation is the notation for the arity of a role. Instead of indication it by drawing a line above the role we underline the Role Name of the unary kind, so we can save space. We also decided to attribute Objects a rounded rectangle of the same size to improve the scalability of the model (El Dammagh & De Troyer, 2011).

The Draw2Model interface is shown in Figure 5.11.

5.5 The Integrated Process: Draw2Model

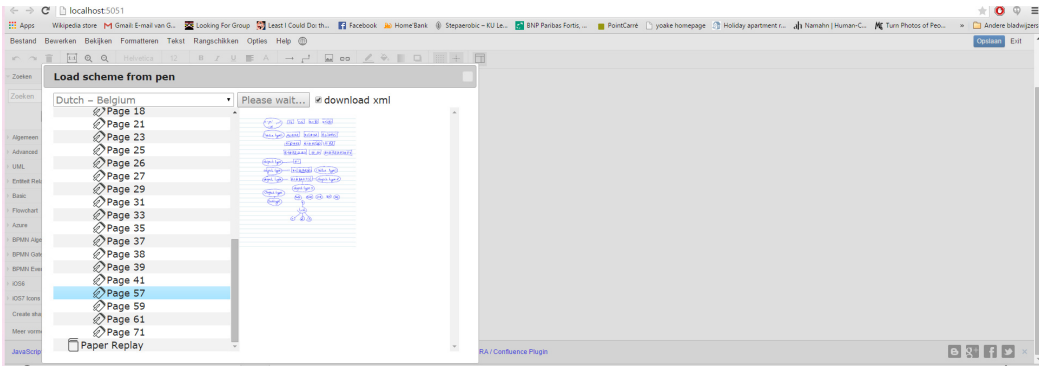


Figure 5.11: The Draw2Model interface

6 Evaluation

This chapter describes the evaluation of Draw2Model and its results. We performed this evaluation to investigate the correctness of the recognition/-conversion. Note that we didn't perform a usability study, as no formal usability testing method was used. Instead, we measured some performance aspects and correctness, and asked some questions to the subjects involved in the evaluation in order to be able to improve the application. After the description of the evaluation and its results we draw conclusions and discuss the elements that might have influenced the recognition. We performed the following three tests for measuring the performance and accuracy of the application. In the first two test no users were involved.

For the first test, we measure the processing speed of the different steps involved for the same schema drawn by different users. We processed the same model multiple times to get a good average of the processing speed. We also investigated the impact of the notation to see to which degree the notation restriction affected the expressiveness of a model.

For the second test, we drew the different ORM elements and connections in different ways and under different circumstances. This should give us insight into the elements that influence the accuracy of the whole recognition process. We did this by analyzing the deviations in each step of the pro-

cess (capturing the ink, MyScript recognition, and the grouping algorithm). For this test, we varied the arrow notation; the way of drawing connections: touching the shapes, not touching the shapes, and going through the shapes; and the way of drawing rectangles: four individual strokes, one stroke, and two times two strokes. We also alternated the lighting conditions and writing surface to test the influence of external factors.

For the last test, the same model was drawn several times but by different users. This test involved 15 subjects age ranging from 16 till 82 with different professional backgrounds and sex (see Figure 6.2). The subjects were asked to recreate a given model using the Echo smartpen. They were given the guidelines that the different objects could be placed at different positions on the page but that the shapes had to stay the same. They were made aware of the fact that connection lines should touch the objects they connected and that they could use their own handwriting for the textual elements. After finishing the drawing, they were asked to fill in a small questionnaire to provide us with some background information about the subject, and to give the subjects the opportunity to give us some feedback about the adapted notation used and the smartpen. The questionnaire that was used for the testing.

This last test enables us to formulate a conclusion about the overall transformation process for multiple users, each with their own personal handwriting. We analyzed the overall recognition score and investigated which elements of the model were best recognized and which were not.

6.1 Correctness Measurement

In order to measure the correctness of the recognized schema the following scoring system was used. Each element of the original schema was attributed one point as shown in Figure 6.1, so we start with a given number of points. For each missing element in the recognized schema one point was subtracted. For each deviated object (mistake in the text or a arrow that has been interpreted as a line and vice versa) half a point was subtracted.

6.2 Performance Test

For this test, we processed the same model multiple times to get a good average of the processing speed. The overall time needed to transfer the data,

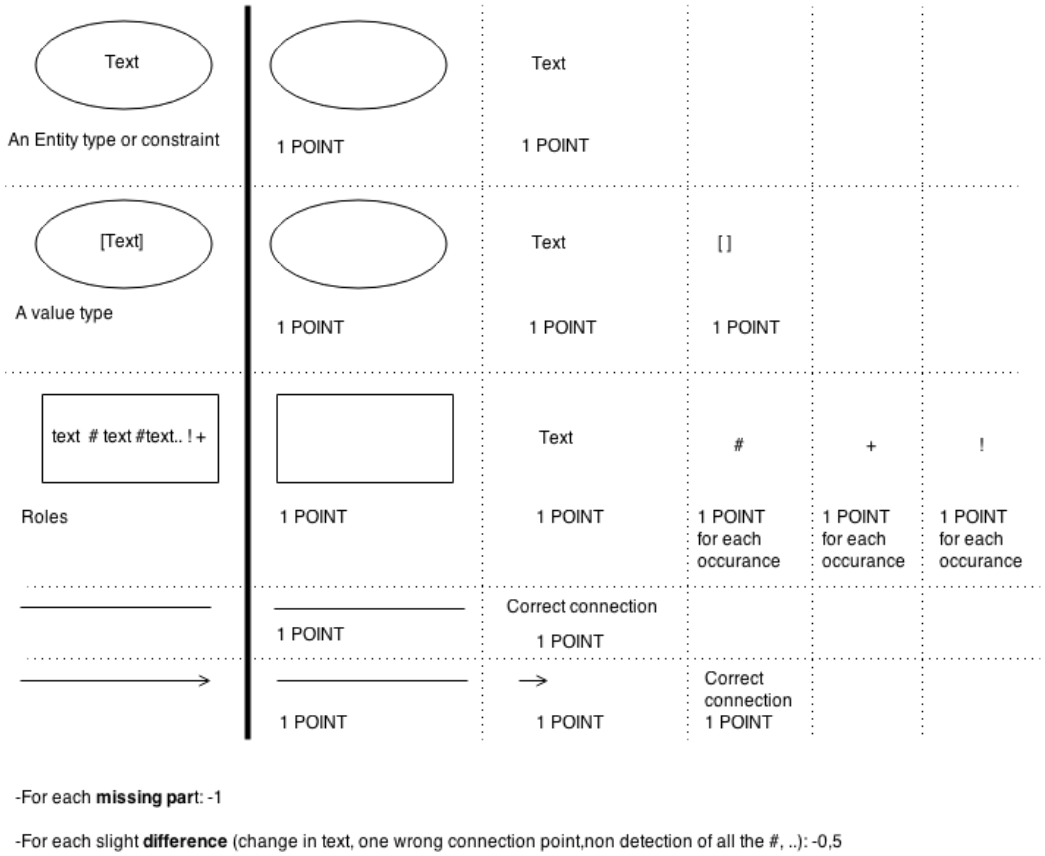
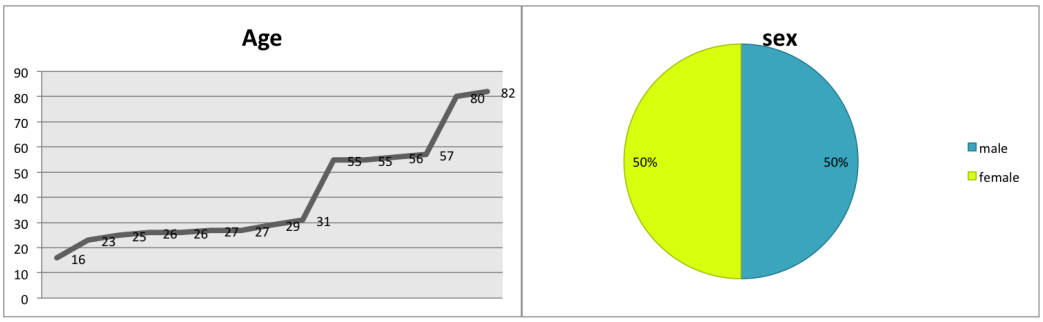


Figure 6.1: Scoring system



(a) Age of the Test Users (b) Sex of the Test Users

Figure 6.2: Demography of the test users

the MyScript recognition, grouping the elements and generating the XML is around 19 seconds (as can be seen in Figure 6.3) for an average model. It is

however difficult to measure what exactly influences the processing speed as there are a lot of variables involved. The number of coordinates and strokes will influence the data transfer and MyScript recognition, and the number of recognized elements and connections will influence the grouping process and conversion to XML. It is however not possible to vary the number of stroke coordinates without influencing the recognition, in order to investigate the influence of one single component.

However, the data transfer seems to take most of the time (around 14,5 seconds) as can be seen in Figure 6.4a followed by the MyScript recognition (4,15 seconds) as seen in Figure 6.4b. The speed of grouping the elements seems to depend mainly on the number of groups MyScript detects and the number of shapes it detects as the algorithm loops over all the groups to create the groups and then again over all the shapes, to find the lines, and over the created objects' bounding boxes. Converting the objects into an XML format is the least time consuming step of the process and seems to depend on the number of objects that were detected.

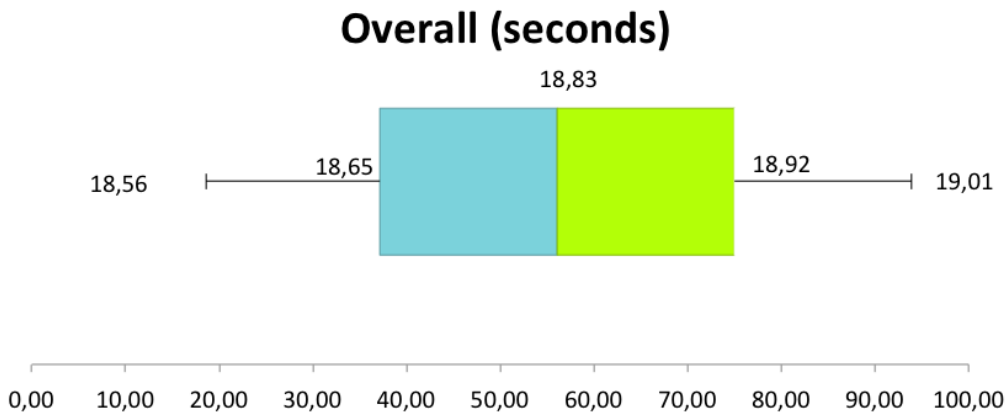
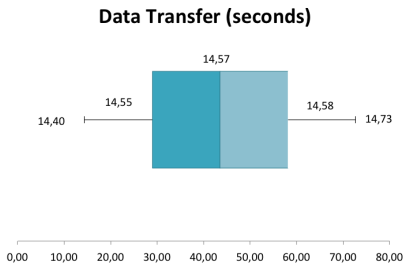


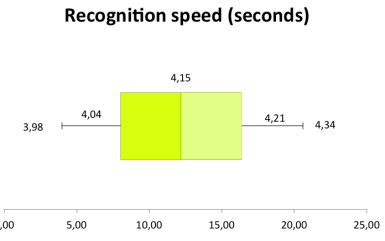
Figure 6.3: BoxPlot of the application speed

6.2.1 Influence of the Different Components

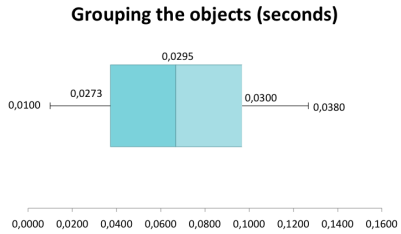
Unfortunately, it is difficult to correctly examine the influence of the different factors, such as pen hold and handwriting, as for this it is necessary to keep



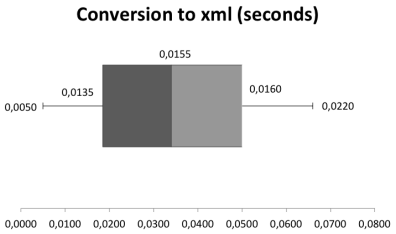
(a) Boxplot of the Data Transfer Speed



(b) Boxplot of the Recognition Speed



(c) Boxplot of the Grouping Algorithm Speed



(d) Boxplot of the Transformation to XML Speed

Figure 6.4: Speed performance of the application (for an average schema)

all variables constant except one to investigate its influence on the transformation process. This was not possible as the pen itself is variable and the handwriting is also variable, even for the same user.

We noted that the Echo smartpen was mainly responsible for badly recognized connections, as the lines are often cut short, and missing objects, as unclosed shapes are not detected as shapes by the MyScript recognizer. The MyScript recognizer on the other hand is responsible for the creation of wrongful connections and bad text recognition. Overall the text recognition is rather good if the user has a rounded separated handwriting, it is more error prone when the user uses calligraphy. We discuss these aspects in more detail in the next sub sections. Note that the Draw2Model application itself currently lacks the ability to correct mistakes made by the MyScript recognizer certainly when it comes to the creation of connections. To a certain extent this would be possible if the application would use the semantics of ORM.

The Stroke Capturing

After visualizing different schemas with the Echo desktop software we were able to see that the Echo smartpens recording of the strokes is not optimal. We noticed that when a user writes/draws too fast the coordinates that the pen records are far from each other and subtle nuances between letters are lost as most letters resemble the cuneiform script. We were able to deduce however that pressure on the pen is a very important factor. Writing with too many pressure can lock the ink cartridge to the pressure sensor, in which case all letters and shapes are connected, whilst not enough pressure may result in not recording the strokes. The surface on which one writes is also an important factor. When a user writes on a soft surface the paper will dent a bit and the dotted pattern will not be recognized that well. It may also result in not enough pressure on the pen to enable the pressure sensor for recording.

We also investigated the influence of lighting conditions and wrote with the pen upside down. This did not seem to influence the capturing of the strokes except that the coordinates are a bit shifted in the last case.

MyScript Recognizer

The MyScript recognizer works well when the strokes captured by the Echo smartpen are clear. It however has a huge deficiency when it comes to line recognition. In some cases the MyScript recognizer detected more lines then created and in other cases lines disappeared because the were detected as poly-lines or arc of an ellipse.

6.3 The Transformation Process

As shown in Figure 6.5 the end result is far from good. The median recognition score is 3,4/10 and the best score is 5,7/10. We had hoped that the resulting model would be a better reflection of the original model. We observed that the greatest problem in the transformation process is in the detection of lines/connections.

The highest score was obtained by a subject which was left-handed. We did not have enough left-handed subjects to investigate whether the difference between left-handed and right-handed is important. It could be interesting to investigate this further.

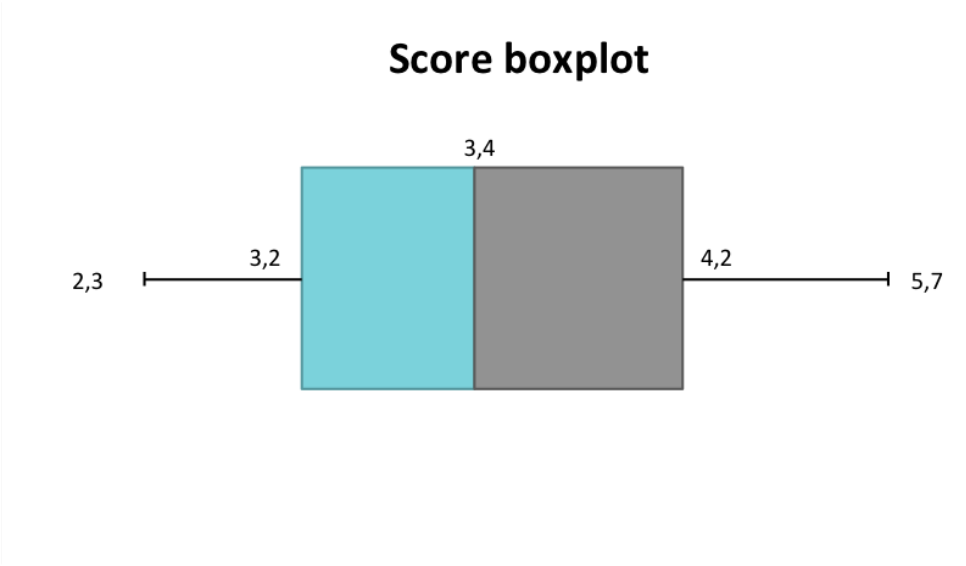
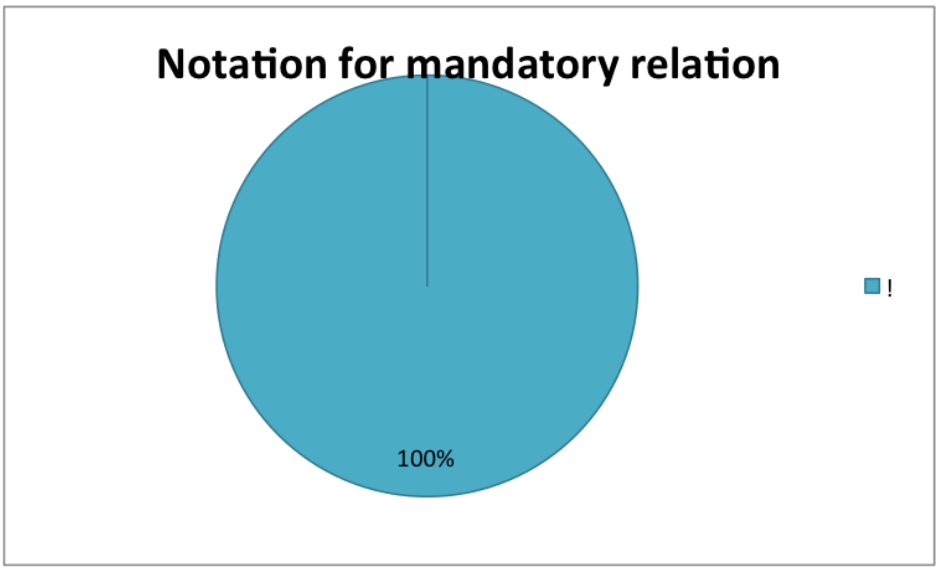


Figure 6.5: An overview of the final scores

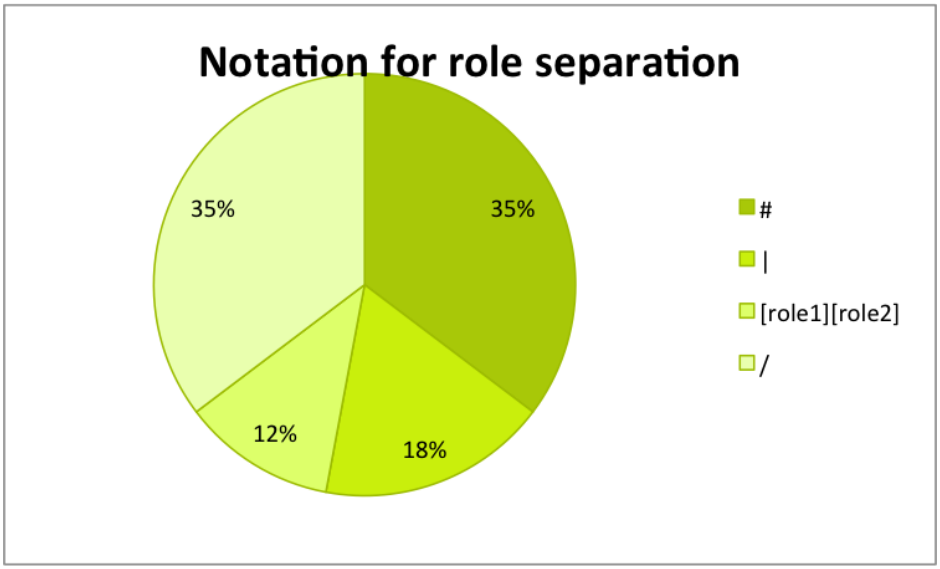
The Notation

In the last test, after the subjects recreated the schema, the subject received a brief explanation about ORM and its notation. We explained that in our notation the exclamation mark means that the relation is mandatory, the hash symbol indicates the parting between the roles, and the '' symbolizes a many-arity for roles. We then proceeded by asking their opinion about these notations and whether they had suggestions for alternative notations.

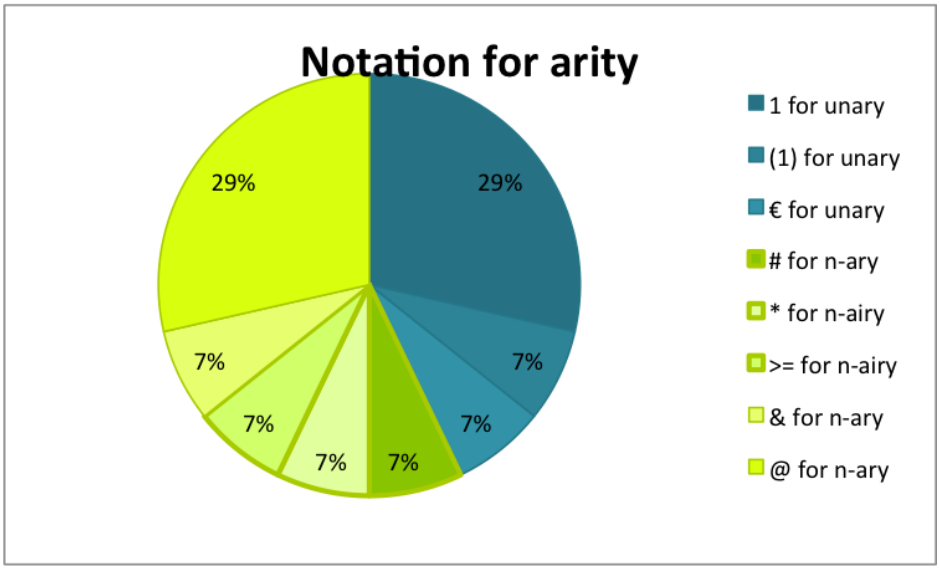
The subjects unanimously agreed that the exclamation mark was a good notation to indicate that a role is mandatory as it commands attention and emits a sense of commandment.
For the hash notation the opinions were more divergent. Some subjects found it a good notation but most would prefer another notation.
For the arity notation most users would have preferred to indicate that a role is identifying(unary) instead of indicating that a role is n-ary, as they believe an identifying role(unary) is less common and more important than an n-ary. In Figure 6.6 the distribution among the suggested notations and their alternatives are depicted.



(a) The Mandatory Constraint



(b) The Role Division



(c) The Arity

Figure 6.6: Users opinion about the notations

User's Opinion and Observations

The subjects gave the pen an average score of 3,5/5 for the overall idea and the pen itself. This moderate score was mainly due to the design of the pen. Most of the subjects had some comments about the ergonomics of the pen. The pen was too heavy, too large and too slick, this combined with the pressure that needs to be applied to the pen made writing with the pen rather tiresome. As a result of the bad design of the pen some users wrote with the pen upside down, obstructed the camera with their fingers, or did not exert enough pressure on the pen. Ergonomics aside, some subjects saw the potential of using a smartpen for modeling, but some still preferred using a tablet or PC directly for the modeling. It were mainly the elder subjects and teenagers who preferred to still use pen and paper as they are not that acquainted with computers and can write faster on paper then with a keyboard. For the most positive aspect about using a smartpen, most users said that it was the fact that they had a paper copy (so they can not lose changes due to a computer failure and have paper proof for important documents). As negative aspects they often listed the ergonomics and the fact that errors are also permanently recorded (they can not use tip-ex on the paper, and crossed out errors are still recorded digitally).

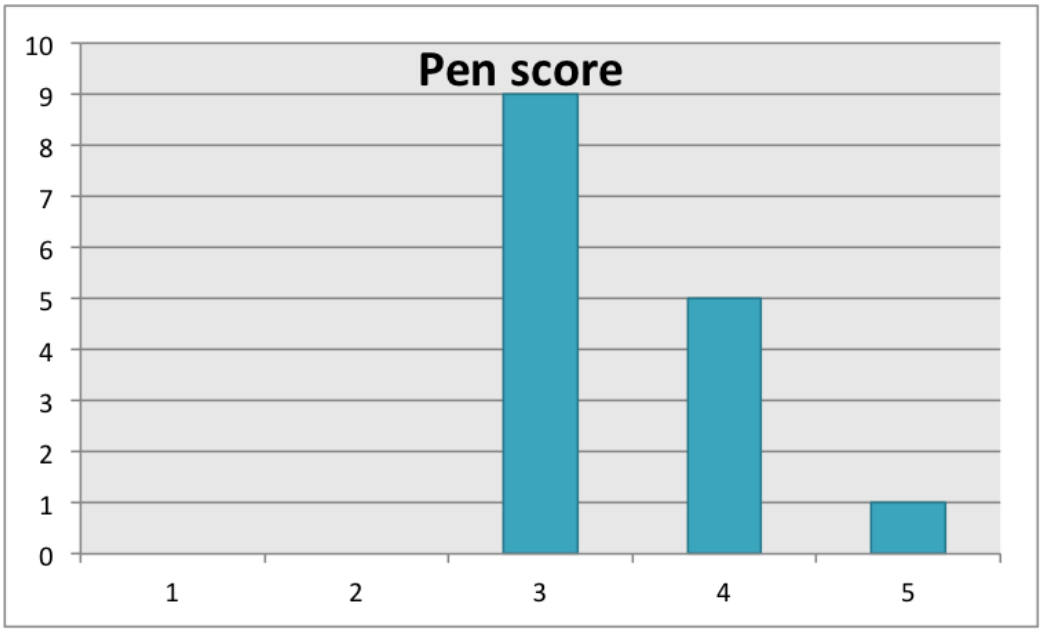


Figure 6.7: The user score for the pen (ergonomics + possibilities of a smartpen)

7 Conclusions & Future Work

In this last chapter we reflect on the results: the final software, the followed development process, and the soft and hardware used. Some considerations about the different components are discussed and suggestions for future work are given.

7.1 Echo smartpen

Whilst the Echo smartpen seems to be a promising technology there still are some concerns. First of all, the pen is not that comfortable ergonomic wise. As the pen contains a camera to capture the strokes the pen is rather thick and is not easy to hold. The user also needs to make sure not to block the view of the camera with his/her fingers, so the user cannot hold the pen by its tip but has to hold the pen rather high. This grip may have an impact on the quality of the writing. Because the stroke capturing is only enabled when the pressure sensor detects pressure, the user also has to apply enough pressure to activate the detection, which can be tiresome after a while. These characteristics may induce the need for a learning period to become acquainted with the pen.

A second consideration is the lack of support by LiveScribe for the development of new applications for this pen. While this pen could have a

lot of application domains, by means of customized penlets, Livescribe unfortunately discontinued their development kit therefore limiting the use of this device. However, the available development kit itself was rather neat. Developers just have to design their own paper, with designated active areas and describe the matching event handlers in the penlet itself. The creation of such customized penlets and access to the data opens a range of possibilities from hospital applications (i.e. preventing double input of patient information by extracting the patient information and sending it directly to the digitized dossiers) to learning aid (e.g. the Smart Study application described in Chapter 2). Understanding the code is rather straightforward as most of the code can be auto generated and the names are self-explanatory. Unfortunately, there is not a lot of information about the development kit regarding its restrictions (e.g. the SDK does not support a full version of Java and the local memory can only store files with an extension of three characters) or about how to access the data.

The last consideration is the accuracy of the pen. The accuracy is rather good but as coordinates are not continuously recorded (there is a small delay in order to save the coordinates and detect them) the resulting ink file consists of polygons. When a user draws an elliptical figure too fast this results in a polygon that does not approximate the elliptical shape.

7.2 MyScript

The MyScript recognition engine seems to use a similar approach to recognition as described in the Tahuti paper. It seems to sort the different ink coordinates and look at the overall ink instead of looking at the individual strokes themselves. This means that the order in which strokes are drawn are not important. This has the advantage that corrections to text and shapes can be done (e.g. adding a character in between text adding dots to the after writing everything and so on). The downside of this is that it does not seem to use the context of when strokes are drawn to facilitate recognition, as described in the LADDER paper. They however seem to use nearness context to decide whether it is a shape or a textual element (i.e. when placed near other textual elements it probably is a textual element or when containing textual elements it probably is a shape).

Their recognition engine also seems to be a learning engine as when sending the same data set (set of strokes) multiple times the first recognition will deviate from the second, third or fourth recognition. This however means

that sometimes a correct recognition can become an erroneous recognition. This leads us to believe that the learning algorithm they use is based on the number of occurrences of a certain shape or textual element to factor in their probability.

7.2.1 Service Quality

Overall MyScript is a good product. They provide multiple recognition engines based on what is required (shape, text, math, music or multiple recognition), ample documentation and a free trial of 90 days. However their customer support is lacking efficiency (they did not respond at all to our questions) and their cloud service was often unreachable due to maintenance. Also the pricing is rather high and they do not seem to offer multiple recognition in their E-commerce offer (MyScript, n.d.).

7.2.2 Recognition Accuracy

Overall the MyScript recognition engine worked better than we expected. The handwriting recognition seems to work well for text, both for the English and Dutch language. For multiple handwritings, however, it seems to work better when the userś handwriting is:

- Rounded rather than elongated
- Big
- Consists of separated letters/not written in Calligraphy letters

The MyScript group recognition also works well as long as the Echo smart pen has correctly recorded the strokes (shapes are closed and rounded sides are round not linear due to lack of recorded coordinates). The biggest problem with the MyScript recognition accuracy seems to lie in their shape recognition approach. Shapes which resemble each other (i.e. circles and ellipses, all the different quadrangles, lines and poly-lines, ...) are often mistaken for one other. After running several tests we concluded that mistakes in the recognition of the objects were mainly due to the inaccuracy of the Echo smartpen, which does not record the pen's position when the user lessens the pressure on the pen tip. This was often the case when users had to draw elliptical shapes, they tend to lessen the pressure on the pen resulting in a non closed ellipse as can be seen in Figure 7.1.

The biggest problems with the MyScript recognition are in the recognition of connections, as the recognition engine seems to recognize lines that

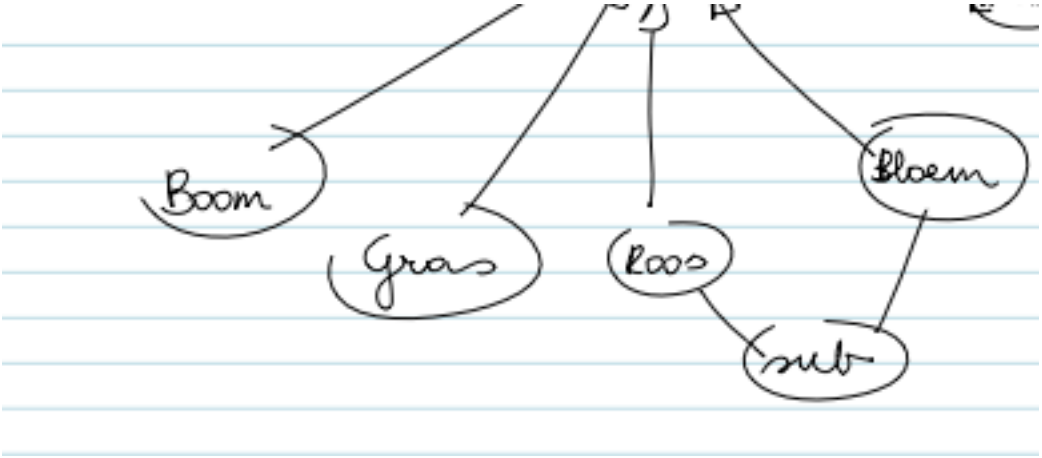


Figure 7.1: Echo smartpen stroke detection of an ellipse

intersect with other lines/shapes as poly-lines and sometimes recognizes two lines, which lay in each others extension, as one single line.

The Text recognition, however, works very well, as its recognition relies on probability and context. The shape recognition also works well but the right answer is often only the second or third choice and the line recognition could be better. It is also a pity that the shape recognition does not recognize filled in shapes, as the MyScript recognizer will perceive the filling as a tangle of strokes instead, as it is not capable to perceive the stokes as a whole like the human brain can.

7.3 Drawing Restrictions

Overall enabling the recognition of an ORM schema required a lot of restrictions. Users cannot rotate the page, have to write horizontally, and exert enough pressure on the pen so that the strokes are recorded. We also noted that some handwritings are recognized more easily than others.

Notation wise the restrictions also have a big impact. Symbols could not be used for constraints and had to be replaced by textual elements. This is in conflict with the use of a standard notation and because constraints are added textually the components often are larger than when using the standard notations, making the models larger and less scalable. It also imposes the user to first write down all the textual elements before encompassing the text

with the shape to ensure that all text elements lie within the shapes. Users are also required to create neat models, which results in slower drawing and thus undermines the purpose of using pen and paper, i.e., being faster than a modeling tool.

7.4 Recognizing the Concepts

It was fairly easy to group the text with the appropriate shapes because MyScript provides group recognition. However, creating the connections between the different objects and roles and identifying the correct objects turned out to be more difficult. We tried different approaches but had to opt for the current one which is not at all accurate.

The wrongful attribution of connections and lack of connections are mainly due to the MyScript recognition engine and partially to the accuracy of the Echo smartpen. As the MyScript recognition engine seems to map the shapes from their database to the digital ink patterns, and not rely on context, lines that lie within each others extension are perceived as one, and lines that intersect with other lines are perceived as poly-lines. In Figure 7.2 an example of such wrongful recognition is given.

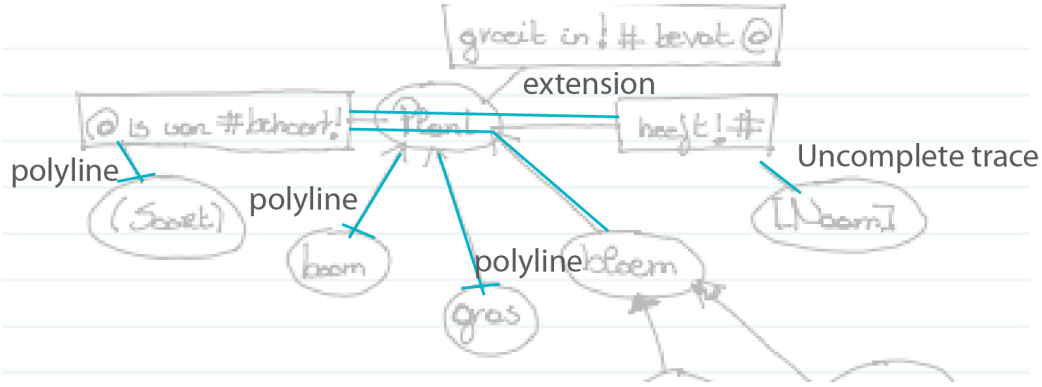


Figure 7.2: Wrongful detection of relations

Overall the textual recognition of the symbols for dividing the roles (#), attributing arity (@), and mandatory-ness (!), seems to work rather well. However, the user evaluations have shown that, while every subject understood the meaning of the '!' symbol, they would prefer another symbol for

the arity and role division.

7.5 Summary

It would have been nice that the whole process works impeccably, but unfortunately the quality of the recognized schemas is not high. This is mostly due to the different external resources that were used. The Echo smartpen for the stroke capturing and the MyScript recognizer for the recognition, are still not optimal.

To obtain schemas of high quality, the user is required to put enough pressure on the pen, draw the shapes clearly, and see that no lines intersect with the shapes. When these conditions are satisfied the program will recognize all the objects and constraints accurately. This, however, does not solve the problem. To solve this, an alternative way to create connections should be found.

One proposal would be to create a custom penlet that will record coordinate pairs created by double tap events. Then, a user could double tap in one object and then the other to indicate that those two objects should be connected. The object recognition would then still rely on the current Draw2Model code but mapping the paired coordinates to the objects in which they lie could attribute the connections. The type of connection (object type-role or sub-type) can be deduced with the help of the objects involved, if object types are connected it is an sub-type relation else it is an object type-role connection.

Unfortunately, we have to conclude that our approach is not a success story. We tried a lot of alternatives and figured out why they were not working and which alternatives may work if the recognition and pen accuracy were better. However we still believe in creating a bridge between paper models and the digital models. Most of our subjects also believed that it is easier and faster to write ideas down on paper. Many of them were enthusiastic about the future possibilities such a bridge could bring. We list some of them:

- Converting handwritten (sloppy) notes into colorful and clean MindMap syntheses as illustrated in Figure 7.3
- Converting handwritten patient information into the corresponding digital patient files

- Transferring fieldwork notations (written on paper because it can resist the rough circumstances of field work) directly into the database.

- Converting ORM or ER schemas from paper drawn concept into a concrete database.

7.6 Future Work

Because the end result of the conversion deviates a lot from the original model we had to conclude that this attempt in creating a bridge between a handwritten model and a digital modeling tool was not very successful. We could try to improve the end result but unless the recognition software and pen accuracy gets better, different work around would be needed. This would affect the usability and compromise the purpose of using pen and paper, i.e. being faster and more flexible than when using a computer tool.

In this chapter we have already described some ideas to improve the current tool. In the next sections, we propose some other further research.

7.6.1 Creating Models from Scanned Documents

The fact that our current approach relies on the Echo smartpen has some downsides:

- The user needs to own/buy an Echo smartpen

- The ergonomics of the Echo smartpen is not optimal.

- The ink of the Echo smartpen can not be removed.

A solution to these down sides would be to create an alternative to our program where the digital ink is obtained by scanning the paper document and converting this image into digital ink. A user could then use whatever pen or pencil he likes to write with on any kind of paper. This would also improve the recognition as a scanned document is more accurate then the recording of the pen strokes. A down side to this approach is that the user is required to scan the documents he wants to transform.

ICT

- Chapter 1

  * data are facts or events which are recorded in a certain way, so that processing is possible. They are the base of all information.

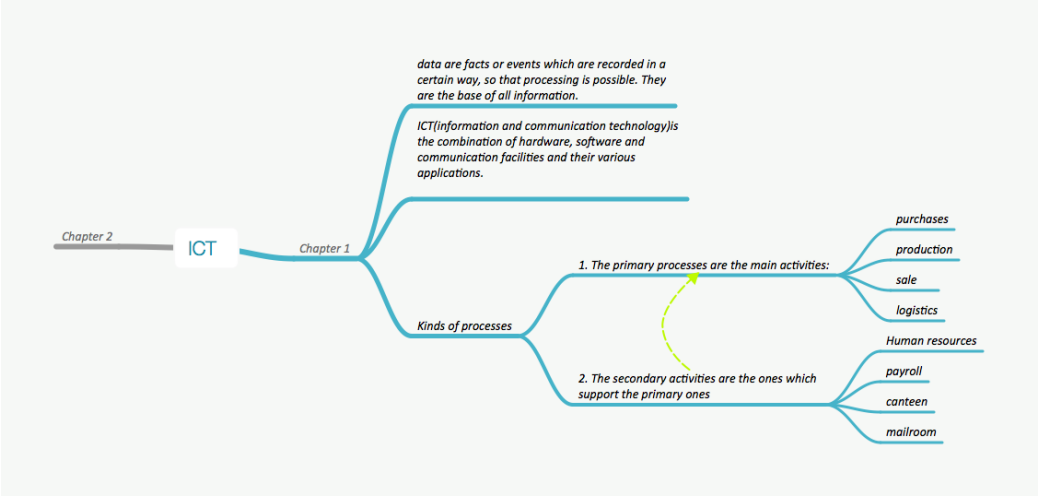
  * ICT(information and communication technology)is the combination of hardware, software and communication facilities and their various applications.

  * Kinds of processes

  1. The primary processes are the main activities:
     + purchases
     + production
     + sale
     + logistics
  2. The secondary activities are the ones which support the primary ones (1)
     + Human resources
     + payroll
     + canteen
     + mailroom

- Chapter 2

(a) The original synthesis document (created by hand)



(b) The same schema transformed into a MindMap

Figure 7.3: Transformation of a synthesis into a MindMap model

7.6.2 Improving Notations

Due to the imperfect recognition of the MyScript recognition engine an alternative notation for ORM2 had to be created. This notation is an experimental one aimed for distinguishing objects rather than for readability or ease of drawing. Further research on notations that can distinguish the different objects from each other while being universally understandable and intuitive has to be done. It could be noted that the current notations of ORM2 can also use some improvements to make it more universally understandable (notation for constraints and arity of roles amongst others). So, further research could focus on combining the two issues.

Direct Connection between Pen and Output

A customized penlet could be exploited to aid a modeler with the modeling. The tap interactions, display and audio features of the pen could then be used to improve the modeling experience. We provide some possibilities.

The pen interactions (on tap, on double-tap, and on area tap) can be explored for improving the recognition and for adding actions such as delete shape, clean page, or switch between text and shape. The pen display could be used to assist the modeler by providing error detection, e.g. display: "this shape is not a valid one, please indicate which shape you meant to draw" or "it is not possible to use this connection between these two types of objects". The audio feature can also be exploited for instance to give a short signal in case of an error or for recording on going discussions while making the model or to record modeling choices.

7.6.3 Other Types of Diagrams

In this thesis, the focus was on ORM models. However, it is also possible to consider other types of models, for instance Unified Modeling Language (UML) class diagrams and Mind Maps.

Because UML Class diagrams mainly consist of rectangles and text it should be possible to create a similar application for class diagrams. Note that a similar application for UML class diagrams would also require some adjustments to the notations, as MyScript does not recognize lines with small shapes connected to them (as for the aggregation and composite relation type), sub divisions of a class object (for attributes and methods), or when a text element should be grouped with a line (relation name).

Creating MindMaps with a similar tool is more feasible as Mind Maps are basically graphs connecting textual objects with each other. An ellipse could then encompass the textual objects, and the connection algorithm could be the same as the one used for the Draw2Model application for connection two Objects Types with each other.

7.6.4 Create an Online Editing Tool for the Resulting Diagrams

With the current application, the diagrams can be visualized and edited in draw.io, which is a general modeling tool. However, it would be nice to be able to open the diagram in an ORM specific modeling tool. This could be achieved by providing more export formats of the data so that it could be exported to a format fitting ORM specific modeling tools such as Visio or NORMA. Another alternative is to provide complementary custom made cloud based application for ORM modeling. In the research training accompanying this thesis some suggestions for such a cloud-based application are provided.

7.6.5 Create a better recognition algorithm

instead of using the MyScript recognition engine, can also explore the option of creating our own (simplified) recognition algorithm. As the targeted modeling language (ORM) is known, we could limit the resources to which the shapes should be matched, to the shapes used in the ORM-language. To further improve the recognition we could use hard and soft constraints (based on contextual information like time of creation and coordinates that are members of the same stroke).

References

Brandl, P., Haller, M., Oberngruber, J., & Schafleitner, C. (2008). Bridging the gap between real printouts and digital whiteboard. In S. Levialdi (Ed.), *Proceedings of the working conference on advanced visual interfaces* (pp. 31–38).

Chee, Y.-M., Froumentin, M., & Watt, S. M. (2006). Ink markup language (inkml). *W3C Working Draft*, *23*.

Chen, Q., Grundy, J., & Hosking, J. (2003). An e-whiteboard application to support early design-stage sketching of uml diagrams. In *Human centric computing languages and environments, 2003. proceedings. 2003 ieee symposium on* (pp. 219–226).

CMedia. (2015). *Inkling van wacom: Schetsen op papier, digitaal verfijnen.*

Dachselt, R., Frisch, M., & Decker, E. (2008). Enhancing uml sketch tools with digital pens and paper. In *Proceedings of the 4th acm symposium on software visualization* (pp. 203–204).

DigitaalSchrijven. (2011, June). *Digitaal schrijven, hoe werkt de anoto technology (digitale pen en grid).* Retrieved from `https://www.youtube.com/watch?v=As5IQK6JxJc`

Draw.io. (n.d.). *Draw.io.* Retrieved May 20 2015, from `https://www.draw.io/`

Dymetman, M., & Copperman, M. (1998). Intelligent paper. In *Electronic publishing, artistic imaging, and digital typography* (pp. 392–406). Springer.

El Dammagh, M., & De Troyer, O. (2011). Feature modeling tools: evaluation and lessons learned. In *Advances in conceptual modeling. recent developments and new directions* (pp. 120–129). Springer.

Evans, K., & Hallock, P. (2003). *Database modeling with microsoft visio for enterprise architects.* Morgan Kaufmann.

FiftyThree, I. (2015). *Think fast. think clearly.* Retrieved May 20 2015, from `http://www.fiftythree.com/think`

Halpin, T. (2001). Object role modeling: An overview. *white paper,(online at www. orm. net). gadamowmebulia*, *20*, 2007.

Halpin, T. (2005). Orm 2. In *On the move to meaningful internet systems 2005: Otm 2005 workshops* (pp. 676–687).

Halpin, T. (2006). Object-role modeling (orm/niam). In *Handbook on architectures of information systems* (pp. 81–103). Springer.

Halpin, T. (2009). Object-role modelin. *Encyclopedia of Database Systems*.

Hammond, T., & Davis, R. (2005). Ladder, a sketching language for user interface developers. *Computers & Graphics*, *29*(4), 518–532.

Hammond, T., & Davis, R. (2006). Tahuti: A geometrical sketch recognition system for uml class diagrams. In *Acm siggraph 2006 courses* (p. 25).

Inc, L. S. (2015). *Diagrams done right.* Retrieved May 20 2015, from `https://www.lucidchart.com/personahomepage`

Lee Garber, C. (2013). High-tech pen checks spelling while users write. *IEEE Computer Society*.

Livescribe. (2015). *Livescribe smartpens.* Retrieved May 20 2015, from `http://www.livescribe.com/en-us/smartpen/`

Livescribe, I. (2007-2009). *Getting started with the livescribe platform sdk (livescribe^{TM} platform sdk version 1.0.1).* Retrieved May 20 2015, from `http://www.spiska.sk/manuals/Getting_Started_Platform_SDK.pdf`

Livescribe, I. (2015a). *echo overview.*

Livescribe, I. (2015b). *Livescribe 3 smartpen.* Retrieved from `http://www.livescribe.com/nl/smartpen/ls3/`

Livescribe, I. (2015c). *Livescribe wifi smartpen overview.* Retrieved from `http://www.livescribe.com/nl/smartpen/wifi-smartpen/`

Matt. (2015, May). *Livescribe 4gb echo smartpen – redefine your life.* Retrieved 5 June 2015, from `http://selectroniks.com/livescribe-4gb-echo-smartpen-redefine-your-life`

Media, H. (2015). *The compromise pin.* Retrieved 1 June 2015, from `http://www.heise.de/ct/artikel/Der-Stift-Kompromiss-290368.html`

Meersman, R. (2000). *Conceptual schema design steps.* Retrieved from `http://starlab.vub.ac.be/website/files/Infosys%20Ch4.pdf`

Microsoft. (2015). *Belangrijkste functies van visio.* Retrieved 5 June 2015, from `https://products.office.com/nl-nl/Visio/microsoft-visio-2013-top-features-diagram-software`

Missfeldt, M. (2015). *Wacom intuos graphics tablet per m (medium).* Retrieved 1 June 2015, from `http://www.zeichnen-am-pc.de/grafiktabletts/`

Moody, D. L. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on*, *35*(6), 756–779.

MyScript. (n.d.). *Getting started with myscript ink cartridges.* Retrieved May 20 2015, from `https://dev.myscript.com/getting-started-with-myscript-ink-cartridges/`

MyScript. (2015a). *Analyzer recognition.* Retrieved from `http://doc.myscript.com/MyScriptCloud/3.0.0/index.html`

MyScript. (2015b). *Handwriting recognition and digital ink.*

MyScript. (2015c). *Myscript cloud developer guide.* Retrieved May 20

2015, from http://doc.myscript.com/MyScriptCloud/3.0.0/index.html

MyScript. (2015d). *Myscript cloud documentation.* Retrieved 5 June 2015, from http://doc.myscript.com/MyScriptCloud/2.2.0/html/appendices/MS_recognition_overview.html

MyScript. (2015e). *What can be recognized?* Retrieved from http://doc.myscript.com/MyScriptCloud/3.0.0/index.html

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

Personalized, T. (2014). *Does your tablet need a stylus?* Retrieved 1 June 2015, from http://techpp.com/2012/04/03/does-your-tablet-need-a-stylus/

Postscapes. (n.d.). *Educational pen: Lernstift.* Retrieved 1 June 2015, from http://postscapes.com/educational-pen-lernstift

REPORTER, D. M. (2010). Hospitals replace pen and paper with new hi-tech 'digi-pens'. *MailOnline*.

Rose, M., & Bezjak, A. (2009). Logistics of collecting patient-reported outcomes (pros) in clinical practice: an overview and practical examples. *Quality of Life Research*, *18*(1), 125–136.

Sellen, A. J., & Harper, R. H. (2002). *The myth of the paperless office.* MIT press.

Signer, B. (2013, April). *Cross-media information spaces and architectures (cisa).* Retrieved 1 June 2015, from http://www.slideshare.net/signer/crossmedia-information-spaces-and-architectures-cisa

Signer, B., & Norrie, M. C. (2007). Paperpoint: a paper-based presentation and interactive paper prototyping tool. In B. Ullmer & A. Schmidt (Eds.), *Proceedings of the 1st international conference on tangible and embedded interaction* (pp. 57–64).

Signer, B., Norrie, M. C., Grossniklaus, M., Belotti, R., Decurtins, C., & Weibel, N. (2006). Paper-based mobile access to databases. In C. Yu, P. Scheuermann, & S. Chaudhuri (Eds.), *Proceedings of the 2006 acm sigmod international conference on management of data* (pp. 763–765).

Spence, R. (2001). *Information visualization* (Vol. 1). Springer.

Van Thienen, D. (2014a). *Smart Study: An Educational Platform Using Digital Pen and Paper* (Unpublished master's thesis). Vrije Universiteit Brussel.

Van Thienen, D. (2014b). *Studying the Effects on Modelling Tools using Digital Pen and Paper.* (Research Training, Vrije Universiteit Brussel)

Wacom. (n.d.-a). *Wacom bamboo tablet.* Retrieved from `http://bamboo.wacom.com/`

Wacom. (n.d.-b). *Wacom inkling.* Retrieved from `http://inkling.wacom.com/`

Yeh, R., Klemmer, S., & Paepcke, A. (2007). Design and evaluation of an event architecture for paper uis: Developers create by copying and combining. *Stanford InfoLab Publication Server*.

Yen, P.-Y., & Gorman, P. N. (2005). Usability testing of a digital pen and paper system in nursing documentation. In *Amia annual symposium proceedings* (Vol. 2005, p. 844).

A Code

A.1 Original Penlet Code

Fantastico.java

```
package orm;

import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

import orm.exceptions.StrokeContainerNotLoadedException;
import orm.exceptions.StrokeContainerNotSavedException;

import com.livescribe.penlet.Command;
import com.livescribe.penlet.DataReceiver;
import com.livescribe.penlet.DataSender;
import com.livescribe.penlet.Penlet;
import com.livescribe.penlet.Remote;
import com.livescribe.afp.PageInstance;
import com.livescribe.afp.Document;
import com.livescribe.display.Display;
import com.livescribe.event.PaperListener;
import com.livescribe.penlet.Region;
import com.livescribe.storage.PenletStorage;
import com.livescribe.storage.StrokeStorage;
```

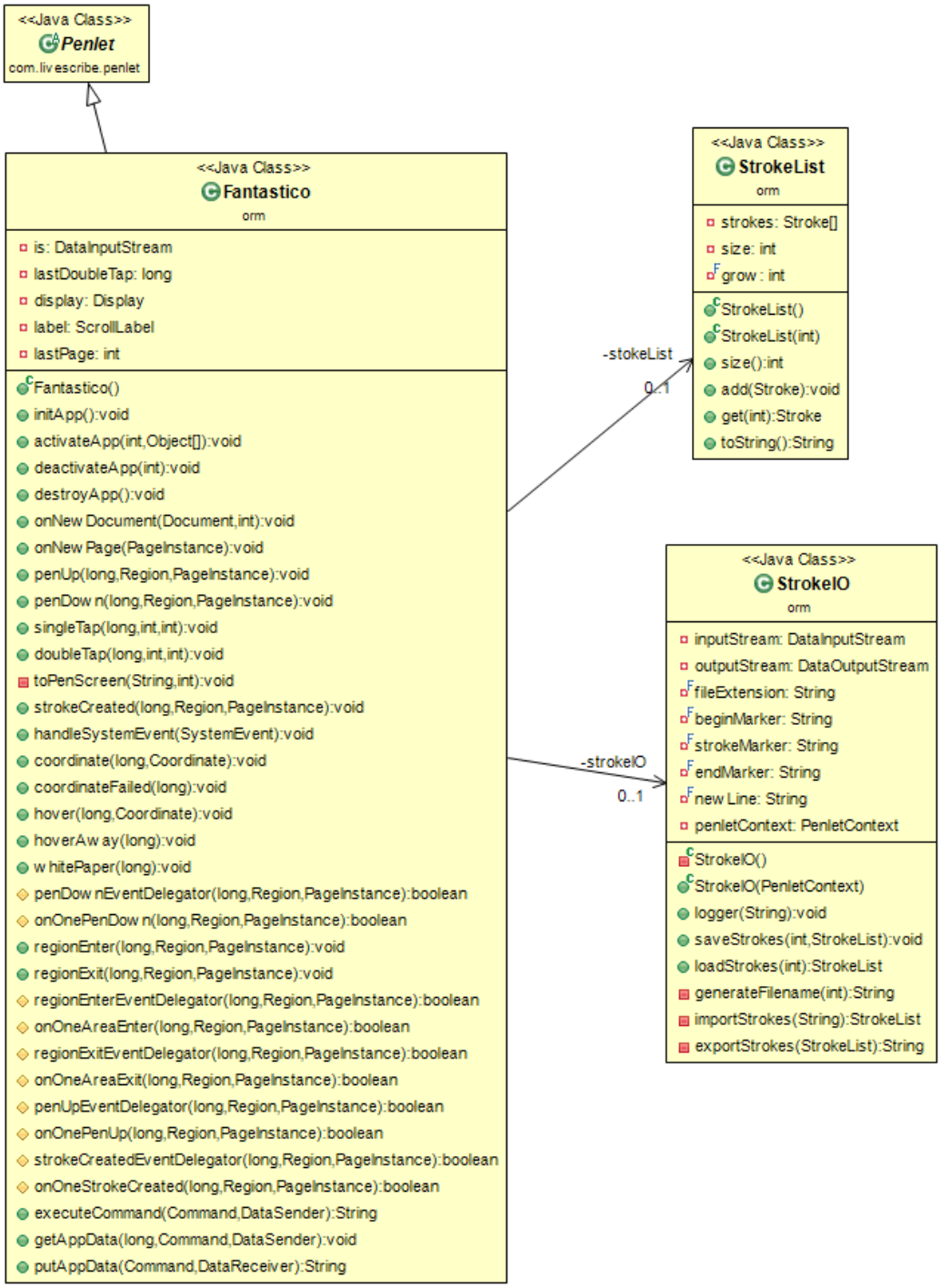


Figure A.1: class diagram of the penlet code

```
22 import com.livescribe.ui.ScrollLabel;
   import com.livescribe.event.PenTipListener;
24 import com.livescribe.event.StrokeListener;
   import com.livescribe.event.SystemEvent;
26 import com.livescribe.event.SystemEventListener;
   import com.livescribe.geom.Coordinate;
28 import com.livescribe.geom.Stroke;
   import com.livescribe.event.CoordinateListener;
30 import com.livescribe.event.RegionEnterExitListener;

32
   public class Fantastico extends Penlet implements PaperListener,
       PenTipListener, StrokeListener, SystemEventListener,
       CoordinateListener, RegionEnterExitListener, Remote {
34

36   private DataInputStream is;

38     private long lastDoubleTap;

40   private Display display;
       private ScrollLabel label;
42
       private StrokeList stokeList = new StrokeList();
44     private StrokeIO strokeIO;

46     private int lastPage = -1;


48
       public Fantastico() {
50
       }
52
       /**
54      * Invoked when the application is initialized.  This happens
       once for an application instance.
        */
56     public void initApp() {

58       this.strokeIO = new StrokeIO(this.getContext());
         this.display = this.getContext().getDisplay();
60         this.label = new ScrollLabel();
       }
62
       /**
64      * Invoked each time the penlet is activated.  Only one
       penlet is active at any given time.
        */
66     public void activateApp(int reason, Object[] args) {
```

```
68      context.addPaperListener(this);
        context.addPenTipListener(this);
70      context.addStrokeListener(this);
        context.addCoordinateListener(this);
72      context.addRegionEnterExitListener(this);
        }
74
        /**
76       * Invoked when the application is deactivated.
         */
78      public void deactivateApp(int reason) {

80      context.removePaperListener(this);
        context.removePenTipListener(this);
82      context.removeStrokeListener(this);
        context.removeCoordinateListener(this);
84      context.removeRegionEnterExitListener(this);

86      }

88      /**
         * Invoked when the application is destroyed.  This happens
       once for an application instance.
90       * No other methods will be invoked on the instance after
       destroyApp is called.
         */
92      public void destroyApp() {
          try {
94        this.strokeIO.saveStrokes(this.lastPage, this.stokeList);
        } catch (Exception e) {
96        // TODO Auto-generated catch block
          e.printStackTrace();
98      }

100     }

102   public void onNewDocument(Document document, int copy) { }

104   public void onNewPage(PageInstance pageInstance) { }

106   public void penUp(long time, Region region, PageInstance page)
       {
        penUpEventDelegator(time, region, page);
108   }

110   public void penDown(long time, Region region, PageInstance page
       ) {
        penDownEventDelegator(time, region, page);
```

```
112   }

114   public void singleTap(long time, int x, int y) {

116   }

118   public void doubleTap(long time, int x, int y) {

120     if(time-200<this.lastDoubleTap && stokeList.size()>0) {
        this.stokeList = new StrokeList();
122       try {
          this.strokeIO.saveStrokes(this.lastPage, this.stokeList);
124
          this.toPenScreen("Erased page "+this.lastPage +" ï", 200)
      ;
126
          this.toPenScreen("Erased page "+this.lastPage + "  ",
      200);
128
          this.toPenScreen("Erased page "+this.lastPage+ " ï", 200)
      ;
130
          this.toPenScreen("Erased page "+this.lastPage+ "  ", 200)
      ;
132
          this.toPenScreen("Erased page "+this.lastPage+" ï", 200);
134
          this.toPenScreen("Erased page "+this.lastPage + "  ",
      200);
136
          this.toPenScreen("Erased page "+this.lastPage+ " ï", 200)
      ;
138       } catch (Exception e) {
          // TODO Auto-generated catch block
140       e.printStackTrace();
        }
142
      }
144
      this.lastDoubleTap = time;
146

148   }

150   private void toPenScreen(String toDisp, int time) {
        strokeIO.logger(toDisp);
152
      this.label.draw(toDisp, true);
154
```

```
        this.display.setCurrent(this.label);
156
          try {
158
    Thread.sleep(time);
160
    } catch (InterruptedException e) { }
162


164
  }
166
  public synchronized void strokeCreated(long time, Region region
    , PageInstance page) {
168
    //User is trying to double tap
170  if(time-5<this.lastDoubleTap) {

172    return;

174  }

176  strokeCreatedEventDelegator(time, region, page);

178  //If the page has changed, we save the old, en load the new
    page
    if(page.getPage()!=this.lastPage){
180
      try {
182
        this.strokeIO.saveStrokes(this.lastPage, this.stokeList);
184
        this.lastPage = page.getPage();
186
        this.stokeList = this.strokeIO.loadStrokes(page.getPage()
    );
188
      } catch (StrokeContainerNotLoadedException e) {
190
          strokeIO.logger(e.getMessage());
192

194       this.toPenScreen("err-" + e.getMessage(),1000);

196     } catch (StrokeContainerNotSavedException e) {
          strokeIO.logger(e.getMessage());
198
        this.toPenScreen("err-" + e.getMessage(),1000);
200
```

```
      }
202
    }
204

206 StrokeStorage strokeStorage = new StrokeStorage(page);

208 Stroke capturedStroke = strokeStorage.getStroke(time);

210 if (capturedStroke == null) {
      this.toPenScreen("a NULL stroke", 1);
212 } else if (capturedStroke.getXY().length == 0) {
      this.toPenScreen("an empty stroke", 1);
214 }

216 stokeList.add(capturedStroke);

218
    this.toPenScreen("P "+this.lastPage+" ï "+stokeList.toString(),
      1);
220
    }
222
    public void handleSystemEvent(SystemEvent sysEvent) {
224
      if (sysEvent.eventId== 142) { //USB CONNECTED
226
        try {
228
          this.strokeIO.saveStrokes(this.lastPage, this.stokeList);
230
        } catch (Exception e) {
232
          // TODO Auto-generated catch block
234         e.printStackTrace();

236     }

238   }

240 }

242 public void coordinate(long time, Coordinate coord) {

244


246 }

248 public void coordinateFailed(long time) {
```

```
250   }

252   public void hover(long time, Coordinate coord) {

254
      }
256
      public void hoverAway(long time) {
258
      }
260
      public void whitePaper(long time) {
262
      }
264

266   // *** GENERATED METHOD -- DO NOT MODIFY ***
      /**
268    * Responsible for delegating generic PEN_DOWN events to area
       specific pen down methods based upon the area in which
       * they occurred. This method is generated and managed by the
       penlet project nature. Users should modify the
270    * individual area event handlers (e.g. on<AreaName>PenDown) or
        the generic event handler (e.g. penDown)).

272    * @param time The time at which the event occurred
       * @param region The region in which the event occurred
274    * @param page The page on which the event occurred
       * @return true if the event was successfully handled, false
       otherwise
276    */
      protected boolean penDownEventDelegator(long time, Region
       region, PageInstance page) {
278     boolean eventHandled = true;
        switch (region.getAreaId()) {
280   case 1:
          eventHandled = onOnePenDown(time, region, page);
282       break;
        default:
284       eventHandled = false;
        }
286     return eventHandled;
      }
288   // *** END OF GENERATED CODE ***

290   protected boolean onOnePenDown(long time, Region region,
       PageInstance page) {
        return true;
```

```
292   }

294   public void regionEnter(long time, Region region, PageInstance
       page) {
        regionEnterEventDelegator(time, region, page);
296
      }
298
      public void regionExit(long time, Region region, PageInstance
       page) {
300    regionExitEventDelegator(time, region, page);

302   }

304   // *** GENERATED METHOD -- DO NOT MODIFY ***
      /**
306    * Responsible for delegating generic PEN_DOWN events to area
       specific pen down methods based upon the area in which
       * they occurred. This method is generated and managed by the
       penlet project nature. Users should modify the
308    * individual area event handlers (e.g. on<AreaName>PenDown) or
        the generic event handler (e.g. penDown)).

310    * @param time The time at which the event occurred
       * @param region The region in which the event occurred
312    * @param page The page on which the event occurred
       * @return true if the event was successfully handled, false
       otherwise
314    */
      protected boolean regionEnterEventDelegator(long time, Region
       region, PageInstance page) {
316    boolean eventHandled = true;
       switch (region.getAreaId()) {
318   case 1:
         eventHandled = onOneAreaEnter(time, region, page);
320      break;
       default:
322      eventHandled = false;
       }
324    return eventHandled;
      }
326   // *** END OF GENERATED CODE ***

328   protected boolean onOneAreaEnter(long time, Region region,
       PageInstance page) {
       return true;
330   }

332   // *** GENERATED METHOD -- DO NOT MODIFY ***
```

```
    /**
334  * Responsible for delegating generic PEN_DOWN events to area
     specific pen down methods based upon the area in which
     * they occurred. This method is generated and managed by the
     penlet project nature. Users should modify the
336  * individual area event handlers (e.g. on<AreaName>PenDown) or
      the generic event handler (e.g. penDown)).

338  * @param time The time at which the event occurred
     * @param region The region in which the event occurred
340  * @param page The page on which the event occurred
     * @return true if the event was successfully handled, false
     otherwise
342  */
    protected boolean regionExitEventDelegator(long time, Region
     region, PageInstance page) {
344   boolean eventHandled = true;
      switch (region.getAreaId()) {
346 case 1:
        eventHandled = onOneAreaExit(time, region, page);
348     break;
      default:
350     eventHandled = false;
      }
352   return eventHandled;
    }
354 // *** END OF GENERATED CODE ***

356 protected boolean onOneAreaExit(long time, Region region,
     PageInstance page) {
      return true;
358 }

360 // *** GENERATED METHOD -- DO NOT MODIFY ***
    /**
362  * Responsible for delegating generic PEN_DOWN events to area
     specific pen down methods based upon the area in which
     * they occurred. This method is generated and managed by the
     penlet project nature. Users should modify the
364  * individual area event handlers (e.g. on<AreaName>PenDown) or
      the generic event handler (e.g. penDown)).

366  * @param time The time at which the event occurred
     * @param region The region in which the event occurred
368  * @param page The page on which the event occurred
     * @return true if the event was successfully handled, false
     otherwise
370  */
```

```
    protected boolean penUpEventDelegator(long time, Region region,
       PageInstance page) {
372   boolean eventHandled = true;
      switch (region.getAreaId()) {
374 case 1:
        eventHandled = onOnePenUp(time, region, page);
376     break;
      default:
378     eventHandled = false;
      }
380   return eventHandled;
    }
382 // *** END OF GENERATED CODE ***

384 protected boolean onOnePenUp(long time, Region region,
     PageInstance page) {
      return true;
386 }

388 // *** GENERATED METHOD -- DO NOT MODIFY ***
    /**
390  * Responsible for delegating generic PEN_DOWN events to area
     specific pen down methods based upon the area in which
     * they occurred. This method is generated and managed by the
     penlet project nature. Users should modify the
392  * individual area event handlers (e.g. on<AreaName>PenDown) or
      the generic event handler (e.g. penDown)).

394  * @param time The time at which the event occurred
     * @param region The region in which the event occurred
396  * @param page The page on which the event occurred
     * @return true if the event was successfully handled, false
     otherwise
398  */
    protected boolean strokeCreatedEventDelegator(long time, Region
       region, PageInstance page) {
400   boolean eventHandled = true;
      switch (region.getAreaId()) {
402 case 1:
        eventHandled = onOneStrokeCreated(time, region, page);
404     break;
      default:
406     eventHandled = false;
      }
408   return eventHandled;
    }
410 // *** END OF GENERATED CODE ***
```

```
412  protected boolean onOneStrokeCreated(long time, Region region,
       PageInstance page) {
       return true;
414  }

416

     public String executeCommand(Command arg0, DataSender arg1) {
418



420

       int to = arg0.toString().indexOf(":");
422    String pageNummer = arg0.toString().substring(0, to);
       String filename =  pageNummer + ".ink";
424

         String in = "";
426    try {
           PenletStorage store = this.getContext().
       getInternalPenletStorage();
428

           //"+pagenr+".note",
430        this.toPenScreen("ï Sending page " + pageNummer + " ï",
       1);
         is = new DataInputStream(store.openInputStream(filename));
432      this.toPenScreen("ï Sent page"+pageNummer +"ï", 1);
         in = is.readUTF();
434      is.close();
       } catch (IOException e) {
436      this.toPenScreen("ï No page "+pageNummer +" ï", 1);
         e.printStackTrace();
438    }
       return in;
440  }

442  public void getAppData(long arg0, Command arg1, DataSender arg2
       )
         throws IOException {
444    // TODO Auto-generated method stub

446  }

448  public String putAppData(Command arg0, DataReceiver arg1)
         throws IOException {
450    // TODO Auto-generated method stub
       return null;
452  }

454
```

```
}
```

StrokeIO.Java

```
1  package orm;
2
3  import java.io.DataInputStream;
4  import java.io.DataOutputStream;
5  import java.io.IOException;
6  import java.util.Date;
7
8  import orm.exceptions.StrokeContainerNotLoadedException;
9  import orm.exceptions.StrokeContainerNotSavedException;
10
11 import com.livescribe.geom.Stroke;
12 import com.livescribe.penlet.PenletContext;
13 import com.livescribe.storage.PenletStorage;
14
15 public class StrokeIO {
16
17   private DataInputStream  inputStream;
18     private DataOutputStream outputStream;
19
20
21     private final String fileExtension = "ink";
22
23   private final String   beginMarker = "1734 2227\n3\n1\n";
24   private final String  strokeMarker = "4278190080 3 3 0";
25   private final String     endMarker = "2 1\n0\n70\n0\n0";
26
27   private final String       newLine = "\n";
28   private PenletContext penletContext;
29
30
31   //prevents creation without context
32   private StrokeIO() {}
33
34   public StrokeIO(PenletContext penletContext) {
35
36     this();
37     this.penletContext = penletContext;
38
39   }
40
41   public synchronized void logger(String e) {
42
43     try {
44
45     String in = "";
46       PenletStorage store = this.penletContext.
47    getInternalPenletStorage();
```

```
47
      inputStream = new DataInputStream( store.openInputStream(
    this.generateFilename(99) ) );
49
     in = inputStream.readUTF();
51
     outputStream = new DataOutputStream( store.openOutputStream(
    this.generateFilename(99) ) );
53
55      outputStream.writeUTF(in + "<br/>"+new Date().toString()+e)
    ;
     } catch (IOException e1) {
57      // TODO Auto-generated catch block
        e1.printStackTrace();
59   } finally {
     try {
61      inputStream.close();
        outputStream.close();
63   } catch (IOException e1) {
        // TODO Auto-generated catch block
65      e1.printStackTrace();
     }
67   }

69

71

73 }

75
  public synchronized void saveStrokes(int pageNr, StrokeList
    strokeList) throws StrokeContainerNotSavedException {
77
     if(pageNr<0) return;
79

81   try {

83        PenletStorage store = this.penletContext.
    getInternalPenletStorage();

85      outputStream = new DataOutputStream( store.openOutputStream
    ( this.generateFilename(pageNr) ) );

87
        String s = this.exportStrokes(strokeList);
89
```

```
91
                // here comes the simple workaround
93              byte[] b = s.getBytes("utf-8");
                outputStream.writeInt(b.length);
95              outputStream.write(b);

97


99

101       outputStream.flush();

103
      } catch (IOException e) {
105
        throw new StrokeContainerNotSavedException("saving-"+e.
    getMessage()+e.toString());
107
      } finally {
109

111       try {
            outputStream.close();
113       } catch (IOException e) {
            // TODO Auto-generated catch block
115         throw new StrokeContainerNotSavedException("saving-"+e.
    getMessage()+e.toString());
          }
117     }

119 }

121
  public synchronized StrokeList loadStrokes (int pageNr) throws
    StrokeContainerNotLoadedException {
123

125     try {

127


129         PenletStorage store = this.penletContext.
    getInternalPenletStorage();

131       inputStream = new DataInputStream( store.openInputStream(
    this.generateFilename( pageNr ) ) );

133       String fileContents = inputStream.readUTF();
```

```
135       return this.importStrokes( fileContents );

137     } catch (IOException e) {

139       throw new StrokeContainerNotLoadedException("loading-"+e.
      getMessage()+e.toString());

141     } finally {
        try {
143       inputStream.close();
        } catch (IOException e) {
145       // TODO Auto-generated catch block
          throw new StrokeContainerNotLoadedException("loading-"+e.
      getMessage()+e.toString());
147     }
      }
149
    }
151


153
    private synchronized String generateFilename(int pageNr) {
155
      return pageNr + "." + this.fileExtension;
157
    }
159

161 private synchronized StrokeList importStrokes(String in) {

163   int intsML = (strokeMarker + newLine).length();

165   int intbML = (beginMarker).length();

167   int cursor = 0;


169
      cursor = in.indexOf(beginMarker, cursor);
171
      cursor += intbML;
173
      int strokeCount = Integer.parseInt(in.substring(cursor, in.
      indexOf(newLine, cursor)));
175

177   StrokeList strokes = new StrokeList(strokeCount);

179
```

```
        for(int j = 0; j<strokeCount;j++) {
181
183       cursor = in.indexOf(strokeMarker, cursor);
185       cursor += intsML;
187
          int coordCount = Integer.parseInt(in.substring(cursor, in.
      indexOf(newLine, cursor)));
189
191       cursor = in.indexOf(newLine, cursor)+ newLine.length();
193       int[] points = new int[coordCount*2];
195
          for(int i = 0; i <coordCount*2;) {
197
199         points[i++] = Integer.parseInt(in.substring(cursor, in.
      indexOf(" ", cursor)));
201         cursor = in.indexOf(" ", cursor)+1;
203
            points[i++] = Integer.parseInt(in.substring(cursor, in.
      indexOf(newLine, cursor)));
205         cursor = in.indexOf(newLine, cursor)+1;
207
209       }
211
          strokes.add(new Stroke(points));
213
215     }
217
        return strokes;
219
      }
221
223   private synchronized String exportStrokes(StrokeList strokeList
        ){
```

```
225     StringBuffer buffer = new StringBuffer();
227     buffer.append( beginMarker );
229     buffer.append( strokeList.size() );
231     buffer.append( newLine );
233
        for( int i = 0 ; i < strokeList.size() ; i++ ) {
235
          Stroke stroke = strokeList.get(i);
237
          int[] coords = stroke.getXY();
239
          buffer.append(  strokeMarker);
241
          buffer.append( newLine );
243
          buffer.append( coords.length / 2 );
245
          buffer.append( newLine );
247
          for(int p =0; p<coords.length;p++) {
249
            if( p % 2 == 0 ) {
251
              buffer.append( coords[p] + " " );
253
            } else {
255
              buffer.append( coords[p] + "\n" );
257
            }
259
          }
261
        }
263
265     buffer.append(endMarker);
267     return buffer.toString();
269   }
271 }
```

StrokeList.java

```
package orm;

import com.livescribe.geom.Stroke;


public class StrokeList {

  private Stroke[] strokes;
  private int size = 0;
  private final int grow = 10;

  /**
   * Basic Array List with an initial size of 10
   */
  public StrokeList() {

    this(10);

  }

  /**
   * Basic Array with
   * @param initialSize
   */
  public StrokeList(int initialSize) {

    this.strokes = new Stroke[initialSize];
  }

  public synchronized int size() {

    return this.size;

  }

  public synchronized void add(Stroke e) {

    // If the array is full
    if (this.size == this.strokes.length) {

      // Make a new array, 'this.grow' bigger than the original
      Stroke[] newStrokes = new Stroke[this.strokes.length + this
    .grow];

      // Copy original ref's into new array
      System.arraycopy(this.strokes, 0, newStrokes, 0, this.
    strokes.length);
```

```
46
        //Set the new array as 'the original'
48      this.strokes = newStrokes;
      }
50
      //Add the element
52    this.strokes[size++] = e;
    }
54
    public synchronized Stroke get(int index) throws
     ArrayIndexOutOfBoundsException {
56
      return this.strokes[index];
58
    }
60

62  public synchronized String toString() {

64    return "S " + this.size();

66  }

68 }
```

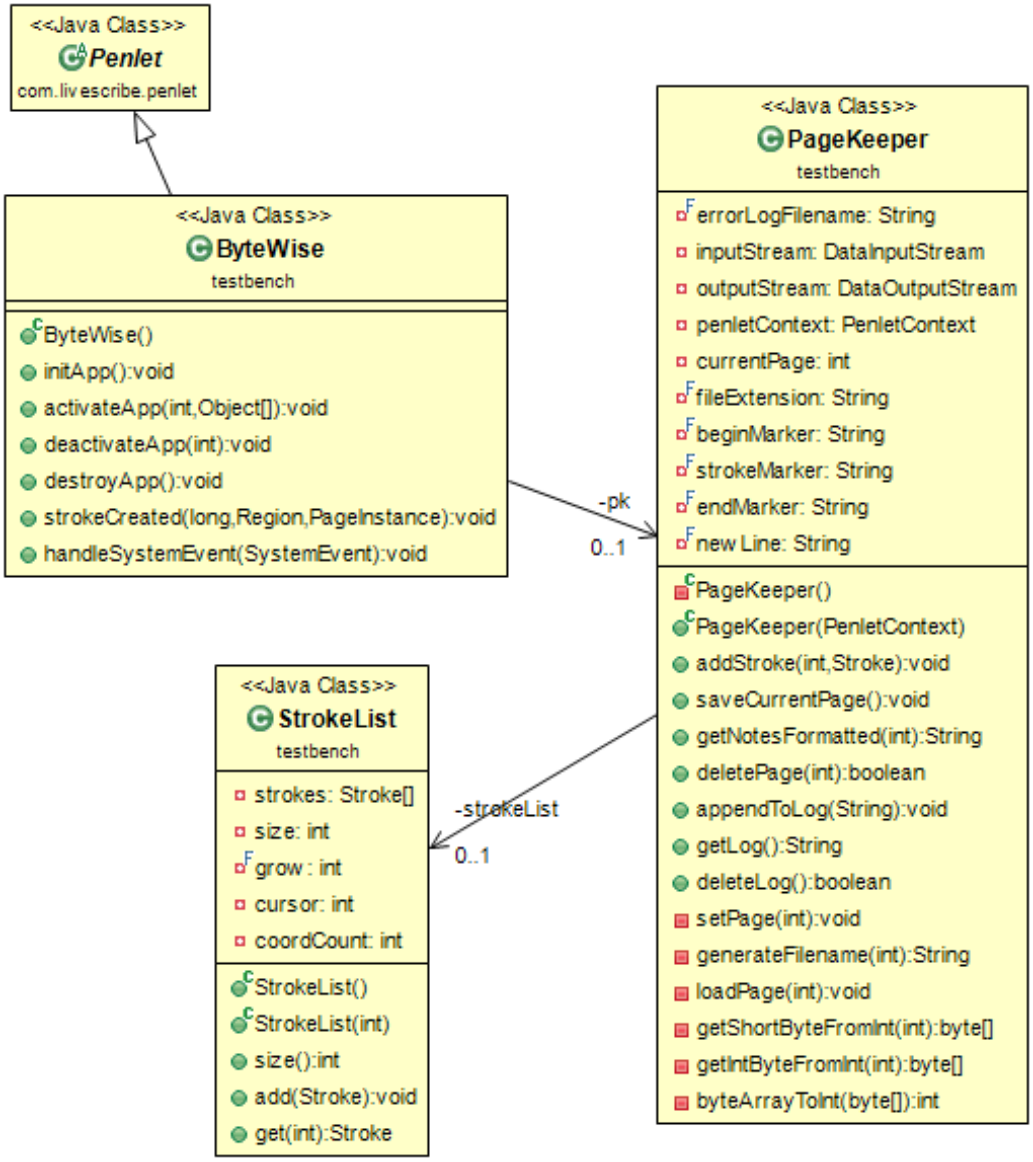
A.2 ByteWise penlet code

Bytewise.java

```
1 package testbench;

3 import java.io.IOException;

5 import com.livescribe.penlet.Penlet;
  import com.livescribe.penlet.Region;
7 import com.livescribe.storage.StrokeStorage;
  import com.livescribe.afp.PageInstance;
9 import com.livescribe.event.StrokeListener;
  import com.livescribe.event.SystemEvent;
11 import com.livescribe.event.SystemEventListener;
  import com.livescribe.geom.Stroke;
13
  public class ByteWise extends Penlet implements StrokeListener,
     SystemEventListener {
15
    private PageKeeper pk = null;
```




```
17
    public ByteWise() {
19  }

21  /**
     * Invoked when the application is initialized. This happens
    once for an application instance.
23   */
```

```
    public void initApp() {
25  }

27  /**
     * Invoked each time the penlet is activated.  Only one
    penlet is active at any given time.
29   */
    public void activateApp(int reason, Object[] args) {
31  context.addStrokeListener(this);
    pk =  new PageKeeper(this.getContext());
33  }

35  /**
     * Invoked when the application is deactivated.
37   */
    public void deactivateApp(int reason) {
39  context.removeStrokeListener(this);
    }
41
    /**
43   * Invoked when the application is destroyed.  This happens
    once for an application instance.
     * No other methods will be invoked on the instance after
    destroyApp is called.
45   */
    public void destroyApp() {
47    try {
      pk.saveCurrentPage();
49  } catch (IOException e) {
      // TODO Auto-generated catch block
51    e.printStackTrace();
    }
53  }

55 public void strokeCreated(long time, Region region,
    PageInstance page) {

57



59
    StrokeStorage strokeStorage = new StrokeStorage(page);
61
    Stroke stroke = strokeStorage.getStroke(time);
63  try {
      this.pk.addStroke(page.getPage(), stroke);
65  } catch (Exception e) {
      // TODO Auto-generated catch block
67    e.printStackTrace();
    }
```

```
69
    }
71
    public void handleSystemEvent(SystemEvent sysEvent) {
73
      if (sysEvent.eventId== 142) { //USB CONNECTED
75
        try {
77
          this.pk.saveCurrentPage();
79
        } catch (Exception e) {
81
          // TODO Auto-generated catch block
83        e.printStackTrace();

85      }

87    }

89  }

91 }
```

PageKeeper.java

```
  package testbench;
2
  import java.io.ByteArrayOutputStream;
4 import java.io.DataInputStream;
  import java.io.DataOutputStream;
6 import java.io.IOException;
  import java.util.Date;
8
  import com.livescribe.geom.Stroke;
10 import com.livescribe.penlet.PenletContext;
  import com.livescribe.storage.PenletStorage;
12
  public class PageKeeper {
14
    private final String errorLogFilename = "error.log";
16  private DataInputStream  inputStream;
      private DataOutputStream outputStream;
18    private PenletContext penletContext;

20    private int currentPage = -1;
    private StrokeList strokeList = new StrokeList();
22
```

```
    private final String fileExtension = ".ink";
24
    private final String   beginMarker = "1734 2227\n3\n1\n";
26  private final String  strokeMarker = "4278190080 3 3 0";
    private final String     endMarker = "2 1\n0\n70\n0\n0";
28  private final String       newLine = "\n";

30
    private PageKeeper(){ //Prevents creation without PenletContext
32
    }
34
    public PageKeeper(PenletContext penletContext) {
36    this.penletContext = penletContext;

38  }

40  /******************
     * Public Methods *
42   ******************/

44  /**
     * Add stroke to certain page
46   * @param page page number where the stroke was added
     * @param stroke the stroke
48   * @throws Exception If adding the stroke failes
     */
50  public void addStroke(int page, Stroke stroke) throws Exception
      {

52    if(stroke.getXY().length > 65534) throw new
     NumberFormatException("Stroke too big");

54    this.setPage(page);
      this.strokeList.add(stroke);
56
    }
58
    /**
60   * Save the current active pages strokes
     * @throws IOException
62   */
    public void saveCurrentPage() throws IOException {
64
      ByteArrayOutputStream stream = new ByteArrayOutputStream();
66
      try {
68
```

```
70        //Get access to internal storage
            PenletStorage store = this.penletContext.
      getInternalPenletStorage();
72        outputStream = new DataOutputStream( store.openOutputStream
      ( this.generateFilename(this.currentPage) ) );

74

76        //Write the amount of strokes
          stream.write( getShortByteFromInt(this.strokeList.size()) )
      ;
78
          //Write the strokes
80        for (int i = 0; i< this.strokeList.size(); i++ ){

82          int[] strokeCoords = this.strokeList.get(i).getXY();

84          //Write the amount of coordinates
            stream.write( getShortByteFromInt(strokeCoords.length/2)
      );
86
            //Write the XY coordinates vector
88          for (int j = 0; j< strokeCoords.length; j++ ){

90            stream.write( getShortByteFromInt(strokeCoords[j]) );

92          }

94        }

96        stream.flush();

98        //Retrieve the size of the stream, and write it
          outputStream.write(getIntByteFromInt(stream.size()));
100
          //Write the stream
102       outputStream.write(stream.toByteArray());

104       outputStream.flush();

106

108     } catch (IOException e) {

110       throw new IOException("saving-"+e.getMessage()+e.toString()
      );

112     } finally {
```

```
114         stream.close();

116         try {

118           outputStream.close();

120         } catch (IOException e) {

122           throw new IOException("saving-"+e.getMessage()+e.toString
      ());

124         }
       }
126
    }
128
    /**
130  * Returns the page strokes in notes-formatting
     * @param page the page to retrieve
132  * @return the requested page in notes-formatting
     * @throws IOException
134  * @throws OutOfMemoryError
     */
136  public String getNotesFormatted(int page) throws IOException,
      OutOfMemoryError {

138   //If there is still a page active, we save it first
      if(this.currentPage>-1) this.saveCurrentPage();
140
      PenletStorage store = this.penletContext.
     getInternalPenletStorage();
142
      inputStream = new DataInputStream( store.openInputStream(
     this.generateFilename( page ) ) );
144
      byte[] byteShort ;
146
      //Retrieve the file size (file size - 4 bytes, since the int
     itself is not counted)
148
      byteShort = new byte[]{inputStream.readByte(), inputStream.
     readByte(), inputStream.readByte(), inputStream.readByte()};
150   int fileSize = this.byteArrayToInt(byteShort);

152   StringBuffer  buffer = new StringBuffer();

154     buffer.append( beginMarker );
```

```
156     byteShort = new byte[]{inputStream.readByte(), inputStream.
        readByte()};
        short strokeCount = (short) ((byteShort[1] << 8) + (byteShort
        [0]&0xFF));
158
           buffer.append(strokeCount);
160        buffer.append( newLine );

162     for(short i = 0; i < strokeCount; i++) {

164           buffer.append(  strokeMarker);
              buffer.append( newLine );
166
           byteShort = new byte[]{inputStream.readByte(), inputStream.
        readByte()};
168        short coordCount = (short) ((byteShort[1] << 8) + (
        byteShort[0]&0xFF));

170           buffer.append( coordCount );
              buffer.append( newLine );
172


174
           for(short j = 0; j<coordCount*2;j++ ) {
176
             byteShort = new byte[]{inputStream.readByte(),
        inputStream.readByte()};
178
             if( j % 2 == 0 ) {
180
               buffer.append( (short) ((byteShort[1] << 8) + (
        byteShort[0]&0xFF)) + " " );
182
             } else {
184
               buffer.append( (short) ((byteShort[1] << 8) + (
        byteShort[0]&0xFF)) + "\n" );
186
             }
188


190
           }
192
        }
194     buffer.append(endMarker);

196     return buffer.toString();
```

```
198   }

200   /**
       * Delete a page
202    * @param page page number
       * @return whether deleting was sucesssful
204    */
      public boolean deletePage(int page) {
206
        PenletStorage store = this.penletContext.
       getInternalPenletStorage();
208     try {

210       //If the file is non existent, deleting 'succeeded'
          if ( !store.exists( ( this.generateFilename( page ) ) ) ) {
212         return true;
          }
214
          //Try to delete the file
216       return store.delete( this.generateFilename( page ) );

218     } catch (IOException e) {

220       return false;

222     }

224   }

226   /**
       * Logs text to error log, with date/time in front
228    * @param err String to log
       */
230   public void appendToLog(String err) {

232     try {


234
          //Get access to internal storage
236         PenletStorage store = this.penletContext.
       getInternalPenletStorage();
          outputStream = new DataOutputStream( store.openOutputStream
       ( this.errorLogFilename , true) ); //Makes append
238
          outputStream.writeUTF((new Date()).toString()+" "+ err + "<
       br/>");
240       outputStream.flush();

242     } catch (IOException e) {
```

```
244     } finally {

246       try {

248         outputStream.close();

250       } catch (IOException e) {

252       }

254     }

256   }

258   public String getLog() {
        PenletStorage store = this.penletContext.
      getInternalPenletStorage();
260     try {

262       inputStream = new DataInputStream( store.openInputStream(
      this.errorLogFilename ) );
        return inputStream.readUTF();
264
      } catch (IOException e) {
266     return "Could not read Log - " + e.getMessage();
      }
268

270   }

272   public boolean deleteLog() {

274     PenletStorage store = this.penletContext.
      getInternalPenletStorage();
        try {
276
        //If the file is non existent, deleting 'succeeded'
278     if ( !store.exists( (  this.errorLogFilename ) ) ) {
          return true;
280     }

282     //Try to delete the file
        return store.delete( this.errorLogFilename );
284
      } catch (IOException e) {
286
        return false;
288
```

```
    }
290
  }
292

294 /*******************
   * Private methods *
296  *******************/

298 private void setPage(int page) throws IOException {

300   if( page != this.currentPage ) {

302     if( this.currentPage > -1 ) {

304       try {

306         this.saveCurrentPage();

308       } catch (Exception e) {

310         throw new IOException("Could not save current page (" +
    this.currentPage + ") - " + e.getMessage());

312       }

314     }

316     this.currentPage = page;

318     try {

320       this.loadPage(this.currentPage);

322     } catch (IOException e) {

324       this.strokeList = new StrokeList();

326     };

328   }

330 }

332 private String generateFilename(int pageNr) {

334   return pageNr + "." + this.fileExtension ;

336 }
```

```
338   private void loadPage(int page) throws IOException {

340     PenletStorage store = this.penletContext.
      getInternalPenletStorage();

342     inputStream = new DataInputStream( store.openInputStream(
      this.generateFilename( page ) ) );

344     byte[] byteShort ;

346     //Retrieve the file size (file size - 4 bytes, since the int
      itself is not counted)
      byteShort = new byte[]{inputStream.readByte(), inputStream.
      readByte(), inputStream.readByte(), inputStream.readByte()};
348     int fileSize = this.byteArrayToInt(byteShort); //Not in use
      for the moment

350     byteShort = new byte[]{inputStream.readByte(), inputStream.
      readByte()};
      short strokeCount = (short) ((byteShort[1] << 8) + (byteShort
      [0]&0xFF));
352
      this.strokeList = new StrokeList(strokeCount);
354     for(short i = 0; i < strokeCount; i++) {

356       byteShort = new byte[]{inputStream.readByte(), inputStream.
      readByte()};
        short coordCount = (short) ((byteShort[1] << 8) + (
      byteShort[0]&0xFF));
358

360       int[]stroke = new int[coordCount*2];

362       for(int j = 0; j<coordCount*2;j++ ) {

364         byteShort = new byte[]{inputStream.readByte(),
      inputStream.readByte()};
          stroke[j] = (int) (short) ((byteShort[1] << 8) + (
      byteShort[0]&0xFF));
366
        }
368
        this.strokeList.add(new Stroke(stroke));
370

372     }

374   }
```

```
376
    /***************
378  * Conversions *
     ***************/
380
    private byte[] getShortByteFromInt(int i) {
382
      return new byte[]{
384
          (byte)(((short)i) ),
386       (byte)((((short)i) >> 8) & 0xff)

388   };

390 }

392 private byte[] getIntByteFromInt(int i) {

394   return new byte[] {

396           (byte) ((i >> 24) & 0xFF),
              (byte) ((i >> 16) & 0xFF),
398           (byte) ((i >> 8) & 0xFF),
              (byte) (i & 0xFF)
400
      };
402
    }
404
    private int byteArrayToInt(byte[] b) {
406
        return   b[3] & 0xFF |
408             (b[2] & 0xFF) << 8 |
                (b[1] & 0xFF) << 16 |
410             (b[0] & 0xFF) << 24;

412 }

414 }
```

StrokeList.java

```
1 package testbench;

3 import com.livescribe.geom.Stroke;

5
```

```
public class StrokeList{

  private Stroke[] strokes;
  private int size = 0;
  private final int grow = 10;
  private int cursor;
  private int coordCount = 0;

  /**
   * Basic Array List with an initial size of 10
   */
  public StrokeList() {

    this(10);

  }

  /**
   * Basic Array with
   * @param initialSize
   */
  public StrokeList(int initialSize) {

    this.strokes = new Stroke[initialSize];
  }

  public synchronized int size() {

    return this.size;

  }

  public synchronized void add(Stroke e) throws
   ArrayStoreException {

    System.out.println(this.coordCount);

    if ((this.coordCount) > 4000000) throw new
   ArrayStoreException("StokeList is full");

    // If the array is full
    if (this.size == this.strokes.length) {

      // Make a new array, 'this.grow' bigger than the original
      Stroke[] newStrokes = new Stroke[this.strokes.length + this
   .grow];

      // Copy original ref's into new array
```

```
51          System.arraycopy(this.strokes, 0, newStrokes, 0, this.
        strokes.length);

53          //Set the new array as 'the original'
            this.strokes = newStrokes;
55      }

57      //Add the element
        this.strokes[size++] = e;
59
        //Add coordCount
61      this.coordCount += ( e.getXY().length );

63    }

65    public synchronized Stroke get(int index) throws
        ArrayIndexOutOfBoundsException {

67      return this.strokes[index];

69    }

71


73  }
```

A.3 Server Code

WebServer.cs

```
1 using System;
  using System.Net;
3 using System.Threading;
  using System.Linq;
5 using System.Text;

7 /**
   * WebServer - Basic webserver
9  * By: Tania Van Denhouwe
   * Based upon: https://codehosting.net/blog/BlogEngine/post/
     Simple-C-Web-Server.aspx
11 * License original: MIT
   */
13
  namespace Draw2Model
```

```
15 {
     public class WebServer
17   {
         private readonly HttpListener _listener = new
     HttpListener();
19       private readonly Func<HttpListenerRequest, SimpleResponse
     > _responderMethod;

21       public WebServer(Func<HttpListenerRequest, SimpleResponse
     > method, string prefixes)
         {
23           if (!HttpListener.IsSupported)
                 throw new NotSupportedException(
25                   "Needs Windows XP SP2, Server 2003 or later."
     );

27           // URI prefixes are required, for example
             // "http://localhost:8080/index/".
29           if (prefixes == null || prefixes.Length == 0)
                 throw new ArgumentException("prefixes");
31
             // A responder method is required
33           if (method == null)
                 throw new ArgumentException("method");
35


37               _listener.Prefixes.Add(prefixes);

39           _responderMethod = method;
             _listener.Start();
41       }


43

45       public void Run()
         {
47           ThreadPool.QueueUserWorkItem((o) =>
             {
49               Console.WriteLine("Webserver running...");
                 try
51               {
                     while (_listener.IsListening)
53                   {
                         ThreadPool.QueueUserWorkItem((c) =>
55                       {
                             var ctx = c as HttpListenerContext;
57                           try
                             {
```

```
59                                  SimpleResponse sr =
    _responderMethod(ctx.Request);
                                    string rstr = sr.content;
61                                  byte[] buf = sr.getContent();
                                    ctx.Response.ContentType = sr.
    contentType;
63                                  ctx.Response.ContentLength64 =
    buf.Length;

65                                  ctx.Response.OutputStream.Write(
    buf, 0, buf.Length);
                                }
67                              catch { } // suppress any exceptions
                                finally
69                              {
                                    // always close the stream
71                                  ctx.Response.OutputStream.Close()
    ;
                                }
73                          }, _listener.GetContext());
                        }
75                  }
                    catch { } // suppress any exceptions
77              });
        }
79
        public void Stop()
81      {
            _listener.Stop();
83          _listener.Close();
        }
85  }
}
```

PenExtractor.cs

```
using Livescribe.DesktopSDK.AFP;
using Livescribe.DesktopSDK.PenComm;
using Livescribe.DesktopSDK.PenComm.Interop;
using Livescribe.DesktopSDK.PenData;
using Livescribe.DesktopSDK.PenData.Drawing;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Runtime.Remoting.Contexts;
using System.Text;
using System.Threading.Tasks;

/**
 * PenExtracor.cs
 * By: Van Denhouwe Tania
 * Mainly based upon a LiveScribe DesktopSDK Example
 *  '-> SDK_Livescribe\Livescribe_DesktopSDK_0.7\Samples\
   DesktopApps\cs\DecodeStrokesForAllAFDs
 */

namespace Draw2Model
{
    class PenExtractor
    {
        private const string PENCOMM_LOG    = "PenInformation.log
    "
                            , beginMarker    = "1734 2227\n3\n1\n"
                            , strokeMarker   = "4278190080 3 3 0"
                            , endMarker      = "2 1\n0\n70\n0\n0"
                            , newLine        = "\n";

        private string  webRoot
                        ,   result  = ""
                        , guid = "",
                        DOC_DATA_FILE = Environment.GetFolderPath(
    Environment.SpecialFolder.ApplicationData) + "\\d2m\\data.zip
    "
                            ;

        private int page, copy, _flag = 0;
        private bool full;
```

```
44          private static object _lock = new object(), _lock_flag =
    new object();

46          //For keeping track of the thumbnails last refresh/load
            private Dictionary<String, long> needToReload = null;
48
            private Smartpens smartpens = null;
50
            private DateTime lastPenComm = DateTime.UtcNow-TimeSpan.
    FromSeconds(7);
52
            private PenExtractor() { }
54
            public PenExtractor(String webRoot)
56          {

58
                //path to the webroot, so we can put the tumbnails/
    images there, extracted from the pen
60              this.webRoot = webRoot;

62
                try
64              {
                    //Try to load the needToReload Dictonary from
    last run
66                  needToReload = Read(Environment.GetFolderPath(
    Environment.SpecialFolder.ApplicationData) + "\\d2m\\ntr.bin
    ");

68              }
                catch (Exception e)
70              {
                    //If loading failes, we start with an empty
    Dictonary
72                  needToReload = new Dictionary<string, long>();

74              }

76          }

78          /**
             * Tries to get all info about booklets/papers from the
    pen
80           */
            public String getAllPageInfo()
82          {
```

```
84          Console.WriteLine("API handeling thread " + System.
    Threading.Thread.CurrentThread.ManagedThreadId + " is waiting
    !");
            //As long as _flag is not > -1
86          //no new request can be made
            lock (_lock_flag)
88          {

90              while (_flag < 0)
                {
92                  //When an other request is done, it should
    pulse
                    //and _flag <0 is reevaluated. After each 15s
     we try anyway
94                  System.Threading.Monitor.Wait(_lock_flag
    ,15000);

96              }

98              _flag--;

100             Console.WriteLine("API handeling thread " +
    System.Threading.Thread.CurrentThread.ManagedThreadId + "
    going on with _flag set to: " + _flag);
            }
102
            lock (_lock)
104         {
                //To be shure the driver doens't act up, we wait
    a minimum of 7seconds between requests
106             bool b = true;
                while (lastPenComm != null && lastPenComm.
    AddSeconds(7) > DateTime.UtcNow)
108             {

110                 //The boolean b ensures we only output the
    message once
                    if(b) Console.WriteLine("Request for pen by
    thread " + System.Threading.Thread.CurrentThread.
    ManagedThreadId + " succeeds the last one to fast. Waiting...
     ");
112                 b = false;
                    //sleep to spare cpu time
114                 System.Threading.Thread.Sleep(500);//re-try
    every 0.5s

116             }
                //Set the time of this request, as the last
    request
```

```
118                 lastPenComm = DateTime.UtcNow;

120
                    //Here the magic happens
122                 if (smartpens != null) smartpens.Destroy();
                    smartpens = null;
124                 if (smartpens == null) smartpens = new Smartpens(
    true, PENCOMM_LOG);

126

128                     //Create a new event, using the method '
    GetPagesEvent' found in this class
                        var cmm = new Smartpens.
    SmartpenChangeCallback(GetPagesEvent);
130
                        //Attach the event to the smatpens
132                     smartpens.SmartpenAttachNormalEvent += cmm;

134                     //TODO check if there is only one pen! :s

136                     //Find pen and make sure the attached event
    is executed
                        smartpens.Find();
138                     if (smartpens.Pens.Count == 1)
                        {
140                         //Wait until the result is received
                            System.Threading.Monitor.Wait(_lock,
    20000);
142                     }
                        else
144                     {
                            lock (_lock_flag)
146                         {
                                //Put the 'flag' back
148                             _flag++;
                                //Pulse the other waiting requests
150                             System.Threading.Monitor.PulseAll(
    _lock_flag);

152                         }
                            if (smartpens.Pens.Count < 1)
154                         {
                                return "[{\"Guid\":\"0\",\"Title\":\"
    NO PEN FOUND\",\"Pages\":[]}]";
156
                            }
158                         else
                            {
```

```
160                             return "[{\"Guid\":\"0\",\"Title\":\"
    MAX 1 PEN ALLOWED\",\"Pages\":[]}]";
                            }
162                     }

164             }

166
                string finalString = this.result + "";
168
                lock (_lock_flag)
170             {
                    //Put the 'flag' back
172                 _flag++;
                    //Pulse the other waiting requests
174                 System.Threading.Monitor.PulseAll(_lock_flag);

176             }

178             return finalString;

180



182


184         }
            /**
186          * Retreive all book/page related info
             */
188         private void GetPagesEvent(Smartpen pen)
            {
190
                lock (_lock)
192             {

194                 Document doc;
                    bool updateNeeded = false;
196                 string outString = "";

198                 try
                    {
200                     SmartpenChangeList changeList = pen.
    ChangeList;
                        //Get al changes since 'the beginning of time
    '
202                     changeList.Update((ulong)0);
                        //Retreive the list books/pages and there
    changes since 'the beginning of time'
```

```
204                     ReadOnlyCollection<ChangeItem> changeItems =
    changeList.ChangeItems;
                    //JSONify the object, and set it as the
    result
206                     outString = JsonConvert.SerializeObject(
    changeItems);

208                     //Get thumbs for all collections/books
                        foreach (ChangeItem item in changeItems)
210                     {
                            //Check - using the Dictonary
    needToReload if the stored image(s) are still good/up2date
212                         updateNeeded = false;
                            foreach (PageChangeItem pageInfo in item.
    Pages)
214                         {

216                             if (!needToReload.ContainsKey(item.
    Guid + "_" + pageInfo.PageAddress) || needToReload[item.Guid
    + "_" + pageInfo.PageAddress] != pageInfo.EndTime)
                            {
218
                                updateNeeded = true;
220                             break;

222                         }

224                     }

226                     if (updateNeeded) //An image from the
    concerned book/collection is outdated
                        {
228                         //Retreive the data (zip) and load it
     as the 'workset'
                            pen.DataGet(item.Guid, 0,
    DOC_DATA_FILE);
230                         doc = new Container(DOC_DATA_FILE).
    Document;

232                         //pen.DataDeleteStrokes() //TODO make
     separate function to delete data
                            foreach (PageChangeItem pageInfo in
    item.Pages)
234                         {
                                //Log that the page is refreshed
236                             needToReload[item.Guid + "_" +
    pageInfo.PageAddress] = pageInfo.EndTime;
```

```
238                                 //Code to make the PNG files /
    thumbnails
                                    {
240                                     Renderer renderer = new
    Renderer(doc);
                                        renderer.PageNumber =
    pageInfo.Page;
242                                     renderer.CopyNumber =
    pageInfo.Copy;

244                                     Image bmp;
                                        bmp = renderer.RenderToImage
    (44);
246                                     Graphics g = Graphics.
    FromImage(bmp);
                                        g.ScaleTransform(0.06f, 0.06f
    );
248                                     renderer.Render(g);
                                        g.Dispose();
250                                     //Save the thumbs in the
    webroot
                                        string path = this.webRoot +
    "\\thumbs\\" + item.Guid + "\\" + pageInfo.Copy;
252                                     if (!System.IO.Directory.
    Exists(path)) System.IO.Directory.CreateDirectory(path);
                                        bmp.Save(path + "\\" +
    pageInfo.Page + ".png", System.Drawing.Imaging.ImageFormat.
    Png);
254                                     bmp.Dispose();
                                    } //End of png/thumbs code
256

258                             }

260                             doc.LSDocument.Close();
                                doc.Close();
262                         }
                        }
264                 }
                    catch (Exception e) {
266                 this.result = "[{\"Guid\":\"0\",\"Title\":\"
    COMMUNICATION ERROR\",\"Pages\":[]}]";

268                 }
                    finally
270                 {
                        //Save new timestamps
272                     if (!System.IO.Directory.Exists(Environment.
    GetFolderPath(Environment.SpecialFolder.ApplicationData) + "
```

```
\\d2m")) System.IO.Directory.CreateDirectory(Environment.
GetFolderPath(Environment.SpecialFolder.ApplicationData) + "
\\d2m");
                    Write(needToReload, Environment.GetFolderPath
(Environment.SpecialFolder.ApplicationData) + "\\d2m\\ntr.bin
");
274                 //set 'the result'
                    this.result = outString;
276                 lock (_lock)
                    {
278                     //unlock the waiting code, that will
return the data
                        System.Threading.Monitor.PulseAll(_lock);
280                     Console.WriteLine(System.Threading.Thread
.CurrentThread.ManagedThreadId + " is done!");
                    }
282             }
            }
284     }


286


288     /**
         * Get the strokes from one page
290      */
        public String getPage(string guid, int copy, int page,
bool full)
292     {
            //this.full will determine if we give notes(true) or
ink(false) style output
294         this.full = full;

296         Console.WriteLine(System.Threading.Thread.
CurrentThread.ManagedThreadId + " is waiting!");
            //As long as _flag is not > -1
298         //no new request can be made
            lock (_lock_flag)
300         {

302             while (_flag < 0)
                {
304                 //When an other request is done, it should
pulse
                    //and _flag <0 is reevaluated. After each 15s
 we try anyway
306                 System.Threading.Monitor.Wait(_lock_flag);

308                 Console.WriteLine(System.Threading.Thread.
CurrentThread.ManagedThreadId + " waiting with " + _flag);
```

```
                }
310             _flag--;
                Console.WriteLine(System.Threading.Thread.
    CurrentThread.ManagedThreadId + " going with " + _flag);
312         }

314         lock (_lock)
            {
316             //To be shure the driver doens't act up, we wait
    a minimum of 7seconds between requests
                bool b = true;
318             while (lastPenComm != null && lastPenComm.
    AddSeconds(7) > DateTime.UtcNow)
                {
320
                    //The boolean b ensures we only output the
    message once
322                 if (b) Console.WriteLine("Request for pen by
    thread " + System.Threading.Thread.CurrentThread.
    ManagedThreadId + " succeeds the last one to fast. Waiting...
     ");
                    b = false;
324                 System.Threading.Thread.Sleep(500);//re-try
    every 0.5s

326             }
                //Set the time of this request, as the last
    request
328             lastPenComm = DateTime.UtcNow;

330             //TODO check if there is only one pen! :s

332             //Set the variables the GetPageEvent mothod will
    use
                this.guid = guid;
334             this.page = page;
                this.copy = copy;
336             //happy fun time...
                if (smartpens != null) smartpens.Destroy();
338             smartpens = null;
                if (smartpens == null) smartpens = new Smartpens(
    true, PENCOMM_LOG);
340             //Create a new event, using the method '
    GetPageEvent' found in this class
                var cmm = new Smartpens.SmartpenChangeCallback(
    GetPageEvent);
342
                //Attach the event to the smatpens
344             smartpens.SmartpenAttachNormalEvent += cmm;
```

```
                //Find pen and make shure the attached event is
    executed
346             smartpens.Find();
                //Wait until the result is received
348             System.Threading.Monitor.Wait(_lock, 20000);
                }
350                 Console.WriteLine(System.Threading.Thread.
    CurrentThread.ManagedThreadId + "is unlocked!");

352
            // Collect the results
354         string finalString = this.result + "";

356         lock (_lock_flag)
            {
358             //Put the flag back
                _flag++;
360             //Pulse waiting _lock_flag waits
                System.Threading.Monitor.PulseAll(_lock_flag);
362         }

364         //Return the results
            return finalString;
366


368


370     }

372     private void GetPageEvent(Smartpen pen)
        {
374
            Console.WriteLine(System.Threading.Thread.
    CurrentThread.ManagedThreadId + "is going to read!");
376
            pen.Hardware.Update();
378         //Set up a buffer for the result
            StringBuilder stringBuilder = new StringBuilder();
380         //Keep track of the amount of strokes
            int counter = 0;
382         //object te keep the open Document/Container of the
    book/collection
            Document doc = null ;
384
                try
386             {

388                 Console.WriteLine("guid: " + this.guid);
                    //Get data from pen and store locally
```

```
390                     pen.DataGet(this.guid,0  , DOC_DATA_FILE);
                        //Open the locally saved file for use
392                     doc = new Container(DOC_DATA_FILE).Document;
                        Console.WriteLine("page: " + this.page + "/
    copy:" + this.copy);
394                     //Get page by page and copy number
                        PatternPage patternPage = doc.GetPatternPage(
    this.page, this.copy);
396


398


400                     //TODO make method for page deletion? tip?
                        //Console.WriteLine(doc.LSDocument.
    DeletePARange(doc.LSDocument.GetPAFromPageAndCopy(this.page,
    this.copy)));
402
                        //Collect all stokes one by one and format
    them into the String buffer
404                     foreach (StrokeOwner penData in patternPage)
                        {
406
                            foreach (KeyValuePair<long, Stroke>
    keyPair in penData)
408                         {

410                             //long creationTime = keyPair.Key;
                                //Get the stroke
412                             Stroke stroke = keyPair.Value;
                                Shape shape = stroke;
414                             //Get the points and thus coordinates
                                System.Drawing.Point[] points = shape
    .GetVertices();
416

418                             //Generate the textual stroke
                                counter++;
420                             if (this.full)
                                {
422                                 stringBuilder.Append(strokeMarker
    );
                                    stringBuilder.Append(newLine);
424                             }
                                stringBuilder.Append(points.Length);
426                             stringBuilder.Append(newLine);

428                             foreach (System.Drawing.Point coord
    in points)
                                {
```

```
430
                                    stringBuilder.Append(coord.X);
432                                 stringBuilder.Append(" ");
                                    stringBuilder.Append(coord.Y);
434                                 stringBuilder.Append(newLine);

436                             }
                            }
438                     }
                        if (this.full)
440                     {
                            stringBuilder.Append(endMarker);
442                     }
                    }
444                 catch (Exception e) {
                        Console.WriteLine(e.ToString());
446                 }
                    finally
448                 {
                        if (doc != null)
450                     {
                            if (doc.LSDocument != null)
452                         {
                                doc.LSDocument.Close();
454                         }
                            doc.Close();
456
                        }
458
                        StringBuilder finalStringBuilder = new
    StringBuilder();
460                     //Build the top of the String
                        if (this.full)
462                     {
                            finalStringBuilder.Append(beginMarker);
464                     }

466                     finalStringBuilder.Append(counter);
                        finalStringBuilder.Append(newLine);
468                     finalStringBuilder.Append(stringBuilder); //
    Put the stokes under the top
                        //Send the final result this.result
470                     this.result = finalStringBuilder.ToString();
                        lock (_lock)
472                     {
                            //apprise the waiting method the results
    have been collected
474                         System.Threading.Monitor.PulseAll(_lock);
```

```
                    Console.WriteLine(System.Threading.Thread
.CurrentThread.ManagedThreadId + " is done!");
476                 }


478


480
                }
482


484
        }
486
        //Methods to save and load the Dictonary wich keeps info
on the timestamps of the thumbs saved to disk
488     //based upon: http://www.dotnetperls.com/dictionary-
binary

490     static void Write(Dictionary<string, long> dictionary,
string file)
        {
492       using (FileStream fs = File.OpenWrite(file))
          using (BinaryWriter writer = new BinaryWriter(fs))
494
          {
496           // Put count.
              writer.Write(dictionary.Count);
498
              // Write pairs.
500           foreach (var pair in dictionary)
              {
502             writer.Write(pair.Key);
                writer.Write(pair.Value);
504
              }
506
          }
508
        }
510
        static Dictionary<string, long> Read(string file)
512     {

514       var result = new Dictionary<string, long>();
          using (FileStream fs = File.OpenRead(file))
516       using (BinaryReader reader = new BinaryReader(fs))

518       {
```

```
520                 // Get count.
                    int count = reader.ReadInt32();
522
                    // Read in all pairs.
524                 for (int i = 0; i < count; i++)
                    {
526
                      string key = reader.ReadString();
528                   long value = reader.ReadInt64();
                      result[key] = value;
530
                    }
532
                }
534
            return result;
536
              }
538


540


542

544         }

546

548 }
```

MainWindow.cs

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

/**
 * MainWindow - Setup webserver and process requests
 * By: Tania Van Denhouwe
 * Lightly based upon: https://codehosting.net/blog/BlogEngine/post/Simple-C-Web-Server.aspx (MIT license)
 */

namespace Draw2Model
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {


        PenExtractor penExtractor;
        private int port = 5050;
        private string webRoot = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\d2m\\WebRoot";
        private Dictionary<string, string> mime = new Dictionary<string, string>();
        private String url = "";
        public MainWindow()
        {


            if (!File.Exists(this.webRoot+"\\index.html"))
```

```
44              {
                    Console.WriteLine("Duplicating webroot for user")
    ;
46                  DirectoryCopy(AppDomain.CurrentDomain.
    BaseDirectory+"WebRoot", this.webRoot, true);
                }
48              else
                {
50                  Console.WriteLine("Webroot already exists for
    user. Skipping copy.");
                }
52

54              Console.WriteLine(Environment.GetFolderPath(
    Environment.SpecialFolder.ApplicationData));
                penExtractor = new PenExtractor(webRoot);
56

                InitializeComponent();
58              WindowState = WindowState.Minimized;
                gobtn.IsEnabled = false;
60

                //Set the used MIME types
62              mime["JS"] = "application/x-javascript";
                mime["PNG"] = "image/png";
64              mime["HTML"] = "text/html";
                mime["CSS"] = "text/css";
66

                bool httpOK = false;
68              while (!httpOK)

70                  try
                    {
72                      url = "http://localhost:" + port + "/";
                        WebServer ws = new WebServer(SendResponse,
    url);
74                      ws.Run();

76                      Console.WriteLine("A simple webserver. Press
    a key to quit.");
                        //Console.ReadKey();
78                      //ws.Stop();
                        httpOK = true;
80                  }
                    catch (HttpListenerException e)
82                  {
                        port++;
84                  }

86              System.Diagnostics.Process.Start(this.url);
```

```
88              this.info.Content = "Running on " + url;
                gobtn.IsEnabled = true;
90          }

92          public SimpleResponse SendResponse(HttpListenerRequest
        request)
            {
94

96


98
                Console.WriteLine("REQUEST FOR: " + webRoot + request
        .RawUrl);
100             string searchFile = request.RawUrl;
                if (request.RawUrl.Equals("/")) searchFile = "/index.
        html";
102             if (File.Exists(webRoot + searchFile))
                {
104
                    String type = "";
106                 String ext = System.IO.Path.GetExtension(webRoot
        + searchFile).ToUpper().Replace(".","");
                    if (mime.ContainsKey(ext))
108                 {
                        type = mime[ext];
110                 }

112
                    return new SimpleResponse(type, File.ReadAllBytes
        (webRoot + searchFile));
114             }

116             return this.apiResponse(request);

118
            }
120
            private SimpleResponse apiResponse(HttpListenerRequest
        request)
122         {

124             string[] args = request.RawUrl.Split("/".ToCharArray
        ());
                try
126             {
                    if (!args[1].Equals("api")) return new
        SimpleResponse("", "File or API not found");
```

```
128                 if (!args[2].Equals("page")) return new
    SimpleResponse("application/json", penExtractor.
    getAllPageInfo());
                    if (args[3].Equals("ink"))
130                 {
                        try
132                     {
                            return new SimpleResponse("application/
    octet-stream", penExtractor.getPage(args[4], Convert.ToInt32(
    args[5]), Convert.ToInt32(args[6].Replace(".ink","")), false)
    );
134                     }
                        catch (FormatException e)
136                     {
                            return new SimpleResponse("", "File or
    API not found");
138                     }

140                 }
                    try
142                 {
                        return new SimpleResponse("text/plain",
    penExtractor.getPage(args[3], Convert.ToInt32(args[4]),
    Convert.ToInt32(args[5]), true));
144                 }
                    catch (FormatException e)
146                 {
                        return new SimpleResponse("", "File or API
    not found");
148                 }
                    catch (OverflowException e)
150                 {
                        return new SimpleResponse("", "File or API
    not found");
152                 }
            }
154
            catch (System.IndexOutOfRangeException e)
156         {
                return new SimpleResponse("", "File or API not
    found");
158


160
            }
162
        }
164
```

```
        private void Button_Click(object sender, RoutedEventArgs
    e)
166     {
            System.Diagnostics.Process.Start(this.url);
168     }

170     private void DirectoryCopy(string sourceDirName, string
    destDirName, bool copySubDirs)
        {
172         // Get the subdirectories for the specified directory
    .
            DirectoryInfo dir = new DirectoryInfo(sourceDirName);
174         DirectoryInfo[] dirs = dir.GetDirectories();

176         if (!dir.Exists)
            {
178             throw new DirectoryNotFoundException(
                    "Source directory does not exist or could not
     be found: "
180                 + sourceDirName);
            }
182
            // If the destination directory doesn't exist, create
     it.
184         if (!Directory.Exists(destDirName))
            {
186             Directory.CreateDirectory(destDirName);
            }
188
            // Get the files in the directory and copy them to
    the new location.
190         FileInfo[] files = dir.GetFiles();
            foreach (FileInfo file in files)
192         {
                string temppath = System.IO.Path.Combine(
    destDirName, file.Name);
194             file.CopyTo(temppath, true);
            }
196
            // If copying subdirectories, copy them and their
    contents to new location.
198         if (copySubDirs)
            {
200             foreach (DirectoryInfo subdir in dirs)
                {
202                 string temppath = System.IO.Path.Combine(
    destDirName, subdir.Name);
                    DirectoryCopy(subdir.FullName, temppath,
    copySubDirs);
```

```
204                         }
                        }
206                 }
            }
208 }
```

SimpleResponse.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Draw2Model
{
    public class SimpleResponse
    {
        public string content { get; set; }
        public byte[] byteContent { get; set; }
        public string contentType { get; set; }

        public SimpleResponse(string contentType, string content)
        {
            this.content = content;
            this.contentType = contentType;

        }
        public SimpleResponse(string contentType, byte[] byteContent)
        {
            this.byteContent = byteContent;
            this.contentType = contentType;

        }

        public byte[] getContent()
        {
            if (this.content != null) return Encoding.UTF8.GetBytes(this.content);
            return this.byteContent;

        }
    }
}
```

A.4 Web Application Code

Index.html

```
<html>

<head>
    <title>Draw2Model</title>

    <link rel="stylesheet" href="jquery-ui.css">

    <script type="text/javascript" src="http://code.jquery.com/
    jquery-2.1.4.min.js"></script>
    <script language="javascript" type="text/javascript" src="
    http://code.jquery.com/ui/1.11.3/jquery-ui.min.js"></script>
    <script type="text/javascript" src="dist/jstree.min.js"></
    script>
    <script language="javascript" type="text/javascript" src="
    XMLWriter.js"></script>
    <script language="javascript" type="text/javascript" src="
    helpFunctions.js"></script>
    <script language="javascript" type="text/javascript" src="
    encloseXML.js"></script>
    <script language="javascript" type="text/javascript" src="
    objects.js"></script>
    <script language="javascript" type="text/javascript" src="
    recognition.js"></script>
    <script language="javascript" type="text/javascript" src="
    convertToJson.js"></script>
    <script language="javascript" type="text/javascript" src="
    groupInObjects.js"></script>


    <link rel="stylesheet" href="dist/themes/default/style.min.
    css" />
    <style>
        div.box {
        }

        div.div1 {
            float: left;
            width: 45%;
            max-width: 45%;
            height: 450px;
            max-height: 450px;
            overflow-x: hidden;
            overflow-y: scroll;
        }

```

```
        div.div2 {
            float: left;
            width: 53%;
            max-width: 53%;
            height: 450px;
            max-height: 450px;
            overflow-x: hidden;
            overflow-y: scroll;
        }
    </style>
</head>

<body>
    <img src="loading.gif" alt="loading" style="visibility:
    collapse" />


    <script type="text/javascript">

        $(document).ready(function () {

            $.get("/api/pages", function (data) {

                $("#optionBox").show();
                books = [];

                for (i = 0; i < data.length; i++) {
                    books[i] = { id: data[i].Guid, text: data[i].
    Title, icon: "book.png", children: [] };

                    for (j = 0; j < data[i].Pages.length; j++) {


                        if (books[i].children[data[i].Pages[j].
    Copy] == undefined) {
                            books[i].children[data[i].Pages[j].
    Copy] = { id: data[i].Guid + "/" + data[i].Pages[j].Copy,
    icon: "book.png", text: "Copy " + data[i].Pages[j].Copy,
    children: [] };

                        }
                        books[i].children[data[i].Pages[j].Copy].
    children.push({ id: data[i].Guid + "/" + data[i].Pages[j].
    Copy + "/" + data[i].Pages[j].Page, text: "Page " + data[i].
    Pages[j].Page, icon: "sketch.png", children: [] })


```

```
76
                    }
78
                }
80
82              $('#jstree_demo_div').jstree({
                    'core': {
84                      "animation": 0,
                        'data': books,
86
                        "themes": { "stripes": true },
88                      "multiple": false
                    },
90                  "plugins": [
                      "contextmenu", "dnd", "search",
92                    "state", "types", "wholerow"
                    ]
94              });

96              $('#jstree_demo_div').on('select_node.jstree',
    function (node, selected) {

98                  if ((selected.selected[0].match(new RegExp("/
    ", "g")) || []).length == 2) {

100                     $("#thumb").attr("src", "thumbs/" +
    selected.selected[0] + ".png");

102                 } else {

104                     $("#thumb").attr("src", "dotpaper.png");

106                 }


108


110             });


112


114         });

116
        });

118
    </script>
120
```

```
122
    <script>
124         jQuery.fn.extend({
                disable: function (state) {
126                 return this.each(function () {
                        this.disabled = state;
128                 });
                }
130         });


132
            $(function () {
134             $("#dialog").dialog({ modal: true, hide:true });
                document.getElementById("drawFrame").onload =
    function() {
136                 $("#dialog").dialog({ modal: true, hide: false,
    resizable: false, position: { my: "center", at: "center", of:
     window }, height: 555, maxHeight: 555, width: 800, maxWidth:
     800 });
                };
138             $('#loadBtn').click(function () {
                    timeStart = new Date();
140                 // displayContents("empty");
                    url = $('#jstree_demo_div').jstree('get_selected'
    )+"";
142                 if((url.match(new RegExp("/", "g")) || []).length
     == 2) {

144                     $('#loadBtn').disable(true);
                        $('#loadBtn').html("Please wait...")
146
                        $('#langBox').disable(true);
148


150                     $('#jstree_demo_div').disable(true);

152                     $.get("/api/page/" + url, function (data) {
                            displayContents(data,$("#langBox").val())
    ;
154                     });
                    }
156


158


160             })
            });
162     </script>
```

```
164
    <div id="dialog" title="Load scheme from pen">
166
        <div class="box">
168
            <div id="optionBox" style="display:none;">
170             <select id="langBox">
                    <option value="af">Afrikaans ? South Africa</
    option>
172                 <option value="sq">Albanian ? Albania</option
    >
                    <option value="ar">Arabic (no specific locale
    )</option>
174                 <option value ="as">Armenian ? Armenia</
    optiovalue>
                    <option value="az">Azeri ? Azerbaijan</option
    >
176                 <option value="eu">Basque ? Spain</option>
                    <option value ="be">Belarusian ? Belarus</
    optionvalue>
178                 <option value="bg">Bulgarian ? Bulgaria</
    option>
                    <option value="ca">Catalan ? Spain</option>
180                 <option value="zh_CN">Chinese (Simplified) ?
    PRC</option>
                    <option value="zh_HK">Chinese (Traditional) ?
     Hong Kong</option>
182                 <option value="zh_TW">Chinese (Traditional) ?
     Taiwan</option>
                    <option value="hr">Croatian ? Croatia</option
    >
184                 <option value="cs">Czech ? Czech Republic</
    option>
                    <option value="da">Danish ? Denmark</option>
186                 <option value="nl_BE" selected>Dutch ?
    Belgium</option>
                    <option value="nl_NL">Dutch ? Netherlands</
    option>
188                 <option value="en_CA">English ? Canada</
    option>
                    <option value="en_GB">English ? United
    Kingdom</option>
190                 <option value="en_US" >English ? USA</option>
                    <option value="et">Estonian ? Estonia</option
    >
192                 <option value="fa">Farsi ? Iran</option>
                    <option value="fi">Finnish ? Finland</option>
```

```
194                     <option value="fr_CA">French ? Canada</option
    >
                        <option value="fr_FR">French ? France</option
    >
196                     <option value="ga">Galician ? Spain</option>
                        <option value="ka">Georgian ? Georgia</option
    >
198                     <option value="de_AT">German ? Austria</
    option>
                        <option value="de_DE">German ? Germany</
    option>
200                     <option value="el">Greek ? Greece</option>
                        <option value="he">Hebrew ? Israel</option>
202                     <option value="hi">Hindi ? India</option>
                        <option value="hu">Hungarian ? Hungary</
    option>
204                     <option value="is">Icelandic ? Iceland</
    option>
                        <option value="id">Indonesian ? Indonesia</
    option>
206                     <option value="ga">Irish ? Ireland</option>
                        <option value="it">Italian ? Italy</option>
208                     <option value="ja">Japanese ? Japan</option>
                        <option value="kk">Kazakh ? Kazakhstan</
    option>
210                     <option value="ko">Korean ? Korea</option>
                        <option value="lv">Latvian ? Latvia</option>
212                     <option value="lt">Lithuanian ? Lithuania</
    option>
                        <option value="mk">Macedonian ? Macedonia</
    option>
214                     <option value="ms">Malay ? Malaysia</option>
                        <option value="mn">Mongolian ? Mongolia</
    option>
216                     <option value="no">Norwegian ? Norway</option
    >
                        <option value="pl">Polish ? Poland</option>
218                     <option value="pt_BR">Portuguese ? Brazil</
    option>
                        <option value="pt">Portuguese ? Portugal</
    option>
220                     <option value="ro">Romanian ? Romania</option
    >
                        <option value="ru">Russian ? Russia</option>
222                     <option value="sr">Serbian (Cyrillic) ?
    Serbia</option>
                        <option value="sr">Serbian (Latin) ? Serbia</
    option>
224                     <option value="sk">Slovak ? Slovakia</option>
```

```
                    <option value="sl">Slovenian ? Slovenia</
    option>
226                 <option value="es_MX">Spanish ? Mexico</
    option>
                    <option value="es">Spanish ? Spain</option>
228                 <option value="sv_SV">Swedish ? Sweden</
    option>
                    <option value="tt">Tatar ? Russia</option>
230                 <option value="th">Thai ? Thailand</option>
                    <option value="tr">Turkish ? Turkey</option>
232                 <option value="uk">Ukrainian ? Ukraine</
    option>
                    <option value="ur">Urdu ? Pakistan</option>
234                 <option value="vi">Vietnamese ? Vietnam</
    option>
                </select>
236

238             <button id="loadBtn">Convert to scheme</button>
                <input type="checkbox" id="xml" name="xml" value=
    "ORMdoc">download xml<br>
240             <!--<input type="checkbox" name="image" value="
    ORMdocImage">download xml file -->
            </div>
242

            <div class="div1"><div id="jstree_demo_div">Loading
    pages...<br/> This may take a while!</div></div>
244         <div class="div2">

246             <img id="thumb" src="loading.gif" />
            </div>
248     </div>


250

    </div>
252

    <iframe id="drawFrame" name="drawFrame" height="500" width="
    500" style="position:fixed; top:0px; bottom:0px; left: 0px;
    right:0px; width:100%; height:100%; border:none; margin:0;
    padding:0; overflow:hidden;"></iframe>
254  <script>
        var editWindow = window.open('https://www.draw.io/?embed
    =1', 'drawFrame');
256  </script>

258 </body>

260
```

```
</html>
```

ConvertToJson.js

```
var theStks;
var strokes = [];
var shapeStrokes = [];
var textStrokes =[];
var shapebox;
var textbox;

 var objects = new Array;
    //bounding boxes of the objects
    var boxes = new Array;
    var subBoxes = new Array;
    var theConnections= new Array;
    theConnections[0]=new Array;
    theConnections[1]=new Array;

    var ids = new Array;


//get the different strokes by searching for the code indicating
    a new stroke
//returns object containing nrOf coordinates per stroke and the
    strokes themselves
function getStrokes(c, code) {
    theStrokes = c.split(code);
    theNrOf = new Array;

    for (i = 0; i < theStrokes.length; i++) {
        newline = theStrokes[i].indexOf("\n");
        theNrOf[i] = theStrokes[i].substring(0, newline);
        theStrokes[i] = theStrokes[i].substring(newline + 1,
    theStrokes[i].length);
    }

  //console.log(theStrokes);
    return {
        number: theNrOf,
        strokes: theStrokes
    }
}


// divide the stroke in coordinates and put them in the correct
    object form
function getStrokeCoord(substroke) {
```

```
41      var substrokearray = substroke.split("\n");
        var xcords = [];
43      var ycords = [];

45      for (var i = 1; i < substrokearray.length; i++) {
            var subsubstroke = substrokearray[i].split(" ");
47          if (subsubstroke[0] != undefined && subsubstroke[0] != ""
        && subsubstroke[1] != undefined && subsubstroke[1] != "") {
                xcords.push(subsubstroke[0]);
49              ycords.push(subsubstroke[1]);
            }
51

        }
53

        return {
55          "type": "stroke",
            x: xcords,
57          y: ycords
        };
59

    }
61

    // push strokes on the correct stroke object  //TODO push on
        right strokes
63  function processStrokes(thestrokes) {

65    for (i = 1; i< thestrokes.length; i++) {
          strokes.push(getStrokeCoord(thestrokes[i]))
67      }

69  }

71  function process(contents) {

73      // trim away unnessecary information
        var c = contents.toString();
75      // get each newline
        var newlines = c.split("\n");
77

79      //get the code that anounces new stroke
        code = newlines[4];
81


83
        //divide the document in strokes
85      theStks = getStrokes(c, code).strokes;

87
```

```
    //create a stroke object
89  processStrokes(theStks);

91

93  return {
      strokes: strokes,
95      shapes: shapeStrokes,
        text: textStrokes
97  };

99 }

101

   function displayContents(contents,language) {
103     timeGotInk = new Date();
    process(contents); //process the content
105

107

    function callbackFunction(results) {
109     timeGotRecognition = new Date();
        console.log("---");
111     console.log(results);
        console.log("---");
113   //  displayResult2(strokes, results);
        //console.log(strokes);
115     groupInObjects(results);

117

    }
119  // groupInObjects(result1);

121   //convertToObjects(result16, strokes);
   recognitionCall(strokes, apiKey, language, callbackFunction);
    //TODO
123

   }
```

recognition.js

```
apiKey = "990e5d83-dc9e-4bc3-9174-d770405dc620";
url = "https://cloud.myscript.com/api/myscript/v2.0/analyzer/doSimpleRecognition.json";
/** This function creates the JSON object, sends it and retrieves the result. */
recognitionCall = function (strokes, apiKey, language, callback) {
    console.log(language);
    url = "https://cloud.myscript.com/api/myscript/v2.0/analyzer/doSimpleRecognition.json";
    var jsonPost = {
        "parameter": {
            "hwrParameter": {
                /** Language is a mandatory parameter. */
                "language":language//"nl_NL"
            }
        },
        "components": strokes
    };
    /** Send data to POST. Give your API key as supplied on registration, or the
     * server will not recognize you as a valid user. */
    var data = {
        "apiKey": apiKey,
        "analyzerInput": JSON.stringify(jsonPost)
    };
    /** Post request.  Careful! If there are no candidates, the sample may crash. */
    $.post(url, data, function(jsonResult) {

        callback(jsonResult);
    }, "json").error(function(XMLHttpRequest, textStatus) {});
};
```

GroupInObjects.js

```
function groupInObjects(Json) {
    var groups = Json.result.groups;
    var shapes = Json.result.shapes;
    var tables = Json.result.tables;
    var texts = Json.result.textLines;
    //get different objects
    for (var a = 0; a < groups.length; a++) {
        ids[groups[a].uniqueID] = groups[a];
    }
    for (var a = 0; a < shapes.length; a++) {
        ids[shapes[a].uniqueID] = shapes[a];
    }
    for (var b = 0; b < texts.length; b++) {
        ids[texts[b].uniqueID] = texts[b];
    }
    for (var c = 0; c < tables.length; c++) {
        ids[tables[c].uniqueID] = tables[c];
    }
    createObjects();
    createConnections();
    //0. Iterate over groups to greate the objects
    function createObjects() {
        for (var a = 0; a < groups.length; a++) {
            var group = groups[a]
            var textAndShape = getTextAndShape(group); //get the
    shape and the text from the group
            if (textAndShape != undefined) {
                var text = textAndShape.theText;
                var shape = textAndShape.theShape;
                console.log(text + shape);
                if (shape!=undefined&&text!=undefined) {
                    var kind = defineObjectsKind(shape, text); //
    define the kind of object this makes
                    makeTheObjects(kind, shape, text);
                }

            }
        }

        function getTextAndShape(group) {
            if (group.type == "TEXT_INSIDE_GRAPHICS") { //create
    function get shape & text
                var text = "";
                var shape = null;
                for (var b = 0; b < group.elementReferences.
    length; b++) {
                    var element = group.elementReferences[b];
```

```
                    var referedElement = ids[element.uniqueID];
45                  if (element.type == "textLine") {
                        text += referedElement.result.
    textSegmentResult.candidates[0].label;
47                  } else {
                        if (element.type == "shape") {
49                          shape = referedElement.candidates[0];
                            if (shape.label == "polygon" &&
    referedElement.candidates[1]!=undefined) {
51                              shape = referedElement.candidates
    [1];
                            }
53                      } else {
                            if (element.type == "group" &&
    referedElement.type == "PARAGRAPH") {
55                              for (var c = 0; c <
    referedElement.elementReferences.length; c++) {
                                var reference =
    referedElement.elementReferences[c].uniqueID;
57                              if (text != "") {
                                    text += "&lt;div&gt;" +
    ids[reference].result.textSegmentResult.candidates[0].label +
     "&lt;/div&gt;";
59                              } else {
                                    text += ids[reference].
    result.textSegmentResult.candidates[0].label;
61                              }
                            }
63                      }
                    }
65                  }
                }
67              return {
                    theText: text,
69                  theShape: shape
                };
71          }
        };
73
        function defineObjectsKind(shape, text) {
75          var kind = 9;

77                  if (shape.label == "circle" || shape.label ==
     "ellipse") { //if it's an ellipse/circle->type or
    constraint
                if (text.toLowerCase() == "ex" || text.
    toLowerCase() == "er") {
79                  kind = 0
                } //
```

```
81                 else {
                       if (text.toLowerCase() == "eq" || text.
    toLowerCase() == "eg") {
83                         kind = 1
                       } //equallity constraint
85                     else {
                           if (text.toLowerCase() == "sub") {
87                             kind = 2
                           } // subset constraint
89                         else {
                               if (text.toLowerCase() == "ior") {
91                                 kind = 3
                               } //inclusive or constraint
93                             else {
                                   if (text.toLowerCase() == "eor")
    {
95                                     kind = 4
                                   } //exclusive or constraint
97                                 else {
                                       if (text.indexOf("[") >= 0 ||
     text.indexOf("']'") >= 0 || text.indexOf("(") >= 0 || text.
    indexOf(")") >= 0) {
99
                                           kind = 5
101                                    } //value type
                                       else {
103                                        kind = 6 //entity type
                                       }
105                                }
                               }
107                        }
                       }
109                }
               } else { //lazy evaluation if it's a square then this
     bif does not have to be tested
111                if (shape.label == "rectangle" || shape.label ==
    "square" || shape.label == "trapezoid" || shape.label == "
    parallelogram" || shape.label == "rhombus" || shape.label ==
    "quadrilateral") { //if it's a square--> role
                       if (text.indexOf("#") < 0) { //its a unary
    role
113                        kind = 7;
                       } else { //it's an n-airy role
115                        kind = 8
                       }
117                }
                       }
119                 /*   if (text.toLowerCase() == "eq" || text.
    toLowerCase() == "eg") {
```

```
                        kind = 1
121                 } //equallity constraint
                    else {
123                     if (text.toLowerCase() == "sub") {
                            kind = 2
125                     } // subset constraint
                        else {
127                         if (text.toLowerCase() == "ior") {
                                kind = 3
129                         } //inclusive or constraint
                            else {
131                             if (text.toLowerCase() == "eor")
    {
                                    kind = 4
133                             } //exclusive or constraint
                            }
135                     }
                    }*/
137             return kind;
            }
139
            function makeTheObjects(kind, shape, text) {
141             switch (kind) {
                    case 0: //Exclusion
143                     objects[objects.length] = new exclusion(shape
    .primitives[0].center);
                        //objects box with boundaries for crating
    relations (creates a squer of boundaries)
145                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
147                 case 1: //equality
                        objects[objects.length] = new equal(shape.
    primitives[0].center);
149                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
151                 case 2: //sub
                        objects[objects.length] = new subset(shape.
    primitives[0].center);
153                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
155                 case 3: //ior
                        objects[objects.length] = new IOR(shape.
    primitives[0].center);
157                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
```

```
159                 case 4: //eor
                        objects[objects.length] = new EOR(shape.
    primitives[0].center);
161                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
163                 case 5: //value
                        objects[objects.length] = new type(escapeHtml
    (text), {
165                         x: shape.primitives[0].center.x - (shape.
    primitives[0].maxRadius / 2),
                            y: shape.primitives[0].center.y - (shape.
    primitives[0].minRadius / 2)
167                     }, "value");
                        boxes[objects.length - 1] = createEnlargedBox
    (shape);
169                     break;
                    case 6: //entity
171                     objects[objects.length] = new type(escapeHtml
    (text), {
                            x: shape.primitives[0].center.x - (shape.
    primitives[0].maxRadius / 2),
173                         y: shape.primitives[0].center.y - (shape.
    primitives[0].minRadius / 2)
                        }, "entity");
175                     boxes[objects.length - 1] = createEnlargedBox
    (shape);
                        break;
177                 case 7: //single role
                        var rolesArray = new Array;
179                     var inf = defineRoleConstraints(text);
                        var placement = objects.length;
181                     rolesArray[0] = new role(inf.text, inf.arity,
     inf.mandatory);
                        objects[placement] = new groupRoles(
    rolesArray, shape.primitives[0].firstPoint); //add the role
    group
183                     boxes[placement] = {
                            box: [createEnlargedBox(shape)],
185                         roles: rolesArray
                        };
187                     break;
                    case 8: //n-airy role
189                     var rolesText = text.split("#"); //spilt in
    the different roles
                        var rolesArray = new Array;
191                     var placement = objects.length;
                        for (var c = 0; c < rolesText.length; c++) {
    //iterate over the different roletext  and the individual
```

```
    roles
193                         var inf = defineRoleConstraints(rolesText
    [c]);
                            rolesArray[c] = new role(inf.text, inf.
    arity, inf.mandatory);
195                     }
                        objects[placement] = new groupRoles(
    rolesArray, shape.primitives[0].firstPoint); //add the role
    group
197                     boxes[placement] = {
                            box: createSubBoxes(createEnlargedBox(
    shape), rolesArray),
199                         roles: rolesArray
                        };
201                     break;
                default:
203                     console.log("kind not detected");
            }
205         }

207         function defineRoleConstraints(theText) {
                var mand = theText.indexOf("!") >= 0 || theText.
    indexOf("?") >= 0;
209             var ar = theText.indexOf("@") >= 0;
                var newesttext = escapeHtml(theText);
211             return {
                    arity: ar,
213                 mandatory: mand,
                    text: newesttext
215             }
            }
217     };
        //1.Get the relation !! add mandatoryness
219     function createConnections() {
            var connections = getTheConnections();
221         theConnections[0].length = connections.length;
            theConnections[1].length = connections.length;
223
            matchRelations(connections,objects, boxes);
225         makeConnections(theConnections,connections);
        }
227
        function getTheConnections() {
229         var connections = new Array;
            for (var i = 0; i < shapes.length; i++) {
231             if (shapes[i].candidates.length != 0) {
                    var theShape = shapes[i].candidates[0].label
233                 if (theShape == "arrow" || theShape == "
    curvedArrow" || theShape == "polyline" || theShape == "line")
```

```
    {
                    var startp = shapes[i].candidates[0].
primitives[0].firstPoint
                    var endp = shapes[i].candidates[0].primitives
[0].lastPoint
                    var startdec = shapes[i].candidates[0].
primitives[0].beginDecoration
                    var enddec = shapes[i].candidates[0].
primitives[0].endDecoration //test if triangle counts
                    connections[connections.length] = {
                        start: startp,
                        end: endp,
                        sdec: startdec,
                        edec: enddec
                    }
                }
            }

        }
        return connections;
    }
    //2.bind the relations
    function testIfConnection(box, line, object) {
        var start = line.start;
        var end = line.end;
        var arrowh = "none"
        var theObject = undefined;
        var position = 0; //position 0 is start position position
1 is end position
        if (coordinateInBox(box, start)) {
            theObject = object;
            position = 0;
            if (line.sdec != "NONE") {
                arrowh = "block";
            }
        } else {
            if (coordinateInBox(box, end)) {
                theObject = object;
                position = 1;
                if (line.edec != "NONE") {
                    arrowh = "block";
                }
            }
        }
        return {
            object: theObject,
            pos: position,
            arrow: arrowh
        };
```

```
277     }

279     function matchRelations(lines, objects, boxes) {
            var information;
281         for (var i = 0; i < lines.length; i++) { //iterate over
        the lines
                // console.log(objects);
283             for (var j = 0; j < objects.length; j++) { //iterate
        over the objects to see if there is a match
                    currentObject = objects[j];
285                 if (currentObject.kind == "roleGroup") { //the
        line is connected to a normal object or a role group
                        // console.log(boxes[j]);
287                     for (var k = 0; k < boxes[j].box.length; k++)
         {
                            var role= boxes[j].roles[k];
289                         information = testIfConnection(boxes[j].
        box[k], lines[i], role);

291                         if(role.mandatory){
                                information.arrow="oval";
293                         }
                        }
295                 } else {
                        information = testIfConnection(boxes[j],
        lines[i], currentObject);
297                 }
                    //console.log(information);
299                 if (information.object != undefined) { //a match
        has been found

301                     theConnections[information.pos][i] =
        information; //put the information in the starto or end array
         on the place of the line
                    }
303             }
            }
305     }

307     function makeConnections(connections,lines) {

309         var starts = connections[0];
            var ends = connections[1];
311         for (var i = 0; i < starts.length; i++) {
                if (starts[i] != undefined && ends[i] != undefined) {
         //match for connection at both ends
313                 var start = starts[i];
                    var end = ends[i];
```

```
315                 objects[objects.length] = new Connection(start.
    object, end.object, start.arrow, end.arrow);
            }
317         /*if (ends[i] == undefined && starts[i] != undefined) {
     //match for connection at both ends
                var start = starts[i];
319             var end = lines[i].end;
                console.log(end);
321             objects[objects.length] = new LooseConnection({ x
    : end.x, y: end.y }, start.object, 0, start.arrow);
     ;
323         }

325         if (starts[i] == undefined && ends[i] != undefined) {
     //match for connection at both ends
                var start = lines[i].start;
327             console.log(start);
                var end = ends[i];
329
                objects[objects.length] = new LooseConnection({ x
    : start.x, y: start.y }, end.object, 1, end.arrow);
331
            }*/
333     }
     }
335  var timeGotTransformation = new Date();

337  // create the xml of the objects
     encloseXML(objects);
339  var timeGotXML = new Date();
     console.log(timeStart);
341  console.log(timeGotInk);
     console.log(timeGotRecognition);
343
     console.log("Time to get Ink: " + (timeGotInk - timeStart));
345  console.log("Time to get recignition: " + (timeGotRecognition
    -timeGotInk));
     console.log("Time to group objects: " + (
    timeGotTransformation - timeGotRecognition));
347  console.log("Time tocreate XML: " + (timeGotXML-
    timeGotTransformation));

349 }
  // connection none and none or none and oval, inheritance nona
     and arrow
```

objects.js

```
// JavaScript source code
var idAttribuation = new Id;
var scale = 4;


function type(name, position, kind) {
    this.name = name;
    this.position = position;
    this.kind = kind;
    this.id = idAttribuation.getID();
    this.addXML = function (doc) {
        doc.writeStartElement("mxCell");
        doc.writeAttributeString("id", this.id);
        doc.writeAttributeString("value", this.name);
        if (kind == "entity") {
            doc.writeAttributeString("style", "rounded=1;whiteSpace=wrap;html=1;strokeColor=#aaaaaa;strokeWidth=2;fontColor=#aaaaaa");
        } else {
            doc.writeAttributeString("style", "rounded=1;whiteSpace=wrap;html=1;dashed=1;strokeColor=#aaaaaa;strokeWidth=2;fontColor=#aaaaaa;perimeterSpacing=1");
        }
        doc.writeAttributeString("vertex", "1");
        doc.writeAttributeString("parent", "1");
        doc.writeStartElement("mxGeometry");
        doc.writeAttributeString("x", this.position.x/scale);
        doc.writeAttributeString("y", this.position.y/scale);
        doc.writeAttributeString("width", "120");
        doc.writeAttributeString("height", "60");
        doc.writeAttributeString("as", "geometry");
        doc.writeEndElement();
        doc.writeEndElement();
    }
};

function role(roleName, unary, mandatory) {
    this.id = idAttribuation.getID();
    this.parentID = 1;
    this.offset = 0;
    this.unary = unary;
    this.mandatory = mandatory;
    this.roleName = roleName;
    this.kind = "role";
    this.addXML = function (doc) {
        doc.writeStartElement("mxCell");
        doc.writeAttributeString("id", this.id);
```

```
44          if (!unary) {
                if (roleName == '') {
46                  doc.writeAttributeString("value", "&lt;u&gt;" +
    this.roleName + ";&amp;nbsp;&lt;/u&gt;");
                    doc.writeAttributeString("value", "&lt;u&gt;&amp;
    nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&lt;/u&gt;");
48              }
                else { doc.writeAttributeString("value", "&lt;u&gt;"
    + this.roleName + "&amp;nbsp;&lt;/u&gt;"); }
50
            }
52          else {
                doc.writeAttributeString("value", this.roleName);
54          }
            if (mandatory) {
56              doc.writeAttributeString("style", "whiteSpace=wrap;
    html=1;strokeColor=#aaaaaa;strokeWidth=2;fontColor=#FFFFFF;
    fillColor=#34a7c1");
            } else {
58              doc.writeAttributeString("style", "whiteSpace=wrap;
    html=1;strokeColor=#aaaaaa;strokeWidth=2;fontColor=#aaaaaa");
            }
60          doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("parent", this.parentID);
62          doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("x", this.offset);//this.offset)
    ;
64          doc.writeAttributeString("width", "60");
            doc.writeAttributeString("height", "30");
66          doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
68          doc.writeEndElement();
        }
70  };

72  function groupRoles(roles,position) {
        this.id = idAttribuation.getID();
74      this.position = position;
        this.roles = roles;
76      this.kind = "roleGroup";
        nrOfRoles = roles.length;
78      for (var i = 0; i < roles.length; i++) {
            roles[i].parentID = this.id;
80          roles[i].offset = 60 * i;
        };
82      this.addXML = function (doc) {
            //the group
84          doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", this.id);
```

```
86          doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "group");
88          doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
90          doc.writeAttributeString("parent", "1");
            doc.writeStartElement("mxGeometry");
92          doc.writeAttributeString("x", this.position.x/scale);
            doc.writeAttributeString("y", this.position.y/scale);
94          doc.writeAttributeString("width", 60 * nrOfRoles);
            doc.writeAttributeString("height", "30");
96          doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
98          doc.writeEndElement();
            //the roles
100         for (var i = 0; i < roles.length; i++) {
               roles[i].addXML(doc);
102         };

104     }
    };
106

108 function exclusion(position) {
        this.id = idAttribuation.getID();
110     this.x = position.x;
        this.y = position.y;
112     this.kind = "exclusion constraint";
        this.addXML = function (doc) {
114         doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", this.id);
116         doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "shape=sumEllipse;
      perimeter=ellipsePerimeter;whiteSpace=wrap;html=1;strokeColor
      =#FFFFFF;strokeWidth=2;fillColor=#00a9c3");
118         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
120         doc.writeAttributeString("parent", "1");
            doc.writeStartElement("mxGeometry");
122         doc.writeAttributeString("x", this.x/scale);
            doc.writeAttributeString("y", this.y/scale);
124         doc.writeAttributeString("width", "30");
            doc.writeAttributeString("height", "30");
126         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
128         doc.writeEndElement();
        }
130 }

132 function EOR(position) {
```

```
    this.id = idAttribuation.getID();
134 this.x = position.x;
    this.y = position.y;
136 this.kind = "exclusif or constraint";
    this.addXML = function (doc) {
138     //group
        doc.writeStartElement("mxCell");
140     doc.writeAttributeString("id", this.id);
        doc.writeAttributeString("value", "");
142     doc.writeAttributeString("style", "group");
        doc.writeAttributeString("vertex", "1");
144     doc.writeAttributeString("connectable", "0");
        doc.writeAttributeString("parent", "1");
146     doc.writeStartElement("mxGeometry");
        doc.writeAttributeString("x", this.x/scale);
148     doc.writeAttributeString("y", this.y/scale);
        doc.writeAttributeString("width", "30");
150     doc.writeAttributeString("height", "30");
        doc.writeAttributeString("as", "geometry");
152     doc.writeEndElement();
        doc.writeEndElement();
154     doc.writeStartElement("mxCell");
        doc.writeAttributeString("id", idAttribuation.getID());
156     doc.writeAttributeString("value", "");
        doc.writeAttributeString("style", "shape=sumEllipse;
    perimeter=ellipsePerimeter;whiteSpace=wrap;html=1;strokeColor
    =#FFFFFF;strokeWidth=2;fillColor=#00a9c3");
158     doc.writeAttributeString("vertex", "1");
        doc.writeAttributeString("connectable", "0");
160     doc.writeAttributeString("parent", this.id);
        doc.writeStartElement("mxGeometry");
162     doc.writeAttributeString("width", "30");
        doc.writeAttributeString("height", "30");
164     doc.writeAttributeString("as", "geometry");
        doc.writeEndElement();
166     doc.writeEndElement();
        doc.writeStartElement("mxCell");
168     doc.writeAttributeString("id", idAttribuation.getID());
        doc.writeAttributeString("value", "");
170     doc.writeAttributeString("style", "ellipse;whiteSpace=
    wrap;html=1;strokeColor=#FFFFFF;strokeWidth=2;fillColor=#
    FFFFFF");
        doc.writeAttributeString("vertex", "1");
172     doc.writeAttributeString("connectable", "0");
        doc.writeAttributeString("parent", this.id);
174     doc.writeStartElement("mxGeometry");
        doc.writeAttributeString("x", "10");
176     doc.writeAttributeString("y", "10");
        doc.writeAttributeString("width", "10");
```

```
178             doc.writeAttributeString("height", "10");
                doc.writeAttributeString("as", "geometry");
180             doc.writeEndElement();
                doc.writeEndElement();
182         }
    }
184
    function IOR(position) {
186         this.id = idAttribuation.getID();
            this.x = position.x;
188         this.y = position.y;
            this.kind = "inclusif or constraint";
190         this.addXML = function (doc) {
                //group
192             doc.writeStartElement("mxCell");
                doc.writeAttributeString("id", this.id);
194             doc.writeAttributeString("value", "");
                doc.writeAttributeString("style", "group");
196             doc.writeAttributeString("vertex", "1");
                doc.writeAttributeString("connectable", "0");
198             doc.writeAttributeString("parent", "1");
                doc.writeStartElement("mxGeometry");
200             doc.writeAttributeString("x", this.x/scale);
                doc.writeAttributeString("y", this.y/scale);
202             doc.writeAttributeString("width", "30");
                doc.writeAttributeString("height", "30");
204             doc.writeAttributeString("as", "geometry");
                doc.writeEndElement();
206             doc.writeEndElement();
                doc.writeStartElement("mxCell");
208             doc.writeAttributeString("id", idAttribuation.getID());
                doc.writeAttributeString("value", "");
210             doc.writeAttributeString("style", "ellipse;perimeter=
        ellipsePerimeter;whiteSpace=wrap;html=1;strokeColor=#aaaaaa;
        strokeWidth=2;fillColor=#ffffff");
                doc.writeAttributeString("vertex", "1");
212             doc.writeAttributeString("connectable", "0");
                doc.writeAttributeString("parent", this.id);
214             doc.writeStartElement("mxGeometry");
                doc.writeAttributeString("width", "30");
216             doc.writeAttributeString("height", "30");
                doc.writeAttributeString("as", "geometry");
218             doc.writeEndElement();
                doc.writeEndElement();
220             doc.writeStartElement("mxCell");
                doc.writeAttributeString("id", idAttribuation.getID());
222             doc.writeAttributeString("value", "");
                doc.writeAttributeString("style", "ellipse;whiteSpace=
        wrap;html=1;strokeColor=#aaaaaa;strokeWidth=2;fillColor=#
```

```
    aaaaaa");
224         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
226         doc.writeAttributeString("parent", this.id);
            doc.writeStartElement("mxGeometry");
228         doc.writeAttributeString("x", "10");
            doc.writeAttributeString("y", "10");
230         doc.writeAttributeString("width", "10");
            doc.writeAttributeString("height", "10");
232         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
234         doc.writeEndElement();
    }
236 }

238 function equal(position) {
    this.id = idAttribuation.getID();
240 this.x = position.x;
    this.y = position.y;
242 this.kind = "equallity constraint";
    this.addXML = function (doc) {
244         //group
            doc.writeStartElement("mxCell");
246         doc.writeAttributeString("id", this.id);
            doc.writeAttributeString("value", "");
248         doc.writeAttributeString("style", "group");
            doc.writeAttributeString("vertex", "1");
250         doc.writeAttributeString("connectable", "0");
            doc.writeAttributeString("parent", "1");
252         doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("x", this.x/scale);
254         doc.writeAttributeString("y", this.y/scale);
            doc.writeAttributeString("width", "30");
256         doc.writeAttributeString("height", "30");
            doc.writeAttributeString("as", "geometry");
258         doc.writeEndElement();
            doc.writeEndElement();
260         doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", idAttribuation.getID());
262         doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "ellipse;perimeter=
    ellipsePerimeter;whiteSpace=wrap;html=1;strokeColor=#aaaaaa;
    strokeWidth=2;fillColor=#ffffff");
264         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
266         doc.writeAttributeString("parent", this.id);
            doc.writeStartElement("mxGeometry");
268         doc.writeAttributeString("width", "30");
            doc.writeAttributeString("height", "30");
```

```
270         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
272         doc.writeEndElement();
            doc.writeStartElement("mxCell");
274         doc.writeAttributeString("id", idAttribuation.getID());
            doc.writeAttributeString("value", "");
276         doc.writeAttributeString("style", "line;html=1;
    strokeColor=#aaaaaa;strokeWidth=2;fillColor=none");
            doc.writeAttributeString("vertex", "1");
278         doc.writeAttributeString("connectable", "0");
            doc.writeAttributeString("parent", this.id);
280         doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("x", "8");
282         doc.writeAttributeString("y", "7");
            doc.writeAttributeString("width", "13");
284         doc.writeAttributeString("height", "10");
            doc.writeAttributeString("as", "geometry");
286         doc.writeEndElement();
            doc.writeEndElement();
288         doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", idAttribuation.getID());
290         doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "line;html=1;
    strokeColor=#aaaaaa;strokeWidth=2;fillColor=none");
292         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
294         doc.writeAttributeString("parent", this.id);
            doc.writeStartElement("mxGeometry");
296         doc.writeAttributeString("x", "8");
            doc.writeAttributeString("y", "14");
298         doc.writeAttributeString("width", "13");
            doc.writeAttributeString("height", "10");
300         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
302         doc.writeEndElement();
    }
304 }

306 function subset(position) {
        this.id = idAttribuation.getID();
308     this.x = position.x;
        this.y = position.y;
310     this.kind = "subset constraint";
        this.addXML = function (doc) {
312         //group
            doc.writeStartElement("mxCell");
314         doc.writeAttributeString("id", this.id);
            doc.writeAttributeString("value", "");
316         doc.writeAttributeString("style", "group");
```

```
            doc.writeAttributeString("vertex", "1");
318         doc.writeAttributeString("connectable", "0");
            doc.writeAttributeString("parent", "1");
320         doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("x", this.x/scale);
322         doc.writeAttributeString("y", this.y/scale);
            doc.writeAttributeString("width", "30");
324         doc.writeAttributeString("height", "30");
            doc.writeAttributeString("as", "geometry");
326         doc.writeEndElement();
            doc.writeEndElement();
328         doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", idAttribuation.getID());
330         doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "ellipse;perimeter=
        ellipsePerimeter;whiteSpace=wrap;html=1;strokeColor=#aaaaaa;
        strokeWidth=2;fillColor=#ffffff");
332         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
334         doc.writeAttributeString("parent", this.id);
            doc.writeStartElement("mxGeometry");
336         doc.writeAttributeString("width", "30");
            doc.writeAttributeString("height", "30");
338         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
340         doc.writeEndElement();
            doc.writeStartElement("mxCell");
342         doc.writeAttributeString("id", idAttribuation.getID());
            doc.writeAttributeString("value", "");
344         doc.writeAttributeString("style", "line;html=1;
        strokeColor=#aaaaaa;strokeWidth=2;fillColor=none");
            doc.writeAttributeString("vertex", "1");
346         doc.writeAttributeString("connectable", "0");
            doc.writeAttributeString("parent", this.id);
348         doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("x", "6");
350         doc.writeAttributeString("y", "17");
            doc.writeAttributeString("width", "18");
352         doc.writeAttributeString("height", "10");
            doc.writeAttributeString("as", "geometry");
354         doc.writeEndElement();
            doc.writeEndElement();
356         doc.writeStartElement("mxCell");
            doc.writeAttributeString("id", idAttribuation.getID());
358         doc.writeAttributeString("value", "");
            doc.writeAttributeString("style", "line;html=1;
        strokeColor=#aaaaaa;strokeWidth=2;fillColor=none");
360         doc.writeAttributeString("vertex", "1");
            doc.writeAttributeString("connectable", "0");
```

```
362             doc.writeAttributeString("parent", this.id);
                doc.writeStartElement("mxGeometry");
364             doc.writeAttributeString("x", "6");
                doc.writeAttributeString("y", "13");
366             doc.writeAttributeString("width", "18");
                doc.writeAttributeString("height", "10");
368             doc.writeAttributeString("as", "geometry");
                doc.writeEndElement();
370             doc.writeEndElement();
                doc.writeStartElement("mxCell");
372             doc.writeAttributeString("id", idAttribuation.getID());
                doc.writeAttributeString("value", "");
374             doc.writeAttributeString("style", "line;html=1;
    strokeColor=#aaaaaa;strokeWidth=2;fillColor=none");
                doc.writeAttributeString("vertex", "1");
376             doc.writeAttributeString("connectable", "0");
                doc.writeAttributeString("parent", this.id);
378             doc.writeStartElement("mxGeometry");
                doc.writeAttributeString("x", "6");
380             doc.writeAttributeString("y", "3");
                doc.writeAttributeString("width", "18");
382             doc.writeAttributeString("height", "10");
                doc.writeAttributeString("as", "geometry");
384             doc.writeEndElement();
                doc.writeEndElement();
386             doc.writeStartElement("mxCell");
                doc.writeAttributeString("id", idAttribuation.getID());
388             doc.writeAttributeString("value", "");
                doc.writeAttributeString("style", "line;html=1;
    strokeColor=#aaaaaa;strokeWidth=2;fillColor=none;rotation=90e
    ");
390             doc.writeAttributeString("vertex", "1");
                doc.writeAttributeString("connectable", "0");
392             doc.writeAttributeString("parent", this.id);
                doc.writeStartElement("mxGeometry");
394             doc.writeAttributeString("x", "2");
                doc.writeAttributeString("y", "8");
396             doc.writeAttributeString("width", "10");
                doc.writeAttributeString("height", "10");
398             doc.writeAttributeString("as", "geometry");
                doc.writeEndElement();
400             doc.writeEndElement();
        }
402 }

404 function Connection(object1, object2,mandatory) {
        this.kind = "simple connection";
406     this.firstObject = object1;
        this.secondObject = object2;
```

```
408     this.mandatory = mandatory;
        this.id = idAttribuation.getID();
410     this.addXML = function (doc) {
            doc.writeStartElement("mxCell");
412         doc.writeAttributeString("id", this.id);
            if (mandatory) { doc.writeAttributeString("style", "
    edgeStyle=orthogonalEdgeStyle;rounded=0;html=1;fontColor=#
    aaaaaa;strokeColor=#aaaaaa;strokeWidth=2;endArrow=oval;
    endFill=1;"); }
414         else { doc.writeAttributeString("style", "edgeStyle=
    orthogonalEdgeStyle;rounded=0;html=1;fontColor=#aaaaaa;
    strokeColor=#aaaaaa;strokeWidth=2;endArrow=none;endFill=0");
    }
            doc.writeAttributeString("edge", "1");
416         doc.writeAttributeString("parent", "1");
            doc.writeAttributeString("source", object1.id);
418         doc.writeAttributeString("target", object2.id);
            doc.writeStartElement("mxGeometry");
420         doc.writeAttributeString("relative", "1");
            doc.writeAttributeString("as", "geometry");
422         doc.writeEndElement();
            doc.writeEndElement();
424     }
};
426
function LooseConnection(coordinate, theobject, place,
    decofObject) {
428     this.theObject = theobject;
        this.theCoordinate = coordinate;
430     dec1 = "none";
        dec2 = "none";
432     this.kind = "Loose Connection";
        if (place == 0) {
434         dec1 = decofObject;
        }
436     else {
            dec2 = decofObject;
438     }
        this.id = idAttribuation.getID();
440     this.addXML = function (doc) {
            doc.writeStartElement("mxCell");
442         doc.writeAttributeString("id", this.id);
            doc.writeAttributeString("style", "edgeStyle=
    orthogonalEdgeStyle;rounded=0;html=1;fontColor=#dddddd;
    strokeColor=#dddddd;strokeWidth=2;startArrow="+dec1+";endFill
    =1;endArrow="+dec2+";endFill=1");
444         doc.writeAttributeString("edge", "1");
            doc.writeAttributeString("parent", "1");
446         doc.writeAttributeString("source", this.theObject.id);
```

```
            doc.writeStartElement("mxGeometry");
448         doc.writeAttributeString("relative", "1");
            doc.writeAttributeString("as", "geometry");
450         doc.writeStartElement("mxPoint");
                doc.writeAttributeString("x", "1");
452             doc.writeAttributeString("x", this.theCoordinate.x /
    scale);
                doc.writeAttributeString("y", this.theCoordinate.y /
    scale);
454       doc.writeAttributeString("as", "targetPoint");
                doc.writeEndElement();
456         doc.writeEndElement();
            doc.writeEndElement();
458     }
};
460
function Connection(object1, object2, dec1, dec2) {
462     this.firstObject = object1;
        this.secondObject = object2;
464     this.kind = "a Connection";
        this.id = idAttribuation.getID();
466     this.addXML = function (doc) {
            doc.writeStartElement("mxCell");
468         doc.writeAttributeString("id", this.id);
            doc.writeAttributeString("style", "edgeStyle=
    orthogonalEdgeStyle;rounded=0;html=1;fontColor=#aaaaaa;
    strokeColor=#aaaaaa;strokeWidth=2;startArrow=" + dec1 + ";
    endFill=1;endArrow=" + dec2 + ";endFill=1");
470         doc.writeAttributeString("edge", "1");
            doc.writeAttributeString("parent", "1");
472         doc.writeAttributeString("source", object1.id);
            doc.writeAttributeString("target", object2.id);
474         doc.writeStartElement("mxGeometry");
            doc.writeAttributeString("relative", "1");
476         doc.writeAttributeString("as", "geometry");
            doc.writeEndElement();
478         doc.writeEndElement();
        }
480 };
```

encloseXML.js

```
function sendData(data) {
    if ($("#xml").prop('checked')) {
        var pom = document.createElement('a');
        pom.setAttribute('href', 'data:text/plain;charset=utf-8,'
    + encodeURIComponent(data));
        pom.setAttribute('download', "ORMdoc.xml");
        pom.style.display = 'none';
        document.body.appendChild(pom);
        pom.click();
        document.body.removeChild(pom);
    }
    editWindow.postMessage(data, 'https://www.draw.io');
    console.log($("#xml").prop('checked'));

    $("#dialog").dialog("close");
}

function encloseXML(objects, filename) {
    var doc = new XMLWriter('UTF-8', '1.0');
    doc.writeStartDocument(false);
    // create root doc
    doc.writeStartElement("mxGraphModel");
    doc.writeAttributeString("dx", "1");
    doc.writeAttributeString("dy", "461");
    doc.writeAttributeString("grid", "1");
    doc.writeAttributeString("gridSize", "10");
    doc.writeAttributeString("guides", "1");
    doc.writeAttributeString("tooltips", "1");
    doc.writeAttributeString("connect", "1");
    doc.writeAttributeString("fold", "1");
    doc.writeAttributeString("page", "1");
    doc.writeAttributeString("pageScale", "1");
    doc.writeAttributeString("pageWidth", "826");
    doc.writeAttributeString("pageHeight", "1169");
    doc.writeAttributeString("style", "default-style2");
    doc.writeAttributeString("math", "0");
    doc.writeStartElement("root");
    doc.writeStartElement("mxCell");
    doc.writeAttributeString("id", "0");
    doc.writeEndElement();
    doc.writeStartElement("mxCell");
    doc.writeAttributeString("id", "1");
    doc.writeAttributeString("parent", "0");
    doc.writeEndElement();
    //HERE ALL THE OBJECTS ARE ADDED
    for (var i = 0; i < objects.length; i++) {
        objects[i].addXML(doc);
```

```
    };
48  //TILL HERe WHERE THE DOCUMENT IS CLOSED AND FLUSHED
    doc.writeEndElement();
50  doc.writeEndElement();
    doc.writeEndDocument();
52  console.log(doc.flush());
    sendData(doc.flush());
54 };
```

helpFunctions.js

```
var timeStart;
var timeGotInk;
var timeGotRecognition;
var timeGotTransformation;
var timeGotXML;


//creates the bounding boxes
function createEnlargedBox(shape) {
    //the case of an elliptical shape
    if (shape.label == "circle" || shape.label == "ellipse") {
        centerx = shape.primitives[0].center.x;
        centery = shape.primitives[0].center.y;
        rlength = shape.primitives[0].maxRadius;
        a1 = centerx - (rlength / 2);
        a2 = centery - (rlength / 2);
        b1 = centerx - (rlength / 2);
        b2 = centery + (rlength / 2);
        c1 = centerx + (rlength / 2);
        c2 = centery + (rlength / 2);
        d1 = centery + (rlength / 2);
        d2 = centery - (rlength / 2);
       // console.log(a1 + " " + a2 + " " + b1 + " " + b2 + " " +
    c1 + " " + c2 + " " + d1 + " " + d2);
        return {
            ax: a1,
            ay: a2,
            bx: b1,
            by: b2,
            cx: c1,
            cy: c2,
            dx: d1,
            dy: d2
        };

    }
    //the case of a quadrangle
    if (shape.label == "rectangle" || shape.label == "square" ||
    shape.label == "trapezoid" || shape.label == "parallelogram"
    || shape.label == "rhombus" || shape.label == "quadrilateral"
    ) {
        d1 = shape.primitives[1].firstPoint.x;
        d2 = shape.primitives[1].firstPoint.y;
        c1 = shape.primitives[2].firstPoint.x;
        c2 = shape.primitives[2].firstPoint.y;
        b1 = shape.primitives[3].firstPoint.x;
        b2 = shape.primitives[3].firstPoint.y;
```

```
44          a1 = shape.primitives[0].firstPoint.x;
            a2 = shape.primitives[0].firstPoint.y;
46          return {
                ax: a1,
48              ay: a2,
                bx: b1,
50              by: b2,
                cx: c1,
52              cy: c2,
                dx: d1,
54              dy: d2
            };
56      }
    };
58
    /*function boxDiercetion(box) {
60      var boxHeight = Math.sqrt(Math.pow((box.ax - box.bx), 2) +
        Math.pow((box.ay - box.by), 2));
        var boxLength = Math.sqrt(Math.pow((box.bx - box.cx), 2) +
        Math.pow((box.by - box.cy), 2));
62      if (boxHeight <= boxLength) {
            return "horizontal"
64      } else {
            return "vertical"
66      }
    }*/
68
    function getObjectsWherLabel(textObjects, text) {
70      for (var i = 0; i < textObjects.length; i++) {
            if (textObjects[i].result.textSegmentResult.candidates
        [0].label == text) {
72              return textObjects[i];
                break;
74          }
        }
76  }

78  //create the bounding boxes of the individual role

80  function createSubBoxes(TheBoundingBox, textArray) {
        var nrOfBoxes=textArray.length;
82      var boxes = new Array;
        var boxHeight = Math.sqrt(Math.pow((TheBoundingBox.ax -
        TheBoundingBox.bx), 2) + Math.pow((TheBoundingBox.ay -
        TheBoundingBox.by), 2));
84      var boxLength = Math.sqrt(Math.pow((TheBoundingBox.bx -
        TheBoundingBox.cx), 2) + Math.pow((TheBoundingBox.by -
        TheBoundingBox.cy), 2));
        var distance = boxLength / nrOfBoxes;
```

```
86      for (var i = 0 ; i < nrOfBoxes ; i++) {
            boxes[i] = {
88              ax: TheBoundingBox.ax + (distance * i), //the y
    coordinates stay the same, the x cordinates not
                ay: TheBoundingBox.ay,
90              bx: TheBoundingBox.bx + (distance * i),
                by: TheBoundingBox.by,
92              cx: TheBoundingBox.cy + (distance * (i+1)),
                cy: TheBoundingBox.cy,
94              dx: TheBoundingBox.dx + (distance * (i+1)) ,
                dy: TheBoundingBox.dy
96          }

98      }
        return boxes;
100 }
    /*function createSubBoxes(textArray,textLineObjects) {
102     var nrOfboxes = textArray.lenth;
        var offsetstart=0;
104     var offsetend;


106
        //find # and coordinate where it is
108     for (var i = 0; i < nrOfBoxes; i++) {
            textObject = getObjectsWhereLabel(TextLineObjects,
    textArray[i]);
110         //find offest of #
            firstpoint = textObject.inkRanges[i].firstpoint;
112         stroke = textObject.inkRanges[i].stroke;
            var coord = getCoord(firstpoint, stroke);
114
            offsetend = coord.x;
116         boxes[i] = {
                ax: bigBox.ax + offsetstart, //the y coordinates stay
     the same, the x cordinates not
118             ay: bigBox.ay,
                bx: bigBox.bx + offsetstart,
120             by: bigBox.by,
                cx: offsetend,
122             cy: bigBox.cy,
                dx: offsetend,
124             dy: bigBox.dy
            }
126         offsetstart = coord.x;

128     }


130
    }*/
```

```
132
/*function createSubBoxes(bigBox, text, texts) {
134     var boxes = new Array;
        var offset = 0;
136     textObject = getObjectsWherLabel(texts, text);

138    // direction = boxDiercetion(bigBox);
        for (var i = 0; i < text.length; i++) {
140         if (text[i] == "#") {
                firstpoint = textObject.inkRanges[i].firstpoint;
142             stroke = textObject.inkRanges[i].stroke;
                var coord = getCoord(firstpoint, stroke);
144            // if (direction == "horizontal") {
                    boxes[i] = {
146                     ax: bigBox.ax + offset, //the y coordinates
    stay the same, the x cordinates not
                        ay: bigBox.ay,
148                     bx: bigBox.bx + offset,
                        by: bigBox.by,
150                     cx: coord.x,
                        cy: bigBox.cy,
152                     dx: coord.x,
                        dy: bigBox.dy
154                 }
                    offset = coord.x;
156             //}
                /*else {
158                 boxes[i] = {
                        ax: bigBox.ax, //the x coordinates stay the
    same the y coordinates change
160                     ay: bigBox.ay + offset,
                        bx: bigBox.bx,
162                     by: bigBox.by + offset,
                        cx: bigBox.cx,
164                     cy: coord.y,
                        dx: bigBox.dx,
166                     dy: coord.y
                    }
168                 offset = coord.y;
                }
170         }
        }
172
        return boxes;
174 }*/

176 function getCoord(point, stroke) {
        console.log(strokes);
178     var theStroke = strokes[stroke];
```

```
        return {
180         x: theStroke.x[point],
            y: theStroke.y[point]
182     };
    }
184


186
    function coordinateInBox(box, coordinate) {
188     var margin = 2;

190     miny = Math.min(box.ay, box.by, box.cy, box.dy) - margin;
        minx = Math.min(box.ax, box.bx, box.cx, box.dx) - margin;
192     maxx = Math.max(box.ax, box.bx, box.cx, box.dx) + margin;
        maxy = Math.max(box.ay, box.by, box.cy, box.dy) + margin;
194     x = coordinate.x;
        y = coordinate.y;
196     return (x <= maxx && y <= maxy && x >= minx && y >= miny);
    }
198
    function testbox(box, coord) {
200     ax = box.ax;
        bx = box.bx;
202     dx = box.cx;
        ay = box.ay;
204     by = box.by;
        dy = box.cy;
206     x = coord.x;
        y = coord.y
208     bax = bx - ax
        bay = by - ay
210     dax = dx - ax
        day = dy - ay
212     if ((x - ax) * bax + (y - ay) * bay < 0.0) return false
        if ((x - bx) * bax + (y - by) * bay > 0.0) return false
214     if ((x - ax) * dax + (y - ay) * day < 0.0) return false
        if ((x - dx) * dax + (y - dy) * day > 0.0) return false
216     return true
    };
218 //create Id object that increments automatically each time a the
       id is attribuated
    function Id() {
220     var id = 1;
        this.id = id;
222     this.getID = function getId() {
            this.id = (this.id) + 1;
224         return this.id;
        };
226 }
```

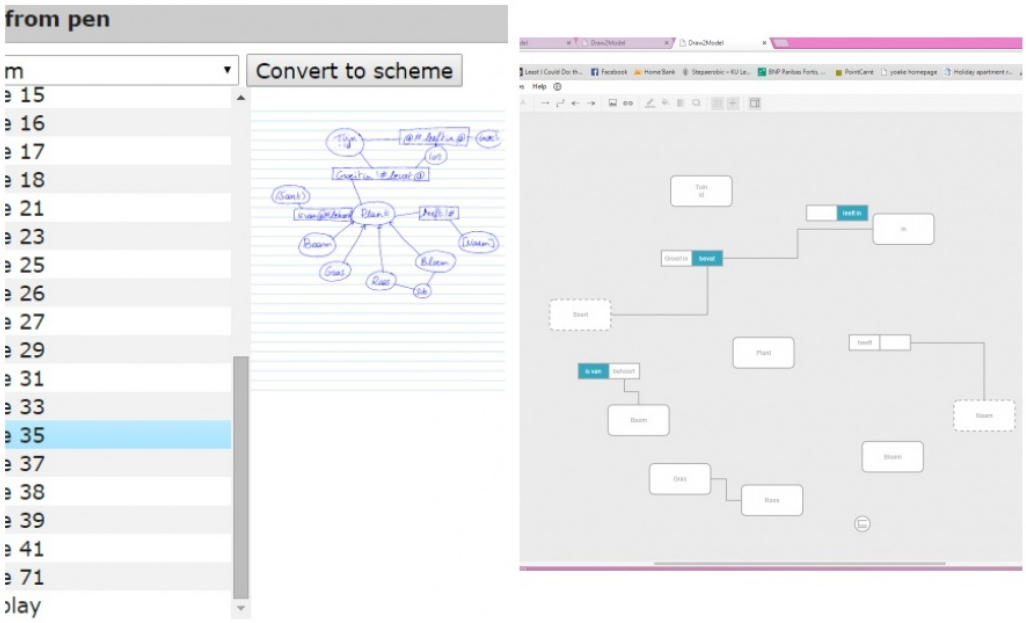
```
function escapeHtml(unsafe) {
    return unsafe
         //.replace(/&/g, "&amp;")
         .replace(/</g, "&lt;")
         .replace(/>/g, "&gt;")
         .replace(/"/g, '')//"&quot;")
         .replace(/'/g, '')//"&#039;")
         .replace('@', '')
        .replace('?', '')
        .replace('(', '')
        .replace(')', '')
        .replace('!', '')
        .replace('#', '')
        .replace('[', '')
        .replace(']', '');
}
```

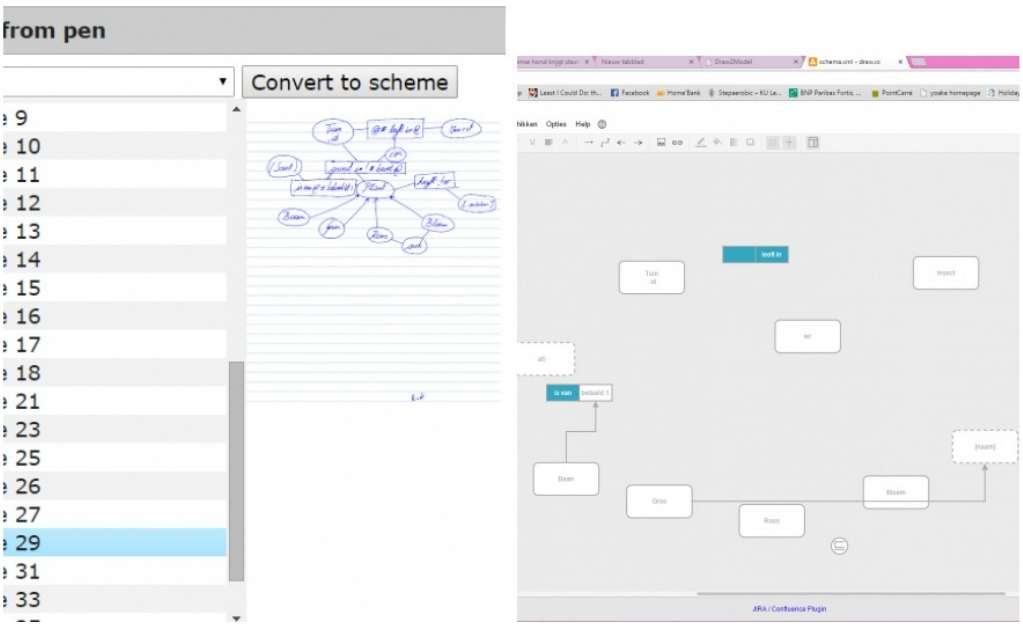
ConvertToJson

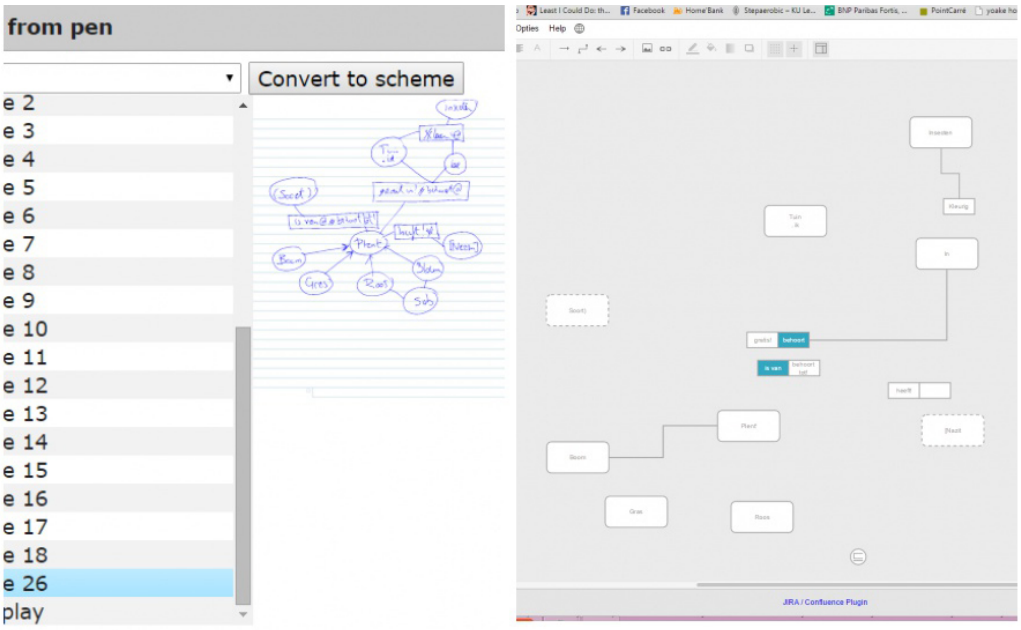
B

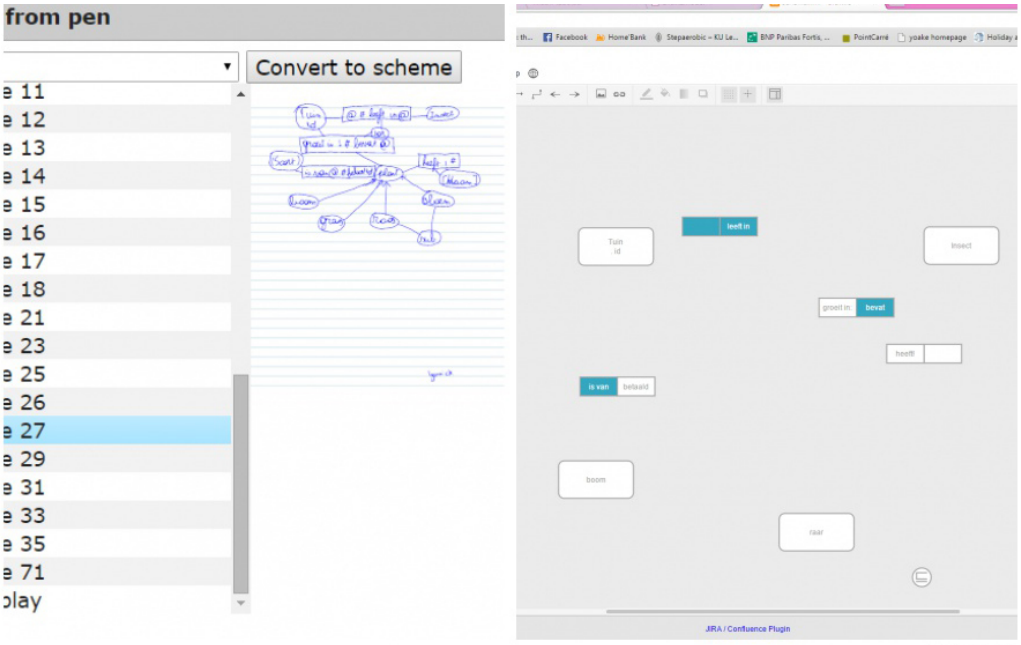
User Evaluation

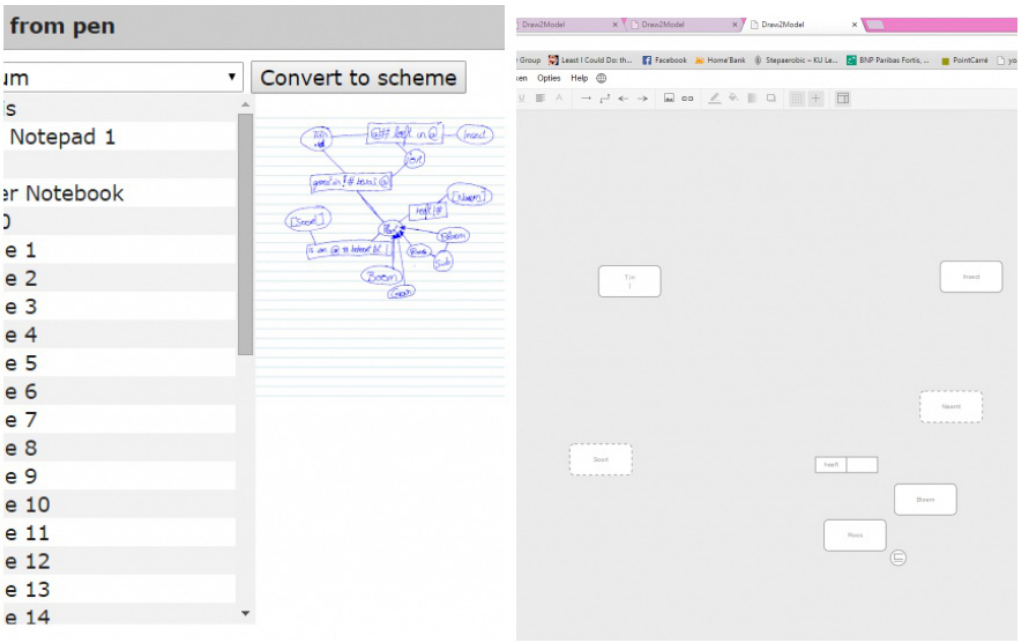
Here you can find some side by side images of the model users drew and the end result after the transformation.

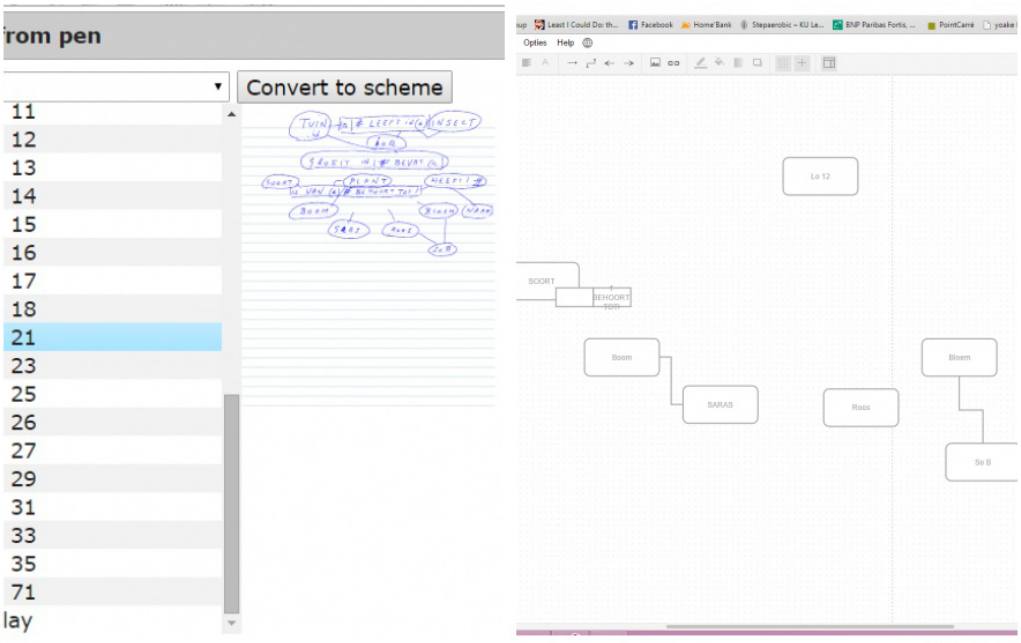


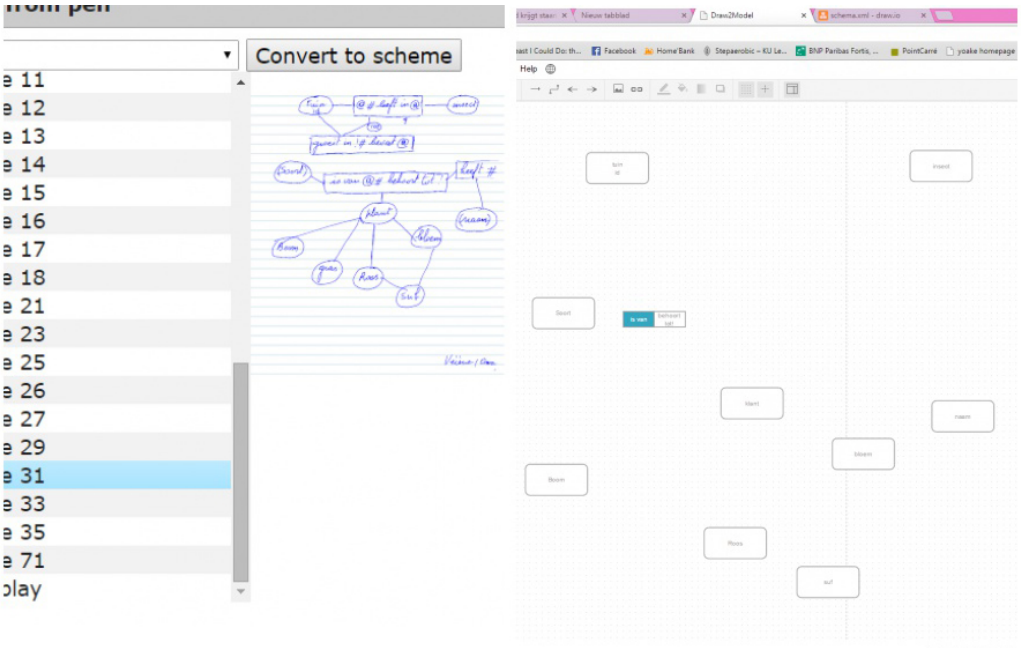

from pen
Convert to scheme
9
10
11
12
13
14
15
16
17
18
21
23
25
26
27
29
31
33

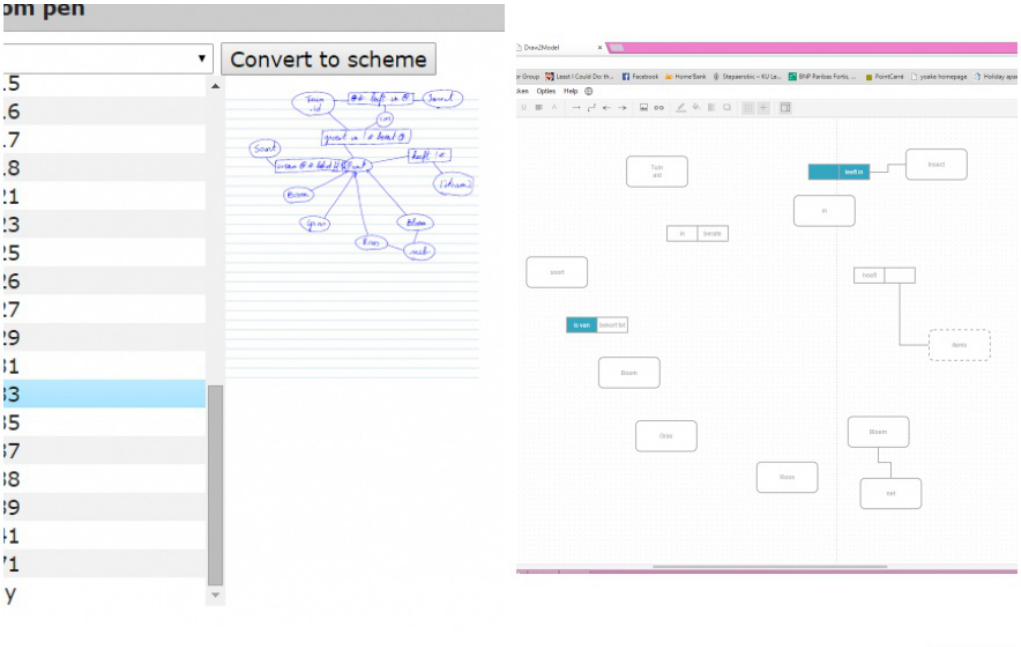

from pen
Convert to scheme
e 2
e 3
e 4
e 5
e 6
e 7
e 8
e 9
e 10
e 11
e 12
e 13
e 14
e 15
e 16
e 17
e 18
e 26
play
JIRA / Confluence Plugin

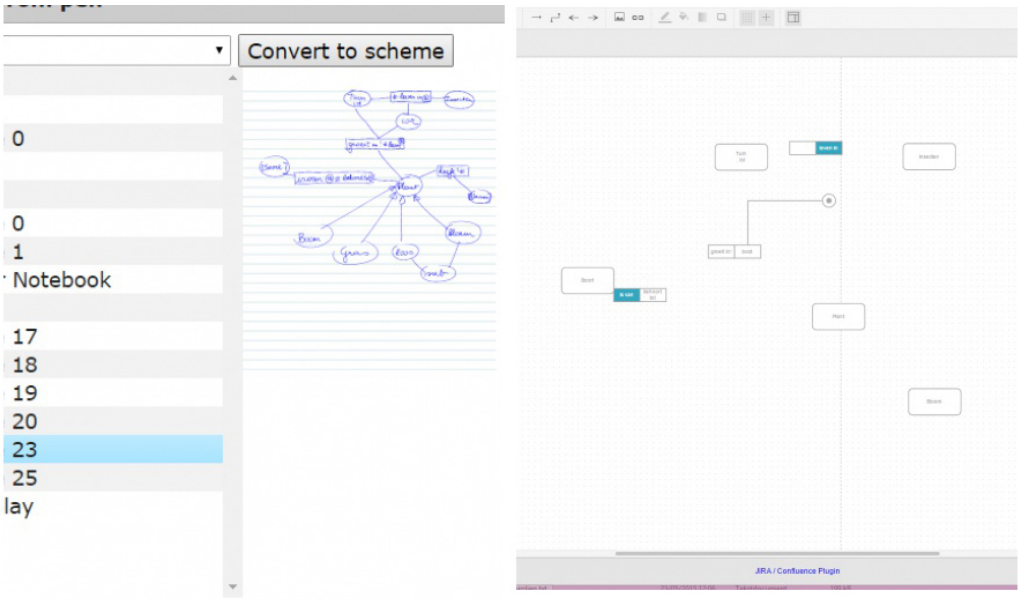

from pen
Convert to scheme
e 11
e 12
e 13
e 14
e 15
e 16
e 17
e 18
e 21
e 23
e 25
e 26
e 27
e 29
e 31
e 33
e 35
e 71
play
JIRA / Confluence Plugin

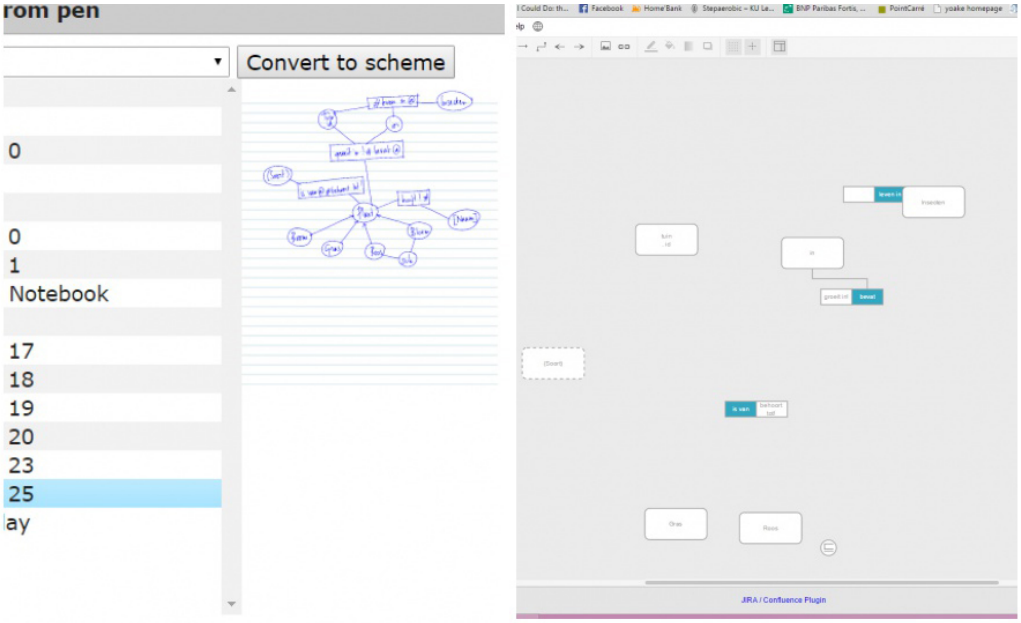

from pen
Convert to scheme
Notepad 1
Notebook
e 1
e 2
e 3
e 4
e 5
e 6
e 7
e 8
e 9
e 10
e 11
e 12
e 13
e 14

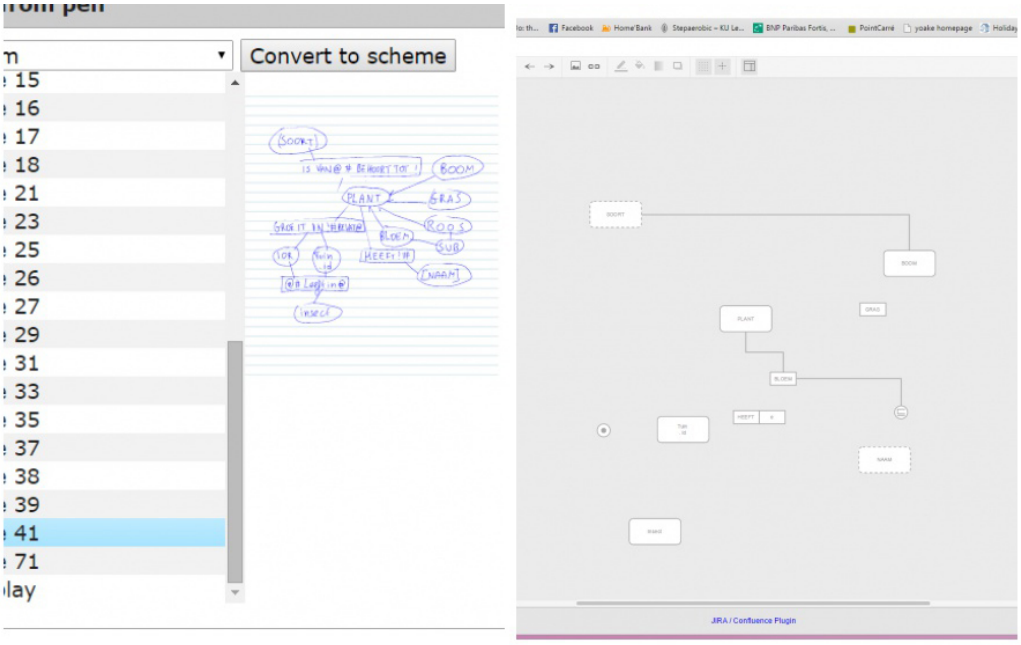

from pen
Convert to scheme
11
12
13
14
15
16
17
18
21
23
25
26
27
29
31
33
35
71
lay


Convert to scheme
e 11
e 12
e 13
e 14
e 15
e 16
e 17
e 18
e 21
e 23
e 25
e 26
e 27
e 29
e 31
e 33
e 35
e 71
play


Convert to scheme
.5
.6
.7
.8
:1
:3
:5
:6
:7
:9
;1
;3
;5
;7
;8
;9
;1
'1
y


Convert to scheme
0
0
1
Notebook
17
18
19
20
23
25
lay
JIRA / Confluence Plugin


rom pen
Convert to scheme
0
0
1
Notebook
17
18
19
20
23
25
ay
JIRA / Confluence Plugin


Convert to scheme
15
16
17
18
21
23
25
26
27
29
31
33
35
37
38
39
41
71
BOOM
PLANT
GRAS
ROOS
BLOEM
SUB
JIRA / Confluence Plugin

C

Evaluation Questionnaire

Draw2Model

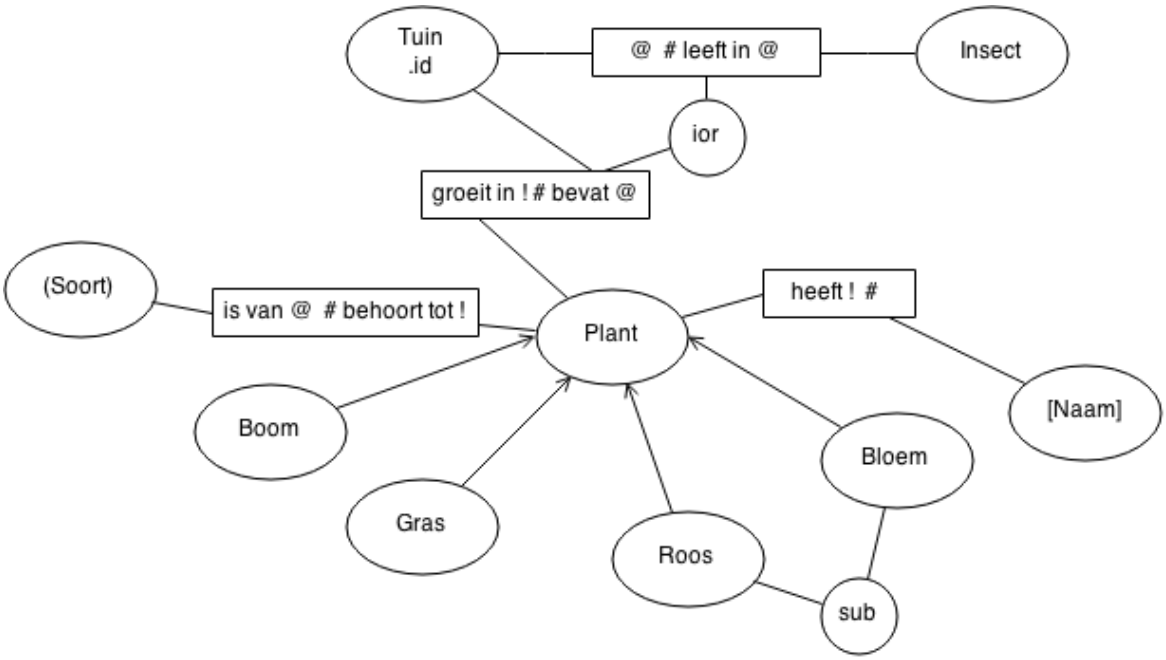
1. Role worden niet gescheiden door de rechthoek van de relatie op te delen maar door het symbool # tussen de rolnamen
2. Een verplichte rol wordt aangegeven door het symbool “!” na de rolnaam
3. Value types worden weergegeven door een volle ellips, maar met de naam tussen rechte haakjes
4. Wanneer de rolenaam een @ bevat betekend dit dat de role n-air is.
5. Constraints worden door cirkels met hun afkortingen voorgesteld.

---

1. A role is not devided by the rectangle to devide the relation. Instead the # symbol is used to indicate the end of a rolename.
2. A mandatory role is indicated by using a "!" symbol in its name.
3. Value types are indicated by an elliptical shape containing the valuename between brackets.
4. When a rolename contains a "@" symbol this meanns that the role is a n-airy role.
5. Constraints are represented by circles containing their abbreviation.

* Required

Recreate this diagram with the Echo smartpen (you may use your own handwriting for the textual part)



1. **Sex** *
   Geslacht
   Mark only one oval.
   - male/man
   - female/vrouw

2. **Age** *
   Leeftijd

3. **Are you left or righthanded** *
   Bent u links of rechthandig
   Mark only one oval.

   - left/links
   - Right/rechts

4. **Work/education** *
   Beroep/opleiding

5. **Do you have any experience with ORM?** *
   Bent u vertrouwd met ORM?
   Mark only one oval.

   - Yes/ja
   - No/nee

6. **The ! symbol in the relation's text means this relation is mandatory. Could you think of any other textual symbol that would make this clearer? If yes which one?**
   Het ! teken in de tekst van de relatie betekend dat de relatie verplicht is . Kan u een ander symbool bedenken die dit duidelijker zou maken en welk?

7. **The # symbol in the relation's text delimits this relation. Could you think of any other textual symbol that would make this clearer? If yes which one?**
   Het # teken in de tekst van de relatie betekend duid de grens van deze relatie aan . Kan u een ander symbool bedenken die dit duidelijker zou maken en welk?

8. **The @ symbol in the relation's text means that the relation involves mor than one of those objects. Could you think of any other textual symbol that would make this clearer? If yes which one?**
   Het @ teken in de tekst van de relatie betekend dat de relatie mogelijk is met meer dan een van zulke objecten . Kan u een ander symbool bedenken die dit duidelijker zou maken en welk?

   .....

   .....

   .....

   .....

   .....

9. **What, do you think, is the biggest disadvantage of using the echo smartpen? ***
   Wat is volgens u de het grootste nadeel van het gebruik van een echo smartpen voor het maken van een dergelijk diagram?

   .....

   .....

   .....

   .....

   .....

10. **How easy is it to make such a diagram with the echo smartpen? ***
    Hoe gemakkelijk vindt u het om zo een diagram te maken met de echo smartpen?
    Mark only one oval.

	1	2	3	4	5	
Not at all/helemaal niet	◯	◯	◯	◯	◯	Very/heel

11. **What, do you think, is the most valuable asset of using the echo smartpen for making this kind of diagram? ***
    Wat is volgens u de grootste meerwaarde van het gebruik van een echo smartpen voor het maken van een dergelijk diagram?

    .....

    .....

    .....

    .....

    .....

12. **Any other comments?**
    Nog andere opmerkingen?

.....

.....

.....

.....

.....
