Vrije Universiteit Brussel

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

Authoring Tool for Smart Study

Graduation thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Sciences and Engineering: Computer Science

Krista Puķe

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

Academic year 2015-2016

I never teach my pupils, I only attempt to provide
the conditions in which they can learn.
Albert Einstein

Abstract

In children development good education is fundamental. However, some children experience learning problems: they are easy distracted, lose their interest, or suffer from specific learning disorders. In our modern and digital society, we therefore aim to provide them an environment and appropriate technologies that make the learning process more interesting, attractive and exciting. In such a digital environment, teachers still play a very important role. However, most teachers were not trained for using digital tools in education. Therefore, it is essential that the new tools that they have to use are easy to learn and to apply, i.e. have a high usability.

In this thesis we present the authoring tool for a Smart Study application. Smart Study is an educational platform targeting primary school children. It was designed to simulate handwriting in primary schools, while also allowing children to work more autonomous by providing automatic feedback. The application works with a digital pen, i.e., the Livescribe Echo smartpen, printed exercise pages (with Anoto dot paper), and an Android tablet for the automatic feedback.

The authoring tool should offer teachers and publishers the necessary tools to prepare course material for the Smart Study application. The authoring tool is developed as a web-based application. Users can upload existing PDF course material, edit it and prepare the content for its use by Smart Study, and export the file in EPS format, as required by the Livescribe platform.

As usability is essential, we have evaluated the authoring tool on its usability with a small group of primary school teachers. The results show that the teachers are indeed looking for a tool that can ease the creation of teaching material and that the developed tool is a good initial concept to provide such functionalities.

To provide more context, we also present an overview of existing authoring tools; discuss eLearning trends, its importance and existing platforms; as well as other digital pen related applications.

Keywords: Authoring Tool, eLearning, PDFMiner, ImageMagick, Livescribe Platform, Pen Application, Education Digitalization

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.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Dr. Olga De Troyer and adviser Dieter Van Thienen for all the useful comments, remarks and directions they provided throughout the entire process of this master thesis. Also, I want to thank them for introducing the topic to me and for the great support during the research and implementation process.

Many thanks I would like to sent to all the participants who have willingly invested their valuable time during the evaluation processes of the authoring tool, and also I want to thank Liga Granta who helped me to organize all primary school teachers together.

I would like to thank my loved ones who always supported and stood by me, helped to keep myself harmonious and always found nice words to tell me. Also, thank you all for helping me get my thought back together and find the motivation when I felt that it will be hard to continue.

My sincere thanks goes to everyone who was involved in this thesis, with thoughts, ideas or any kind of support.

Thank you all!

Krista Puķe
July, 2016

Contents

List of Figures

Acronyms

AFD	Anoto Functionality Document.
AJAX	Asynchronous Javascript and XML.
CL	Collaborative Learning.
CSS	Cascading Style Sheets.
DB	Database.
DSRP	Design Science Research Process.
EPS	Encapsulated PostScript.
ERD	entity-relationship diagram.
HTML	Hyper Text Markup Language.
IDE	Interactive Development Environment.
ITSs	Intelligent Tutoring Systems.
JDK	Java Development Kit.
JRE	Java SE Runtime Environment.
JS	JavaScript.
MLE	Managed Learning Environment.
ORM	Object Role Model.
PDF	Portable Document Format.
PDO	PHP Data Objects.
PHP	Hypertext Preprocessor (HTML-embedded scripting language).
PNG	Portable Network Graphics.
SDK	Software Development Kit.

SUS	System Usability Scale.
UC	Use case.
UI	User Interface.
UTF8	Universal Transformation Format-8.
VLE	Virtual Learning Environment.
XML	Extensible Markup Language.

Part I

Introduction

1 Introduction

The previous century was characterized by the impact of computers and smartphones, which drastically changed the world. Digitalization and the Internet provide us a wide range of information and immediate access to it. This evolution also has its impact on education.

In this thesis we present an authoring tool for a Smart Study application (further referred to as the *authoring tool*). Smart Study is an educational platform targeting primary school children. It was designed to simulate handwriting in primary schools, while also allowing children to work more autonomous by providing automatic feedback. The application works with a digital pen, i.e., the Livescribe Echo smartpen, printed exercise pages (with Anoto dot paper), and an Android tablet for the automatic feedback.

An authoring tool for Smart Study should offer teachers and publishers the necessary tools to prepare course material for the Smart Study platform. The authoring tool is developed as a web-based application. The tool offers teachers and other educational publishers the possibility to digitalize existing course material and integrate it into the Smart Study platform. A user can upload documents in PDF format. The uploaded documents will be parsed using the PDFMiner library. After parsing, the data is analyzed and displayed in the authoring tool. The content of the parsed document is presented separately for each page and in the same sequence as it appeared in the original document. The content within a page is editable and additional information, needed for the Smart Study application, can be specified, i.e., answer and explanation for an exercise, course and category for a page. The content is then uploaded to a Database (DB). The DB is used as the common content repository for the authoring tool and the Smart Study application. The pen application process could not be fully automated due to a limitation of the Livescribe Anoto Functionality Document (AFD) format that is required to create a Penlet application, but which is a strictly encrypted outcome of the fixed print Paper Product. The authoring tool provides the ability to export the PDF document to Encapsulated PostScript (EPS) file format. This format is required by the Livescribe SDK, to build a pen application. In this way, we implemented an easy way to digitalize already existing teaching materials. By realizing the integration with the Smart Study application we open the way for providing creative and attractive learning material.

1.1 Motivation

Digitalization is applied to a wide range of sectors. Also in education it seems to be a promising approach that also allows to make education more appealing. By using digitalization, the teaching and learning processes can be improved, remove some of the time consuming tasks and provide an environment for students and teachers that is more relaxing and fun to use. Another advantage of digitalization is that it makes learning materials accessible from computer devices with an Internet connection anytime, anywhere (Johari, 2013).

A large number of teachers rely on printed teaching material. However, with the rise of digitalization, the content becomes more digitalized and new digital learning tools emerge. Although most teachers realize that the rise of educational technologies is inevitable, it is also known that for a lot of teachers it is a great challenge to start using them.

One of these new educational technologies is Smart Study, an educational platform to stimulate handwriting with primary school children. However, the Smart Study platform was lacking an easy to use authoring environment for creating the learning material. The goal of the thesis work was to develop such an authoring environment usable by non-IT schooled people. The research questions related to this goal can be formulated as follows: Is it feasible and useful to create and integrate an authoring tool for the Smart Study educational platform that can largely automate the creation of its exercise pages? The practical research problem is the creation of such an authoring tool. The target audience of the tool are primary school teachers and educational publishers. However the proposed solution is not limited just to the primary school level, but can be applied to all levels of education.

1.2 Research Goals

The main goal of this thesis was to develop an authoring environment usable by non-IT schooled people for the Smart Study educational platform. The solution could be to create an application where professionals, i.e., teachers and pedagogical experts, could create or upload content and automate (or ease) the generation of the pen application. The tool should be integrated with the Smart Study application.

1.3 Research Methodology

To achieve the research goal we used Peffers et al. (2007) Design Science Research Process (DSRP) model. The proposed process model includes five steps presented in figure 1.1 that will be explained in this section.

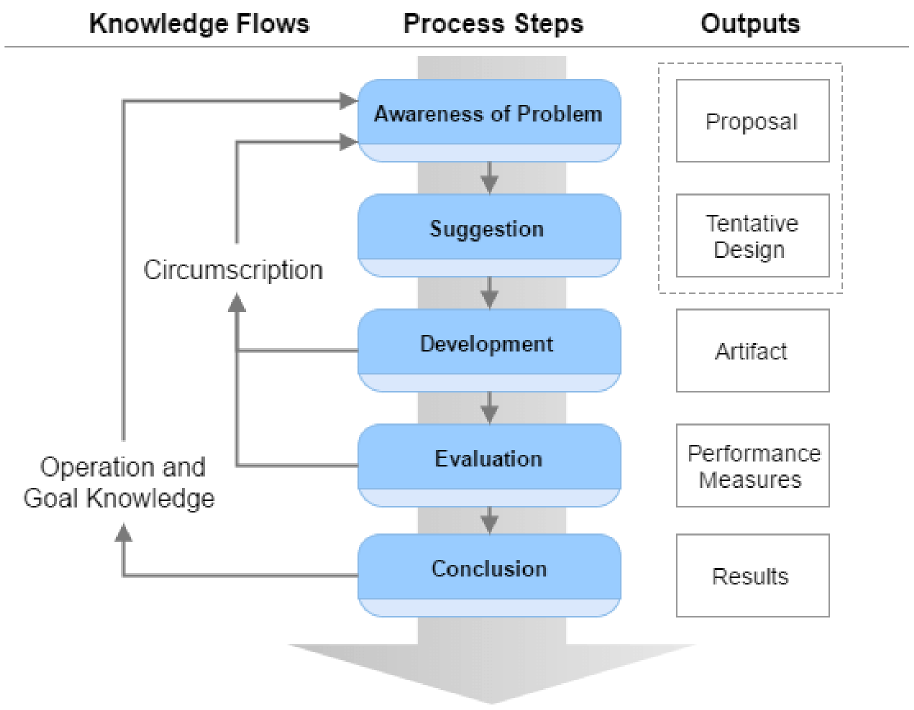


Figure 1.1: Design Science Research Process (DSRP) model (Peffers et al., 2007)

Applied steps:

1. *Awareness of problem*
   With respect to the research question, in order to gain a deeper understanding of the problem, we performed a number of literature studies. This research was on authoring tools for educational purposes, including topics such as eLearning and smartpen applications.

2. *Suggestion*
   For this step, we analyzed existing technologies (PDFMiner, ImageMagick, Livescribe platform), described our users, formulated requirements, defined the different authoring tool components, and designed the User Interface (UI) and the DB.

3. *Development*
   The outcome from the previous step is the *tentative design* that is implemented and developed in this step. To that end we developed the authoring tool as a PHP web application. For dealing with PDF documents we used the HTML5 upload form and to parse them we integrated the PDFMiner (Shinyama, 2010). The UI was constructed based on the *ZURB Foundation*[1] and to convert PDF to EPS file format and to compose images we used ImageMagick (ImageMagick Studio LLC, 2016). As an application server we used Apache and as DB MySQL from the XAMPP controller tool.

4. *Evaluation*
   In this step we evaluated the developed authoring tool. The evaluation took place in three phases where we had open discussions with the participants in a focus group, we asked them to test the application by following a given scenario and afterwards to assess it using a System Usability Scale (SUS) survey and additional questions.

5. *Conclusion*
   In the final step we reflected on the results obtained. We also identified limitations and future work.

1.4 Outline of the Dissertation

The first part of the thesis introduces the problem, research goals and methodology as well as the motivation of the dissertation.

[1]http://foundation.zurb.com/

The second part focuses on and presents the background information. Chapter 2 introduces the topic of authoring and presents a classification of authoring tools. Chapter 3 gives an overview of the eLearning topic, discusses advantages and disadvantages and also introduces and describes the content authoring tools. Chapter 4 presents an overview of the related work.

The third part presents the development process. This part includes the following chapters:

Analysis Chapter 5 presents the main technologies that we discovered and used in the authoring tool development. The technologies we describe are: PDFMiner, ImageMagick and Livescribe platform.

Design Chapter 6 discusses all the design choices. It introduces user classes, requirements as well as the architecture of the application. It also describes the relations between the application's components, user interface and database design.

Implementation Chapter 7 presents the implementation of the authoring tool by explaining the main files and methods, as well as it describes the implementation of the PDFMiner, ImageMagick and the database.

Evaluation and Results Chapter 8 presents the evaluation setup and methodology as well as the results and discussion.

Conclusion Chapter 9 concludes the thesis by describing the summary of the developed application, it also presents the future work and limitations.

Part II

Background

2 Authoring

An author is not only the person who is the writer of a book, essay, article or other text, but also the person who constructs electronic documents and their content, software programs or who is the creator of courseware, web pages, sound, video or animation (Dictionary.com, 2016; Farlex Inc., n.d.). To create such content, authors are using authoring tools.

2.1 Authoring Tool Classification

Dabbagh (2002) presents the authoring tools grouped in three categories. In addition, he also mentions the general features and instructional products for each category.

1. CD-ROM-based Authoring Tools that use CD-ROM as media to deliver the tools (the content is thus stable and cannot be modified by the users)

   - *General features:* Tools generally are used with CD-ROM and videodisc technologies. The interface is tool-based and the content is stable. Mostly the linking is internal, but external links can be created using the web browser. CD-ROM-based authoring tools require installation and are operating system dependent.

The users of such systems are developers, as a result of not having a user interface for learners or instructors.

   - *Instructional Products:* Simulations, tutorials, games, standard testing programs and criterion-references testing (e.g., tests, exams and quizzes), etc.

2. Web-based authoring tools

   - *General features:* The interface is accessible through a web browser therefore Internet connection is required. Tools are extensible, allow dynamic content and collaborative media. This type of tools is in general harder to learn and to take a full advantage of such a tool requires a long learning process.
   - *Instructional Products:* Personal or organization portfolio web pages, web-based resources, blogs, etc.

3. Web-based course management tools

   - *General features:* This type of authoring tools is easy to use, allows dynamic content and enables collaborative media. The tools have speci c parts that are developed for learners, instructors and administrators, also communication tools are embedded to provide discussion forums and to allow manage and exchange emails.
   - *Instructional Products:* Distance education programs, courseware, asynchronous and synchronous learning environments, etc.

(Dabbagh, 2002)

2.1.1 Structured and Unstructured Authoring

Quark Software Inc. (2015) describes two ways how content can be authored: structured and unstructured. As an example of unstructured authoring Quark Software Inc. (2015) mentions Microsoft Word and Google Docs, where a content author traditionally composes text and spends quite a long time formatting it. The process of formatting can get more complex when the author wants to add additional information by reusing some content (copy and paste). As there is no other linking between the original and copied information, redundant copies are created. Structured authoring tools are using Extensible Markup Language (XML), that allows automated formatting and can different have content types. By using structured text the content can be easily reused, to that end the author should point to the original content. If the original content changes, all the linked ones are updated. (Quark Software Inc., 2015)

2.1.2 Specialized and Unspecialized Authoring

Khademi et al. (2011) distinguishes between specialized and unspecialized authoring tools. As a difference the author mentions, that unspecialized authoring tools such as Microsoft PowerPoint, Front Page and Dreamweaver requires a team of programmers around the instructor to develop courseware. Such tools provide a wide range of functionality but do not help or guide the content author while creating interactive course materials. While specialized authoring tools concentrate on course and learning object creation. The author mentions the specialized tools are easy to use and provides the instructor with guidelines while creating rich media courses. The course materials can contain numerous objects and object types (see fig: 2.1). (Khademi et al., 2011)

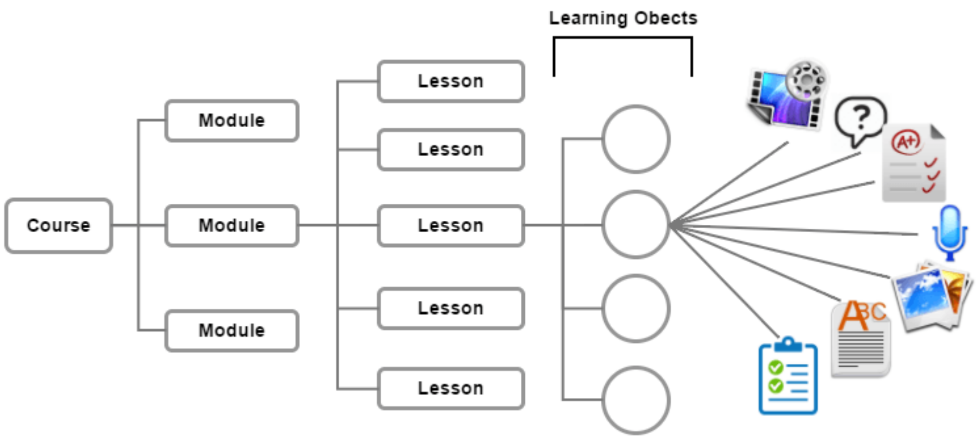


Figure 2.1: Learning Object Hierarchy (Mohan, 2004; Khademi et al., 2011)

2.2 Summary

This chapter presented the characteristics of authoring tools, discussed specialized and unspecialized authoring as well as structured and unstructured authoring.

In the following chapter we present eLearning, a type of authoring concept. Here we discuss several eLearning applications and describe eLearning with its accompanying advantages and disadvantages.

3 eLearning

eLearning promotes the creation of teaching materials, including courses, exercises and tests through means of information and communication technologies. The main key word here is the ‘e’ that stands for electronic and enhanced. The term eLearning involves the incorporation of various electronic media to be used by educational institutions for education and training. (Legault, 2014; Jisc Digital Media, 2016)

The Markotic (2015) blog “E-learning for primary school teachers” mentions the advantages and disadvantages for teachers to use eLearning in primary schools.

Advantages: all the resources are easily accessible for instructors and pupils, from all kind of computer devices that have an active Internet connection. It is a benefit for the authors to create the teaching materials, exercises, tests, homeworks and quizzes from any place and time. Markotic (2015) also mentions that eLearning tools are often costless, which is also why they are available for everyone. Both teachers and students can get immediate feedback. eLearning is also a way to reduce stress and increase the interest of students, because they can pick the tempo they want to go through each topic. They can skip a topic when they have enough knowledge about it and slow down when reaching any unfamiliar ones.

Disadvantages: it requires a computer device and an Internet connection, that might not be available for everyone. As well as it demands computer skills and in some cases the knowledge of document management. Additionally, the students might lack of face-to-face contact with the teacher.

3.1 Interactive whiteboard

The interactive whiteboard is a large size display with which the instructors and pupils can interact using a computer mouse, fingers and special gadgets like stylus or pen. With these tools users can create, move and manage the information all over the whiteboard. W. Caron (2011) mentions that this technology allows pupils to collaborate with each other during classes by working in groups and allows them to be more creative. (W. Caron, 2011)

There is a wide range of applications developed for the interactive whiteboard (or smartboard), covering topics such as: alphabet and ordering, learning languages, literature, reading and story making, music, biology, physics, math, quizzes and much more. Figure 3.1 presents the pupil interaction with the interactive whiteboard. (Vision One, 2015)

Figure 3.1: Interaction with interactive whiteboard (Vision One, 2015)

3.2 Moodle

Moodle is an eLearning project that is a free and open-source Virtual Learning Environment (VLE). The name of the tool stands for Modular Object-Oriented Dynamic Learning Environment. Moodle is used all over the world and according to the statistics in 2015 the tool has more than 400 thousand registered users from 193 countries with it its interface translated in over 86 languages. (Al-Ajlan, 2012; Kumar et al., 2011)

Moodle is a module-based platform that allows users to pick plugins to receive the desired functionality. Authentication and administration are one of the main modules, where the login of all users can be controlled and the users managed. The tool provides 6 user types: administrator, course creator, teacher, non-editing teacher, student and guest. Each of these user definitions have different roles in the system. All the provided modules are grouped in various categories (e.g., activities, themes, users, filters, reports, gradebook, editors, calendar, plagiarism, etc.). The tool allows instructors to create and share their own courses, quizzes, assessments, activities, forums and chats etc. with students. Students can follow the classes and upload homework online. Moodle also provides pupils with the platform were they can talk and collaborate to each other. (Al-Ajlan, 2012; Kumar et al., 2011)

Moodle provides a way to deliver created courses to pupils. All the courses can have multiple sections and can have various activities, for example survey, quiz, assignment or forum. The new courses can be stored locally or published on the Moodle Hub, where they are accessible for all the Moodle users (see fig 3.2). The materials for courses can be retrieved from the local storage or other content repository (see fig: 3.3)(Al-Ajlan, 2012; Kumar et al., 2011)

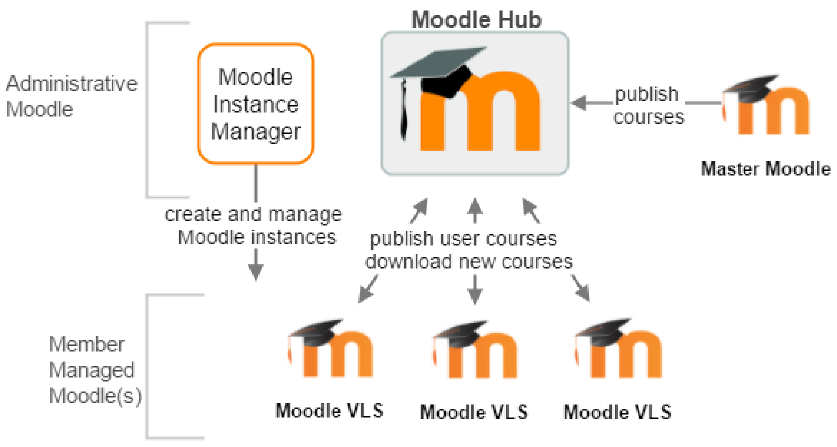


Figure 3.2: Moodle Management diagram (Melia, 2012)

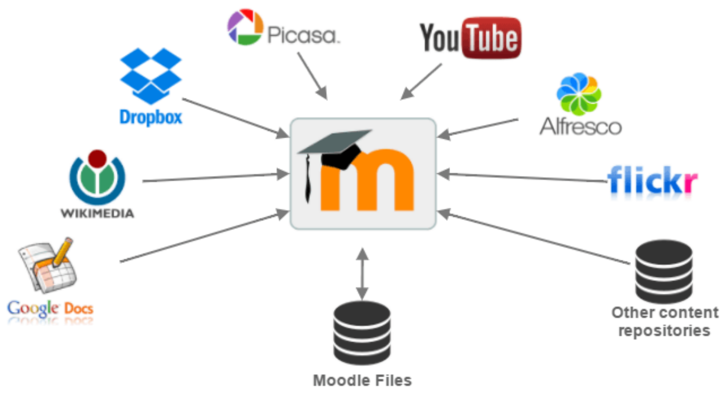


Figure 3.3: Moodle repositories (Dougiamas, 2010; Moodle community, 2015)

3.3 Blackboard

Blackboard (Bradford et al., 2007) is a Managed Learning Environment (MLE) (or VLE) and it is one of the leading learning environments, but it is not free compared to Moodle (iSpring Solutions, Inc., 2015). This system allows instructors to create and manage courses, as well as providing the communication with the students and gives immediate feedback of the pupil performance.

Bradford et al. (2007) mention the potential benefits for users using the Blackboard system:

Increased Availability: The system increases accessibility to all courses and related materials (tests, quizzes, homeworks, notes, slides, etc.), because it is available on various devices with an internet connection.

Quick Feedback: Blackboard provides the student two types of feedback: faculty-initiated and automated. If the teacher sets the automated grading then the student can see immediate grading or results. Also the institution can get an instant feedback from surveys.

Improved Communication: The system after authorization allows users to overview all personal emails, announcements or discussions where he is involved. Blackboard also provides students the functionality to ask a question related to a course.

Tracking: The instructors can overview the statistics of all the courses he is assigned to, as well as each student action separately (for example the homework submission date). Pupils can always track the received grades and their own progress on the *GradeBook*.

Skill Building: Bradford et al. (2007) mention that the use of the Blackboard improve skills like organization and time management.

The Blackboard system supports the integration with other VLE tools such as: Moodle, Pearson LearningStudio, Sakai Collaborative and Learning Environment, BrainHoney, Desire2Learn, itsLearning and Epsilen (FinancesOnline, 2016).

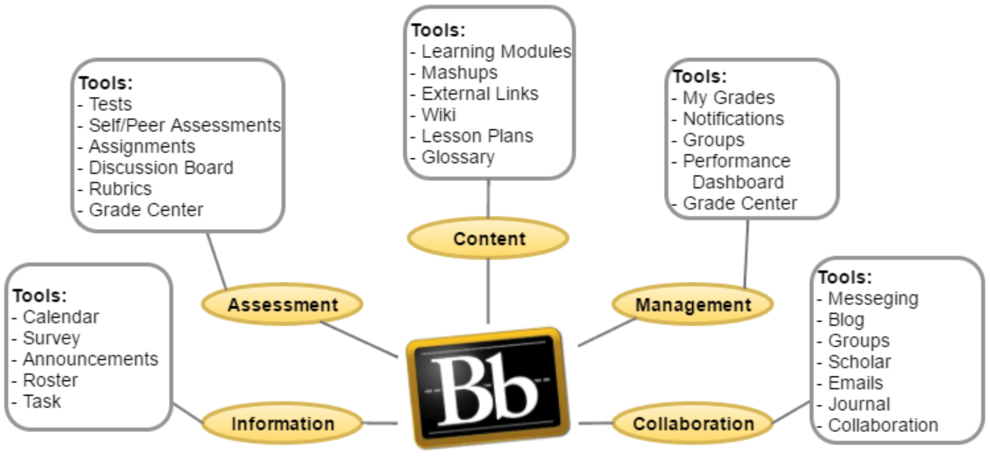


Figure 3.4: Blackboard features (University of Otago, 2014)

3.4 ViSH Editor

ViSH Editor is an eLearning based authoring tool that is built on top of the central web platform called Virtual Science Hub (ViSH). ViSH is an eLearning platform for scientists to share the resources and collaborate with each other. The ViSH Editor is developed in order to allow teachers, scientists and students to access the learning materials from integrated e-Infrastructure[1]. Authors introduced the ViSH Editor to create and share the learning objects (Virtual Executions) from the ViSH resources, and provide this information to the students. Gordillo et al. (2013)

ViSH contains the learning object repository (LOR) that consists of two major components:

- Resources uploaded by the scientists, see in the figure 3.5 e-Infrastructure resource

[1]ec.europa.eu/digital-single-market/en/e-infrastructures

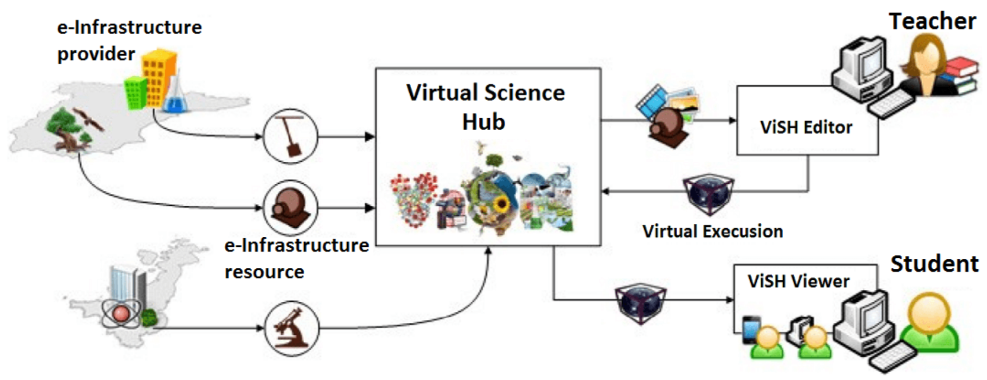


Figure 3.5: Virtual Excursion generation (Gordillo et al., 2013)

- Virtual Executions - created by the teachers

ViSH Editor is accessible from any device with active internet connection and it does not require any additional set up. The teacher can use any resource from the learning object repository and include it in the Virtual Execution. Besides data from LOR the Virtual Execution can contain any additional information provided by the teacher (self created content, PDF file, image or resource from YouTube, Flickr, etc.). The UI to create the Virtual Execution is a composition of slides where each slide can contain various objects (see fig: 3.6).

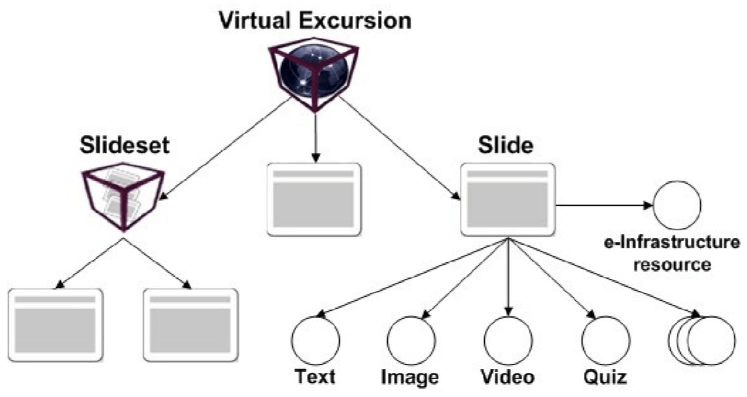


Figure 3.6: Virtual Excursion components (Gordillo et al., 2013)

3.5 Summary

In this chapter we take a deeper look into the eLearning concept. In addition, we discuss several tools that enforce the eLearning concept such as the Interactive whiteboard, the Moodle VLE, the Blackboard VLE, the ViSH Editor.

Next we present the chapter Related Work, where we will see several digital pen related applications.

4 Related Work

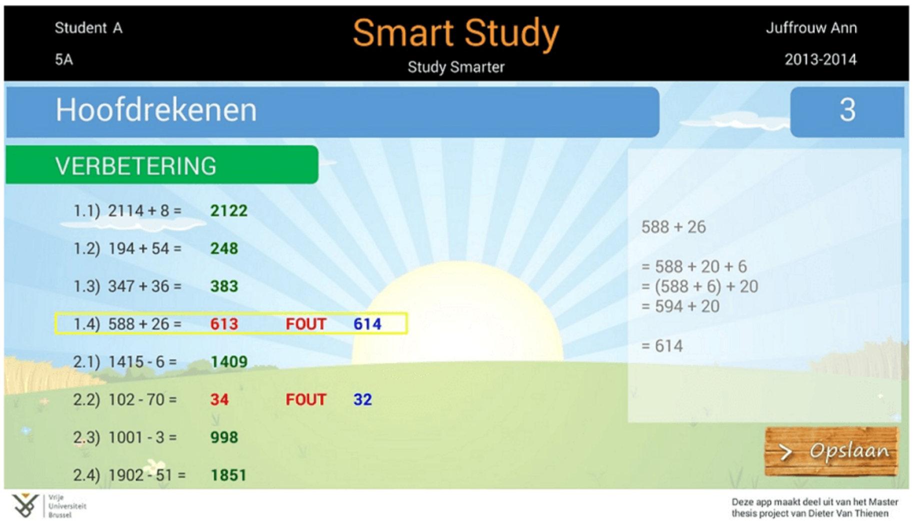
In this chapter, we discuss related work. The first type of related work considered are authoring environments for educational platforms. Most platforms have the authoring environment integrated into the learning environment. Although Smart Study does not yet provide an authoring environment, we discuss it here because in the end it is the goal to also have an authoring environment integrated. The second type of related work are digital pen applications developed in the context of education.

4.1 Smart Study

Smart Study is an educational platform developed by Van Thienen et al. (2015). The platform is targeted at primary school children in order to stimulate their handwriting, get automated feedback without teacher interaction, and make homework more fun. The application works using the Livescribe Echo smartpen, printed exercise pages with an Anoto dot pattern and an Android tablet with the Smart Study application.

When the user fills out an exercise page, the smartpen provides feedback by presenting the written answer on the screen embedded in the pen. After the student finishes filling out an exercise page, the smartpen is connected to a computer device running the Smart Study framework, inserts the data collected from the smartpen into a database, where after the tablet app reads the newly inserted data from the database, it displays this data as feedback to the user.

Next the user can have an overview of the provided answers and get an explanation if the answer was wrong, displayed on the Android tablet application. The figure presents the Smart Study application screen with the correct and wrong answers. It also illustrates the explanation of the exercises if the provided answer is not correct.


Student A
5A
Smart Study
Study Smarter
Juffrouw Ann
2013-2014
Hoofdrekenen
3
VERBETERING
Opslaan
Vrije Universiteit Brussel
Deze app maakt deel uit van het Master thesis project van Dieter Van Thienen

1.1) 2114 + 8 = 2122
1.2) 194 + 54 = 248
1.3) 347 + 36 = 383
1.4) 588 + 26 = 613 FOUT 614
2.1) 1415 - 6 = 1409
2.2) 102 - 70 = 34 FOUT 32
2.3) 1001 - 3 = 998
2.4) 1902 - 51 = 1851

588 + 26

= 588 + 20 + 6
= (588 + 6) + 20
= 594 + 20

= 614

Figure 4.1: Smart Study application presenting overview of the exercises (Van Thienen et al., 2015)

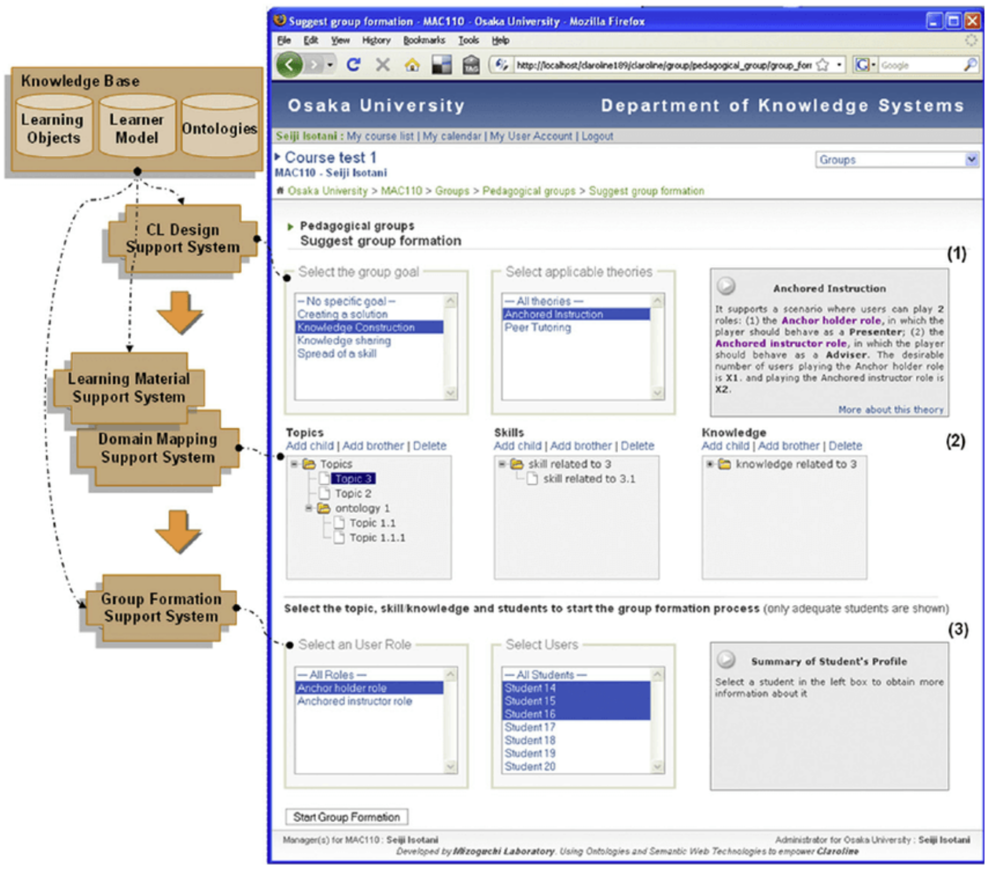
4.2 CHOCOLATO

CHOCOLATO is an authoring tool developed by Isotani et al. (2010) that helps teachers create Collaborative Learning (CL) activities. The name of the tool stands for "Concrete and Helpful Ontology-aware Collaborative Learning Authoring Tool". The creation of a well-designed CL scenario requires expertise and knowledge of multiple theories and practice related to the CL processes. The tool based on the multiple learning theories provides various methodologies that guides the teachers to develop CL scenarios and activities. (Isotani et al., 2013, 2010) CHOCOLATO provides a simple UI that is presented in the figure 4.2. As the figure presents the CHOCOLATO UI consists of three blocks containing dynamic information:

1. (1) In the first block user can select the goal of the group and the preferred learning theory

2. (2) The second block allows user to connect the provided ontologies with the domain content

3. (3) In the third block user can create the student groups


Knowledge Base
Learning Objects
Learner Model
Ontologies
CL Design Support System
Learning Material Support System
Domain Mapping Support System
Group Formation Support System
Suggest group formation - MAC110 - Osaka University - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/claroline189/claroline/group/pedagogical_group/group_for
Google
Osaka University
Department of Knowledge Systems
Seiji Isotani : My course list | My calendar | My User Account | Logout
Course test 1
MAC110 - Seiji Isotani
Groups
Osaka University > MAC110 > Groups > Pedagogical groups > Suggest group formation
Pedagogical groups
Suggest group formation
(1)
Select the group goal
Select applicable theories
(2)
Topics
Skills
Knowledge
Add child | Add brother | Delete
(3)
Select an User Role
Select Users
Start Group Formation

Select the group goal:
- – No specific goal –
- Creating a solution
- Knowledge Construction
- Knowledge sharing
- Spread of a skill

Select applicable theories:
- — All theories —
- Anchored Instruction
- Peer Tutoring

Anchored Instruction

It supports a scenario where users can play 2 roles: (1) the **Anchor holder role**, in which the player should behave as a **Presenter**; (2) the **Anchored instructor role**, in which the player should behave as a **Adviser**. The desirable number of users playing the Anchor holder role is **X1**. and playing the Anchored instructor role is **X2**.

More about this theory

Topics: Topics > Topic 3, Topic 2, ontology 1 > Topic 1.1, Topic 1.1.1

Skills: skill related to 3 > skill related to 3.1

Knowledge: knowledge related to 3

Select the topic, skill/knowledge and students to start the group formation process (only adequate students are shown)

Select an User Role:
- — All Roles —
- Anchor holder role
- Anchored instructor role

Select Users:
- — All Students —
- Student 14
- Student 15
- Student 16
- Student 17
- Student 18
- Student 19
- Student 20

Summary of Student's Profile

Select a student in the left box to obtain more information about it

Manager(s) for MAC110 : **Seiji Isotani**   Administrator for Osaka University : **Seiji Isotani**

*Developed by **Mizoguchi Laboratory**. Using Ontologies and Semantic Web Technologies to empower **Claroline***

Figure 4.2: CHOCOLATO background architecture and the main user interface (Isotani et al., 2010)

The experiment done by the authors proved that the teachers had difficulties to design the CL scenario without support, while the second experiments part that took 2 years gave positive results. The experiment shows that the CHOCOLATO improves the CL scenario creation and it gives a great explanation of theories and recommendations. (Isotani et al., 2013)

4.3 Authoring Tool for Algebra-Related Intelligent Tutoring System

Virvou & Moundridou (2000) introduces the web based authoring tool for the Intelligent Tutoring Systems (ITSs). This tool is usable for all domains that use algebraic equations (e.g., chemistry, medicine). The main goal of this tool is to allow users to create a domain description and the exercises in the instructors mode and give exercises and explanation in the students mode. Figure 4.3 presents the architecture of the system.

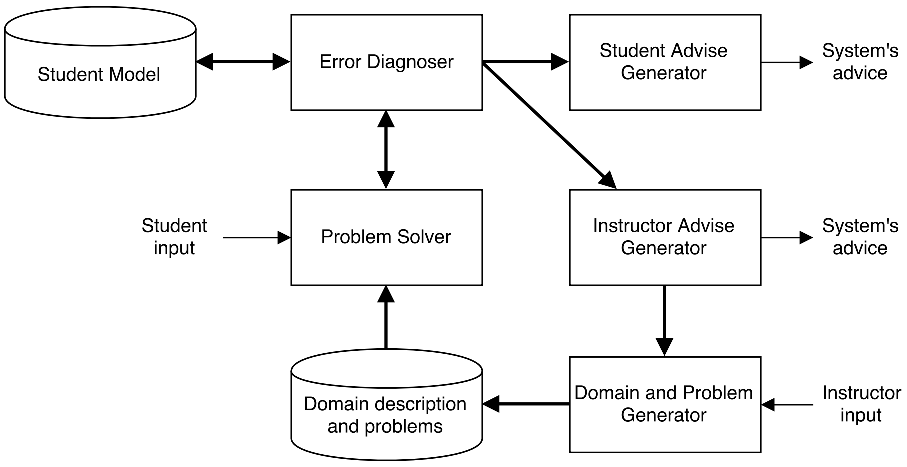


Figure 4.3: The architecture of the system (Virvou & Moundridou, 2000)

The author in the instructor mode is asked to provide the domain description, which implies the formulas, variables and descriptions of them. After the domain description is set the authors can create the exercises and the relevant difficulty level. All the information given by instructor is stored and used in the Problem Solver. The student mode distinguishes each student separately by the user name and password, this is important because every student has its own profile with the knowledge level and past exercise. The knowledge level is changed based on the user performance; the system uses it to decide the next exercise. The system provides the solution, explanation and errors (typo, mistake in equation or lack of knowledge).

The tool also provides the author and student with the feedback of solved problems. Based on the feedback the teacher can reorganize the exercises or increase/decrease the difficulty level. (Virvou & Moundridou, 2000)

As the main characteristics of the system the authors mentioned:

- The tool is a web application that allows the students and teachers to use it on various devices that has an active Internet connection.

- The authors can create multiple domain descriptions and various related exercises. When the author creates the exercises he is asked to provide a couple of variables and the text, next the system builds the equation and the explanation itself.

- The individual feedback of accomplished exercises is provided to the author and student.

4.4 Write-N-Speak

Write-N-Speak is a toolkit to create a digital-paper documents and is developed by Piper et al. (2011). In this paper the authors present and describe the development process of the multi-modal authoring tool. The aim of this research was to help the speech language therapists to build a wide range materials for aphasia patients. *(Aphasia is a brain disease that does not impact the intelligent but create difficulties to speak, listen and read as well as to understand people around (Mayo Clinic, 2015))*

The Write-N-Speak is an application that runs on the Livescribe Pulse smartpen and one of the main investigations in this research was to find an affordable and easy to use tool. The authors also notice that to create a digital paper application (penlet and paper project) requires a knowledge and experience of programming.

The Write-N-Speak system is created to help therapists to create multimodal digital paper documents that consists of four main speech-language therapy components, such as the auditory comprehension and speaking, reading and writing.

The Write-N-Speak authors mention three main aspects of authoring digital materials:

- Paper material configuration:
  All the used materials (images, blank pages, drawings, stickers, etc.) should be applied with the dot pattern or printed on the page with the dot pattern. Also the prepared materials can be re-printed, cut or glued while not loosing the link with the record.

- Multimodal interaction programming:
  The Write-N-Speak tool has two main interactive paper interfaces, the "paper editor mode" and "client mode" (see fig: 4.4). In the paper editor mode the user (therapist) can define regions, by drawing the pattern on the dotted paper (or other material applied with the dot pattern). Next by using the paper editor panel (see Figure:4.4) the author has

the opportunity to link digitally the selected region for example with the video, sound or button (called as pen's action)

- Use of adjustable (custom) content:
  After the materials are created and adjusted they can be used during the sessions with the aphasia clients. The therapists by using the main control panel (see Figure:4.4) can start and stop the interaction with the application as well as to set the volume of the Pulse smartpen, repeat the audio record or change the handwriting recognition speed.

(Piper et al., 2011)

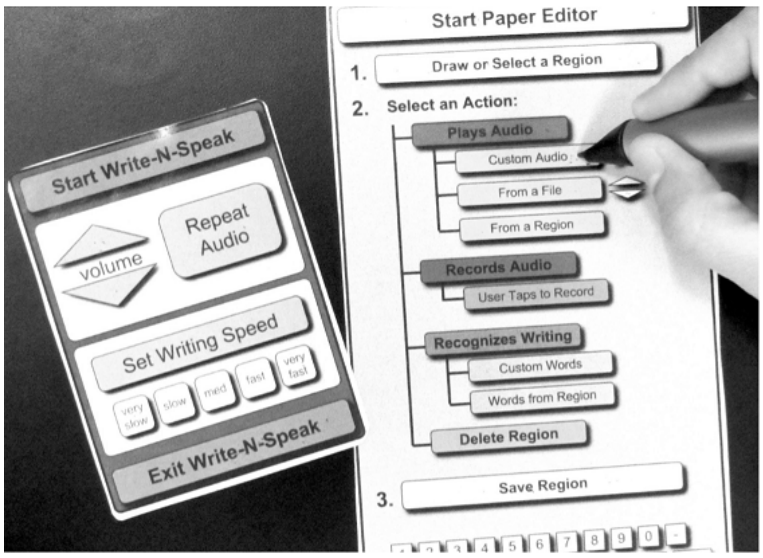


Figure 4.4: Write-N-Speak paper editor mode. From left: Main control panel, paper editor control panel (Piper et al., 2011)

4.5 TAP & PLAY

TAP & PLAY is an authoring toolkit to develop interactive paper documents for language learning actions targeting users with no technical background. The application is an extension of the previous work (see subsection 4.4). Authors justify their choice to use this hybrid digital-paper technology by saying that "*computers are difficult to use or inappropriate for some tasks*" though everybody is familiar with the paper and pen also the majority of language learning materials are paper based. The developed application runs on the *Livescribe smartpen* and is developed for two type users: the authors that create digital content documents and the content users who will work with the materials. End-users can create documents in similar way how it is with *Write-N-Speak* application by using main control panel and general interface control (see Figure:4.5). (Piper et al., 2012)

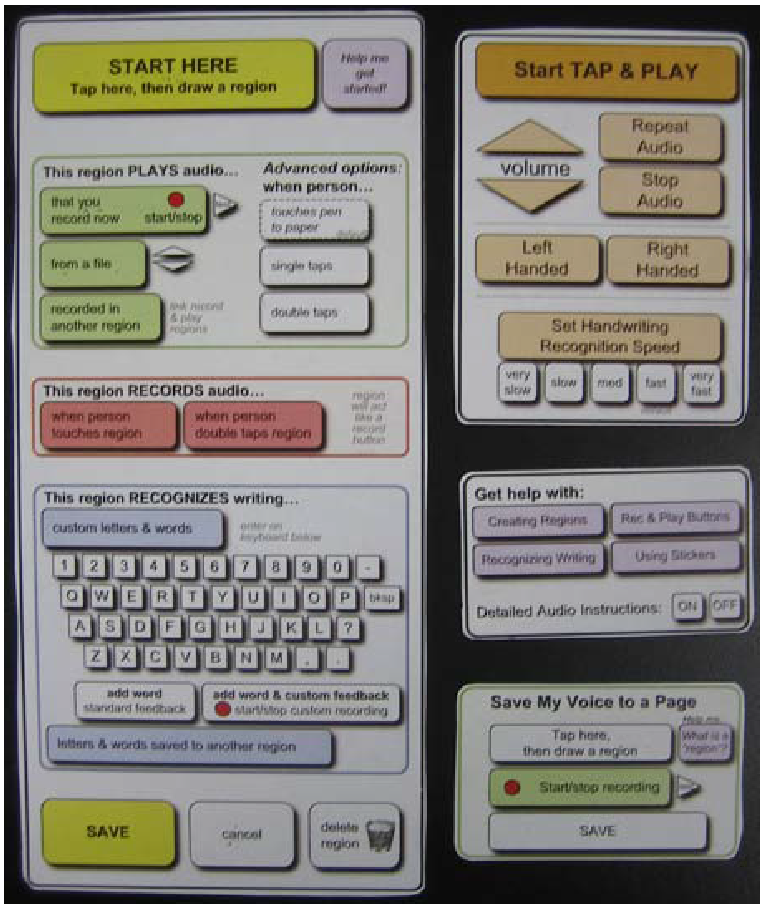

START HERE
Tap here, then draw a region
This region PLAYS audio...
that you record now
start/stop
from a file
recorded in another region
Advanced options: when person...
touches pen to paper
single taps
double taps
This region RECORDS audio...
when person touches region
when person double taps region
This region RECOGNIZES writing...
custom letters & words
add word
standard feedback
add word & custom feedback
start/stop custom recording
letters & words saved to another region
SAVE
cancel
delete region
Start TAP & PLAY
volume
Repeat Audio
Stop Audio
Left Handed
Right Handed
Set Handwriting Recognition Speed
very slow
slow
med
fast
very fast
Get help with:
Creating Regions
Rec & Play Buttons
Recognizing Writing
Using Stickers
Detailed Audio Instructions:
ON
OFF
Save My Voice to a Page
Tap here, then draw a region
Start/stop recording
SAVE

Figure 4.5: TAP & PLAY paper editor mode. From left: Main control panel, general interface control panel (Piper et al., 2012)

The decision to use this approach for authoring digital paper documents, was based on the fact that to develop digital content by using *Livescribe* requires programming proficiency and specific software. (Piper et al., 2012)

4.6 Summary

In this chapter we presented several authoring and digital pen related applications. We looked at Smart Study, an educational platform targeted at primary school children in order to stimulate their handwriting and get automated feedback without teacher interaction. We have seen Write-N-Speak, a toolkit designed to create digital-paper documents for speech language therapists to use while working with aphasia patients. A closely related application to Write-N-Speak developed by the same authors is TAP & PLAY. This is a toolkit used to develop interactive paper documents for language learning activities.

Finally we looked at CHOCOLATO and Algebra-Related Intelligent Tutoring System. These are authoring tools for teachers to create learning materials. Both applications are also accompanied by separate learning modules for students. CHOCOLATO helps teachers create Collaborative Learning (CL) activities by guiding them during the activity creation and providing various learning theories. While the Algebra-Related Intelligent Tutoring System is an authoring tool that is usable for all domains that use algebraic equations.

Next we present the Analysis chapter, where we will have a more detailed look into the technologies we used for the implementation of the authoring tool .

Part III

Development

5 Analysis

This section describes all the analyzed technologies and software needed to develop the authoring tool.

To parse and read the PDF documents we use an open source tool PDFMiner. This tool allows us to successfully analyze the PDF file. To open the pipe between the PDFMiner and our application we use the Hypertext Preprocessor (HTML-embedded scripting language) (PHP) command *popen*, that as a result returns the PDF document written in a PHP string variable in the structure we apply. Next we construct the given string to the objects that we later store in the MySQL database.

For exporting the EPS images we use the open source software ImageMagick command-line tool. This tool provides us all the functionality we need to successfully convert the PDF file to the EPS image file format and afterwards compose it with the "I am ready!" button - that is necessary for the integration with the Smart Study platform.

We also use the Livescribe provided Software Development Kit (SDK) to build the Penlet and the digital document applied with the Anoto dotted pattern. The user is able to import the Livescribe SDK in the Eclipse framework where he can insert the exported EPS images. Inside the Eclipse tool users can mark the pen active areas by dragging and resizing elements from the tool area to the desired places, later this digital paper document can be converted to the Penlet.

In the following sections we give the description of our analyzed technologies that we further use for the application design and implementation. The analysis is later used to determine the limitations of the available technologies. First we describe the PDFMiner tool for PDF document parsing, next the ImageMagick kit to convert the PDF and compose the images. Finally we present the analysis of the Livescribe SDK for building the Paper Design and the Penlet.

5.1 PDFMiner

PDFminer is an open source library developed by Shinyama (2010) to analyze, parse and convert PDF documents. The tool can detect and analyze the document in a tree structure and recognize things like: fonts, spaces, lines and images, blocks and figures as well as the location of them within a page. According to the authors this is the biggest benefit comparing it with other PDF related tools. (Shinyama, 2010; Papathanasiou, 2010)

PDF is not similar to other known document formats like Hyper Text Markup Language (HTML) and Microsoft Office Word, but it is a group of instructions that explains where and how the objects are displayed and placed in the page. As the PDF document is so complex it consumes a lot of time to read it completely. PDFMiner follows the "lazy parsing" strategy meaning it parses the required object when it is needed. *PDFParser* and *PDFDocument* are two classes that are necessary to read the file (see fig:5.1). Figure 5.1 presents the information flow between the PDFMiner classes. The *PDFParser* class is responsible to fetch the data from the file while the *PDF-Document* class holds it. Next the *PDFInterpreter* is in charge to analyze the content and after interpretation the *PDFDevice* class based on the given instructions returns the result. (Shinyama, 2014)

The structured layout of the page and relations between the objects are presented in the figure 5.2. LTPage is the root object and can contain several child objects such as: LTFigure, LTImage, LTRect, LTTextBox and LTLine. LTTextBox element represents a block of text that can contain the list of multiple child elements (LTTextLine). LTFigure contains the object that is a type of PDF Form which is used mostly to present the images by embedding in the document a new PDF file while LTImage represents other image formats i.e. Portable Network Graphics (PNG). LTLine is a straight line and LTRect is a rectangle that can be used for document styling purposes. (Shinyama, 2014)

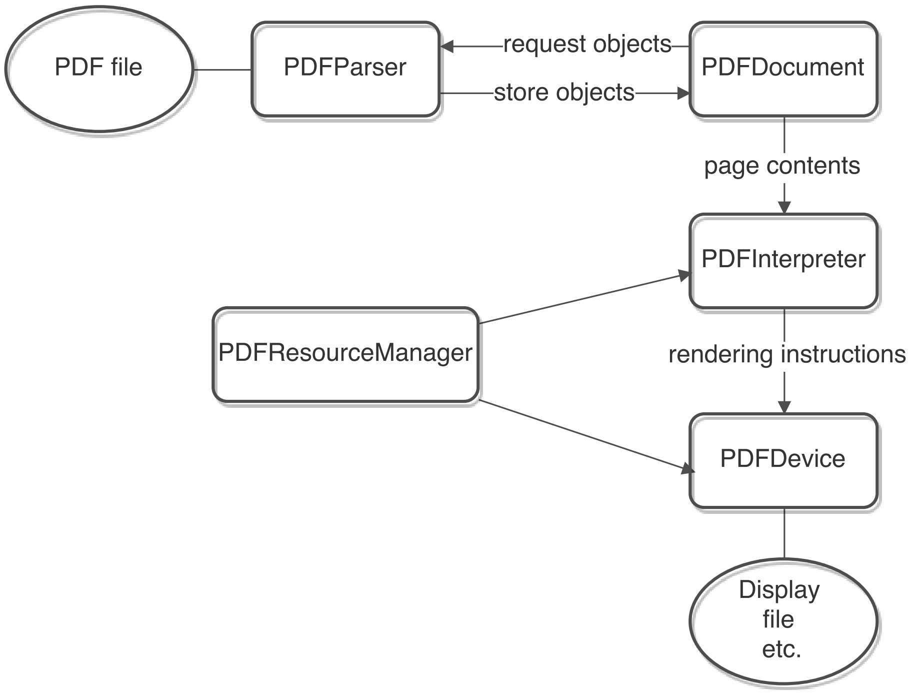


Figure 5.1: PDFMiner: relations between the classes (Shinyama, 2014)

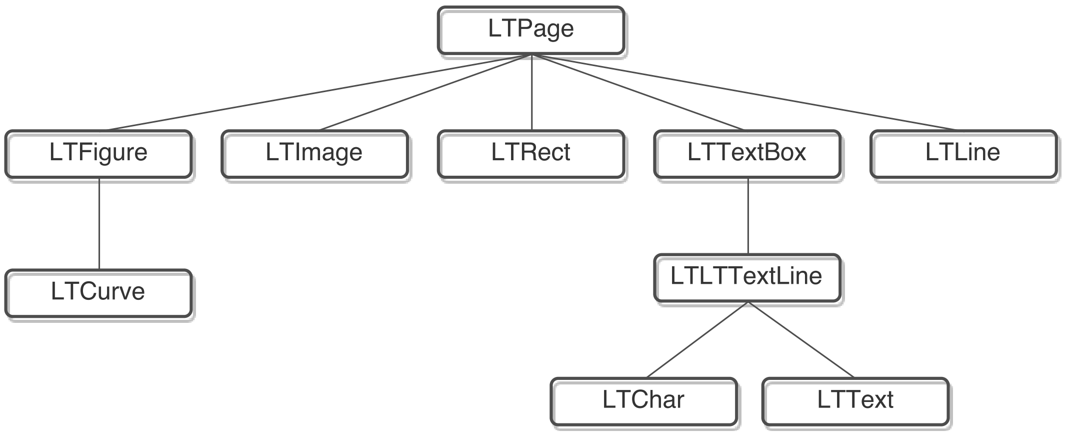


Figure 5.2: PDFMiner: tree structure of the layout objects (Shinyama, 2014)

5.2 ImageMagick

This section will present the open source command-line tool ImageMagick that can be used to convert, edit and display images. The tool is cross-platform and supports read and write functionality on more than 200 file formats. To manage image files ImageMagick provides various functionalities such as: convert, append, mirror, flip, compose, decorate, draw, adjust colors, size, resolution or apply extra effects like lines, polygons etc. In this section we will focus on analyzing functionality that provides image conversion and merge. (ImageMagick Studio LLC, 2016)

With the command *convert* users can not only convert image file format to another, but also crop, resize, drawn on, join, annotate and much more. The first ImageMagick code example presented in listing 5.1 simply shows how to write an image to another file format. The last attribute of all the presented listings determinate the result image. (ImageMagick Studio LLC, 2016)

All images used in this section are from personal resources.

Listing 5.1: Convert file format

```
convert tree.jpg tree.png
```

Listing 5.2 presents the crop functionality. It shows how the attribute -*gravity* can be used. Figure 5.3 illustrates the original image 5.3a and cropped image 5.3b. By using the gravity attribute we can orient the region for action to specific place on the image, e.g. Center (default) , South, West, North, East as well the logic combination of them such as: NorthWest and SouthEast (ImageMagick Studio LLC, 2016). In this example, the -gravity attribute to South and image dimension is set to 640x640.

Listing 5.2: Crop image

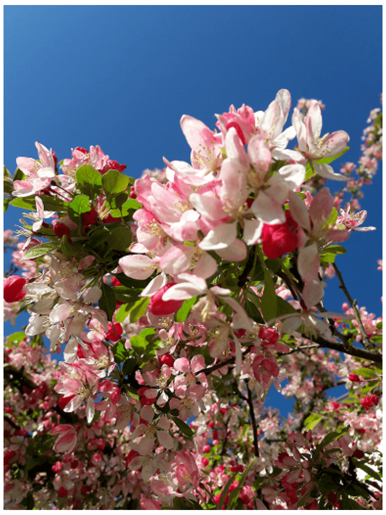
```
convert tree.png -gravity South -crop 640x640+50+50 flower.png
```

The example shown in listing 5.3 and figure 5.4 illustrates how to rotate an image for 45° and also how to add borders. The variable 45 of the attribute rotate determinate the rotation radius. *bordercolor* feature explains itself - the color of the border.

Listing 5.3: Rotate image

```
convert flower.png -rotate 45  -bordercolor red
-border 0x10 flowerRotate.png
```

The next example (see listing: 5.4) presents the append operator, that simply joins the images with no spaces between them. -*append* option appends images vertically while +*append* does it horizontally (see fig: 5.5). The listing 5.4 presents image appending and also how to resize image by using


(a) Original image (tree.png)

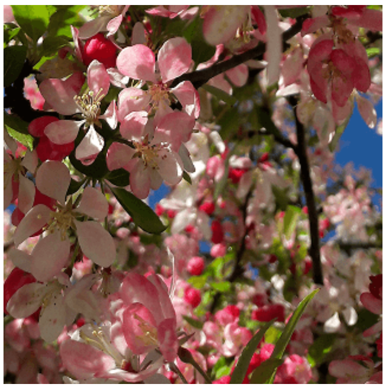
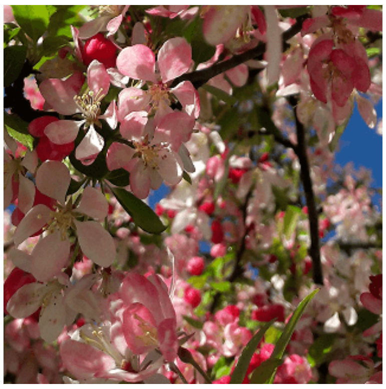

(b) Cut image (flower.png)

Figure 5.3: ImageMagick command *cut* with geometry and gravity attributes


(a) Original image (flower.png)

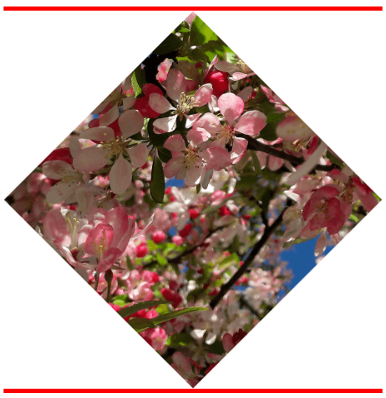

(b) Rotate image 45° (flowerRotate.png)

Figure 5.4: ImageMagick command rotate 45°

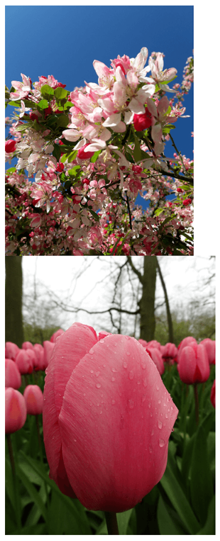
percentage. The original images size is: tree.png 1.74 MB, tulip.png 1.58 MB and the result image (see fig: 5.5a) size is 39.3 KB.

Listing 5.4: Append images

```
convert tree.png tulip.png -append -resize 10% appendImages-.png

convert tree.png tulip.png +append appendImages+.png
```

The listing 5.5 presents how to create a label by providing the string


(a) Append images vertically

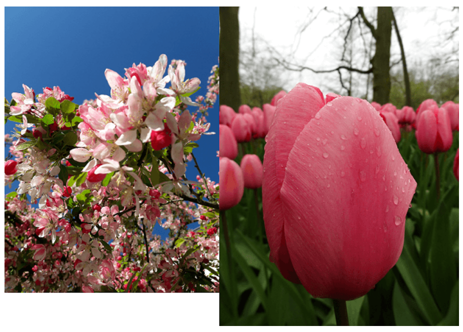

(b) Append images horizontally

Figure 5.5: ImageMagick command append

and setting attributes like background color and border. The result of this command is presented in the figure 5.6b.

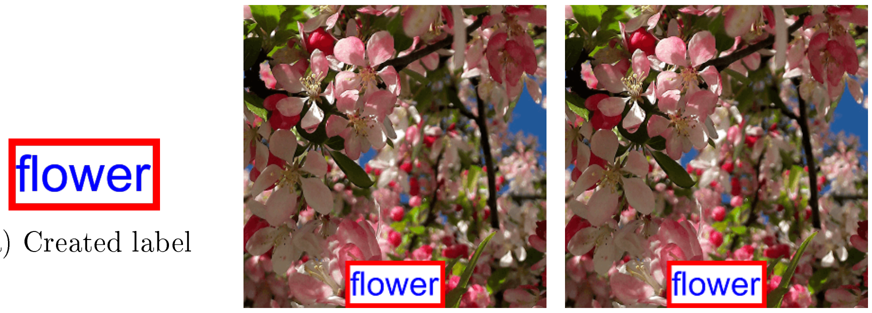
Listing 5.5: Create label

```
convert -background white -fill blue -size x80
-bordercolor red -border 10x10 label:flower label.png
```

Next we present how to compose the label.png with the previously cropped image (see figures: 5.6b and 5.3b). In listing 5.6 we illustrates two ways how to compose the images by having the same attributes and the result is shown in the figure 5.6. In this example we also use the gravity attribute that specifies how to join both images.

Listing 5.6: Simple convert command

```
composite label.png -gravity South flower.png compose.png
# OR
convert flower.png label.png -gravity South
-composite convertCompose.png
```



(a) Created label (b) composite compose.png (c) convert -composite convertCompose.png

Figure 5.6: ImageMagick composite

5.3 Livescribe Platform

Using the Livescribe platform we can develop Pen Applications for the Livescribe smartpen. As it is presented in the figure 5.7 the Pen Application consists of two main components: the Paper Product and the Penlet. Both components and the Livescribe SDK are explained in the following sections.

Penlet + Paper Product = Pen Application

Figure 5.7: Pen Application (Livescribe Inc., 2009)

5.3.1 Livescribe SDK

The Livescribe SDK contains a set of development tools, Livescribe Platform API, documentations and code samples. It also provides a plugin and features for the Eclipse platform, to develop smartpen applications. The complete setup of the Interactive Development Environment (IDE) requires the Eclipse platform (Ganymede or higher - 32 bit version) with the Livescribe plugin, the Java Development Kit (JDK) and GhostScript. GhostScript is required for Eclipse to import the EPS images. The Livescribe plugin for Eclipse contains two main components: the Paper Design to create a paper project and the Penlet to develop penlet projects.

5.3.2 Penlet

The Penlet is a Java application that is developed for a Paper Product. In the penlet application we can define methods for the events (triggered by the smartpen) that are related to active regions on the paper. The figure 5.8 presents the lifecycle methods of the penlet application.

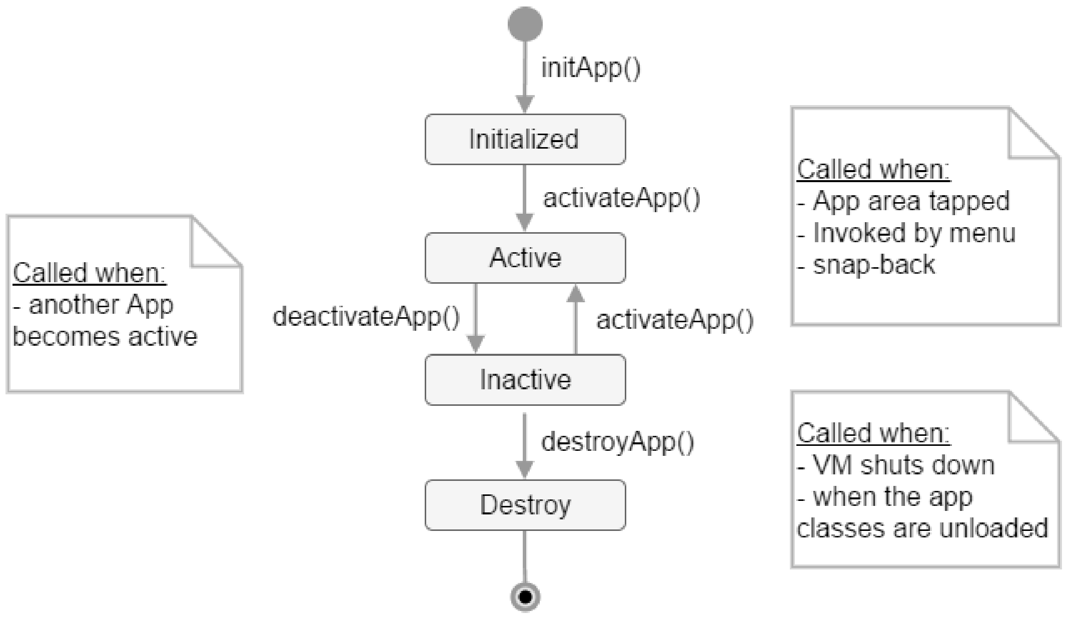
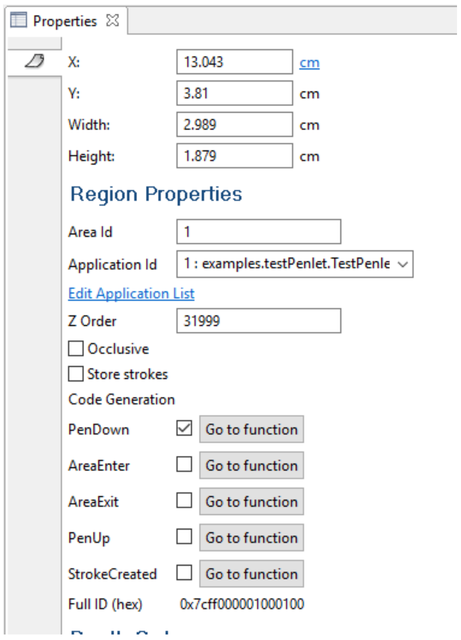


Figure 5.8: Penlet States (Livescribe Inc., 2008)

When the new penlet project is created all the lifecycle methods are already presented and filled with the main functionality. To make the regions active and recognizable for the penlet application we need to introduce new events. To that end we can auto-generate event code from the Paper Project, by selecting a region and from the properties view select the event type and click the button "go to function" (see fig: 5.9). Next, figure 5.10 presents the penlet project with generated functions.

5.3.3 Paper Product

The Paper Product consists of two components: the physical dot paper and a file representation in digital format. The physical dot paper is a paper printed with the dot pattern that the user can use with the smartpen. This paper can be modeled as open or fixed print. The open print is designed so various smartpen applications can use it, while the fixed print refers to


Properties

X: 13.043 cm
Y: 3.81 cm
Width: 2.989 cm
Height: 1.879 cm

Region Properties

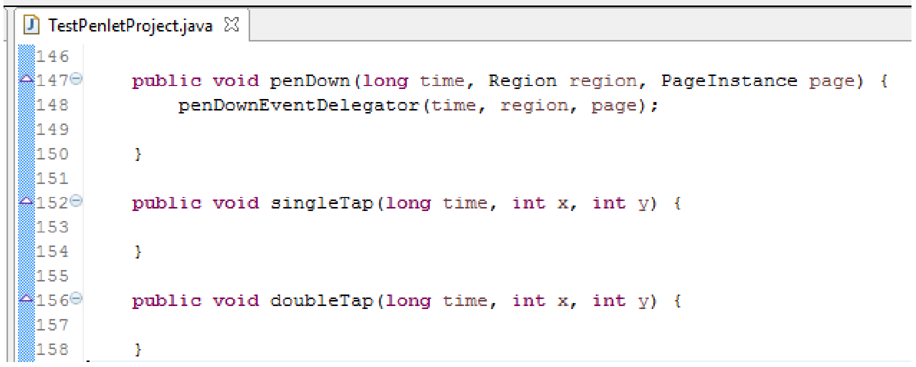
Area Id: 1
Application Id: 1 : examples.testPenlet.TestPenle
Edit Application List
Z Order: 31999
☐ Occlusive
☐ Store strokes

Code Generation

PenDown	✓	Go to function
AreaEnter	☐	Go to function
AreaExit	☐	Go to function
PenUp	☐	Go to function
StrokeCreated	☐	Go to function

Full ID (hex): 0x7cff000001000100

Figure 5.9: Paper Design - Auto-generate code for *PenDown*


TestPenletProject.java

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

}

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

}

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

}
```

Figure 5.10: Penlet - Generated methods

specific region on the Anoto dot paper. These regions are introduced when the developer develops the pen application for the smartpen. Each paper project can be linked or associated with one or more Penlets. (Livescribe Inc., 2009)

The AFD is the digital representation of the paper product. The AFD is the part of the Paper Product that describes each of the paper components (paper attributes, dot space, images) to the Penlet. The AFD file along with the Penlet are stored on the smartpen and helps the pen identify the Paper Product. (Livescribe Inc., 2009)

The Livescribe SDK introduces to the Eclipse a new perspective called "Paper Design". Using this perspective we can build a new Paper Products. The figure 5.12 presents the perspective. The paper design contains various views:

- Thumbnails: in this view all created pages are presented (see fig: 5.11)
- Outline: this view list all graphical elements and active regions (see fig: 5.12)
- Properties: this view will present all available properties for the selected item (see fig: 5.12)

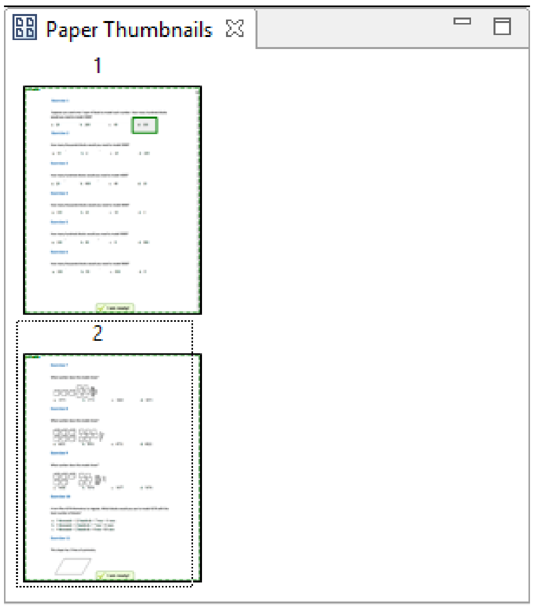
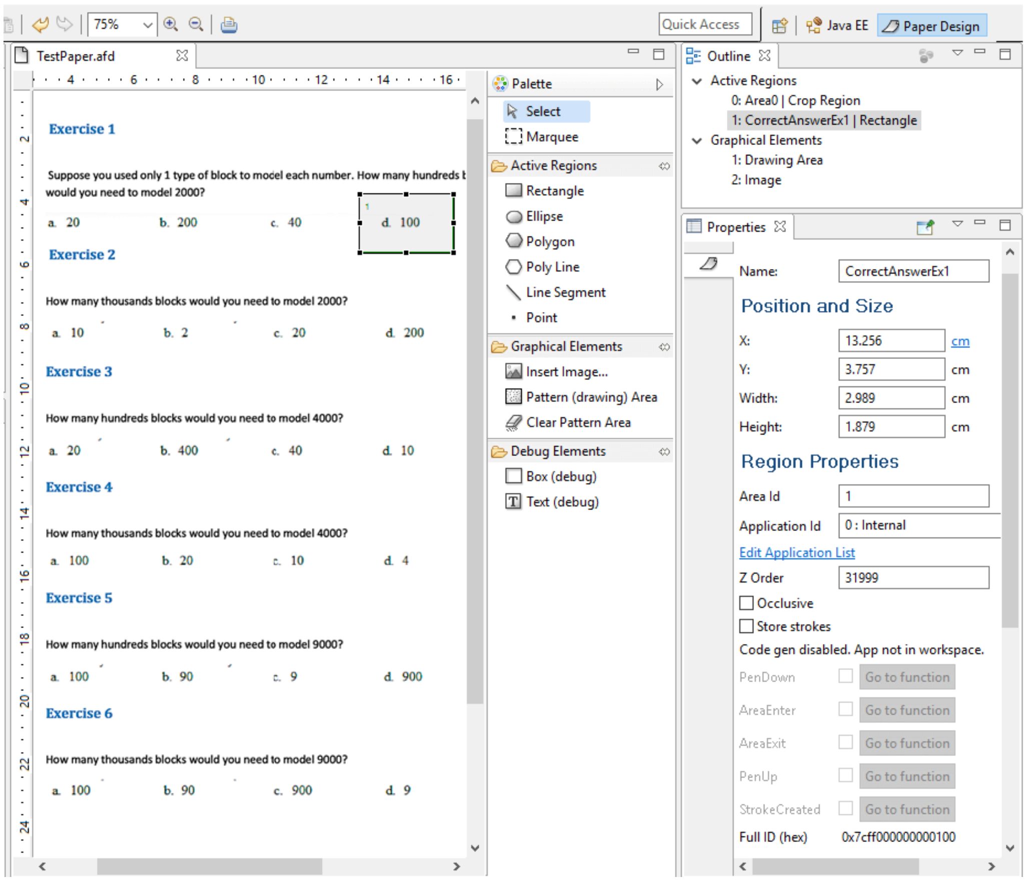


Figure 5.11: Paper Design perspectives

To create an active region on the current page we need to select the shape from the active regions folder in the palette and drag it to the desired place. After the region is placed on the page we can adapt its size. Note, that the area id of this active region (see fig: 5.12) links this region to the penlet (as well as it refers to the exercise id of the database entity *Exercise*).



Exercise 1

Suppose you used only 1 type of block to model each number. How many hundreds b would you need to model 2000?

a. 20 b. 200 c. 40 d. 100

Exercise 2

How many thousands blocks would you need to model 2000?

a. 10 b. 2 c. 20 d. 200

Exercise 3

How many hundreds blocks would you need to model 4000?

a. 20 b. 400 c. 40 d. 10

Exercise 4

How many thousands blocks would you need to model 4000?

a. 100 b. 20 c. 10 d. 4

Exercise 5

How many hundreds blocks would you need to model 9000?

a. 100 b. 90 c. 9 d. 900

Exercise 6

How many thousands blocks would you need to model 9000?

a. 100 b. 90 c. 900 d. 9

Active Regions
- 0: Area0 | Crop Region
- 1: CorrectAnswerEx1 | Rectangle

Graphical Elements
- 1: Drawing Area
- 2: Image

Name: CorrectAnswerEx1

Position and Size

X:	13.256	cm
Y:	3.757	cm
Width:	2.989	cm
Height:	1.879	cm

Region Properties

Area Id: 1
Application Id: 0 : Internal
Edit Application List
Z Order: 31999
☐ Occlusive
☐ Store strokes
Code gen disabled. App not in workspace.
PenDown ☐ Go to function
AreaEnter ☐ Go to function
AreaExit ☐ Go to function
PenUp ☐ Go to function
StrokeCreated ☐ Go to function
Full ID (hex) 0x7cff000000000100

Figure 5.12: Paper Design perspectives

5.4 Summary

In the chapter above we have taken a first look at the technologies we used in the authoring tool implementation. We have seen the Python library, used for PDF document parsing and image extraction. We also have seen ImageMagick, a command-line tool used to convert, compose, compress, edit, display, etc. various file formats.

Finally, we looked at the Livescribe Platform for Pen Applications development. The Pen Application consists of two main components: the Paper Product and Penlet, both of which weâ€™ve seen in this chapter.

Next we present the Design chapter, where we will have a more detailed look into user classes, functional requirements, user interface, database design as well as the architecture of the authoring tool and explain the relations between its components.

6 Design

In this chapter we give an overview of the system design and explain the relations between the authoring tool and Smart Study application. Figure 6.1 presents the general concept. Overall the system consists of the global content repository and three main layers to cover presentation, administration and content authoring processes.

The presentation layer is part of Smart Study, while the content creation layer covers the authoring tool developed for this master thesis, and additionally there is a part of the administration. As illustrated in figure 6.1, the presentation, as well as the content creation layer, requires a common repository to post or get necessary information. For this purpose the system contains a global repository, which plays the information retrieval and storage role.

6.1 User Classes

As mentioned in the introduction, there are two separate applications that work by sharing data from a common repository – the Smart Study application and the authoring tool. Each of them is designed for a specific type of user: students or teachers/authors. The following section describes and contains the specification of user classes for both user types.

User classes explain which level of experience and knowledge users should

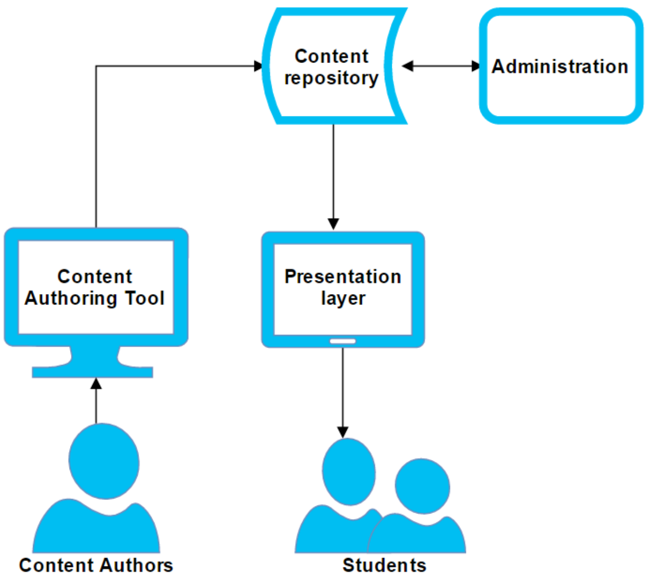


Figure 6.1: The concept of the authoring tool and SmartStudy application. (Blackhall, 2011)

have, what kind of skills are necessary for them to use the system, their motivation and how much training is required.

The teacher/author class describes a primary user for the authoring tool. He is expected to use the application either occasionally or on a regular basis. We expect a training period of 2 hours or less for a user to be able to use the application properly. To successfully create the digital exercise paper the user should have the knowledge and skills equivalent to those of an advanced beginner.

Teacher/Author (for Authoring tool)

- Type of user: Primary user
- Task Experience: Advanced beginner to Competent performer
- Frequency of use: Occasionally to regular
- Level of task knowledge: Average
- Type of device: Personal computer, laptop
- Use: Mandatory

- Existing computer experience/skills: Advanced beginner to Competent performer
- Other systems that will be used concurrently: None (Digital Pen)
- Training: Without training or with 1 up to 2 hours
- Motivation for using system: To create digital exercise paper using existing materials(files).

The student class describes a primary user for the Smart Study application. To use the application the user requires a digital pen and a device (tablet, personal computer or laptop). Similar to the author, a student should have the computer skills and experience equivalent to an advanced beginner.

Student (for response)

- Type of user: Primary user
- Task Experience: Advanced beginner
- Frequency of use: Occasionally to regular
- Level of task knowledge: Average
- Type of device: Digital Pen, Tablet (or personal computer, laptop)
- Use: Mandatory
- Age: Pupil in the 4th to 6th class
- Existing computer experience/skills: Advanced beginner, General use skills of required devices
- Other systems that will be used concurrently : Digital Pen, Tablet(or personal computer, laptop)
- Training: None
- Motivation for using system: To learn (to do exercises and check answers) using modern and motivational technologies.

6.2 Requirements

To implement a usable application we started with the creation of a Use Case model and translated it to the functional requirements that are explained in this section.

6.2.1 Use Case Model

The use case model helps to define, analyze and organize the requirements of the application. The model presents high level interactions between the user and the tool as well as giving a great guidelines of the functional requirements determination. (Rouse, 2007)

The figure 6.2 illustrates the main event flow within the application and as it can be seen in the figure the main goals of the authoring tool is to allow the user to create the digital exercises as well export and print them. The use case model also presents multiple other user actions, such as upload or open the PDF file and save, edit or add exercise content (question/answer/-explanation).

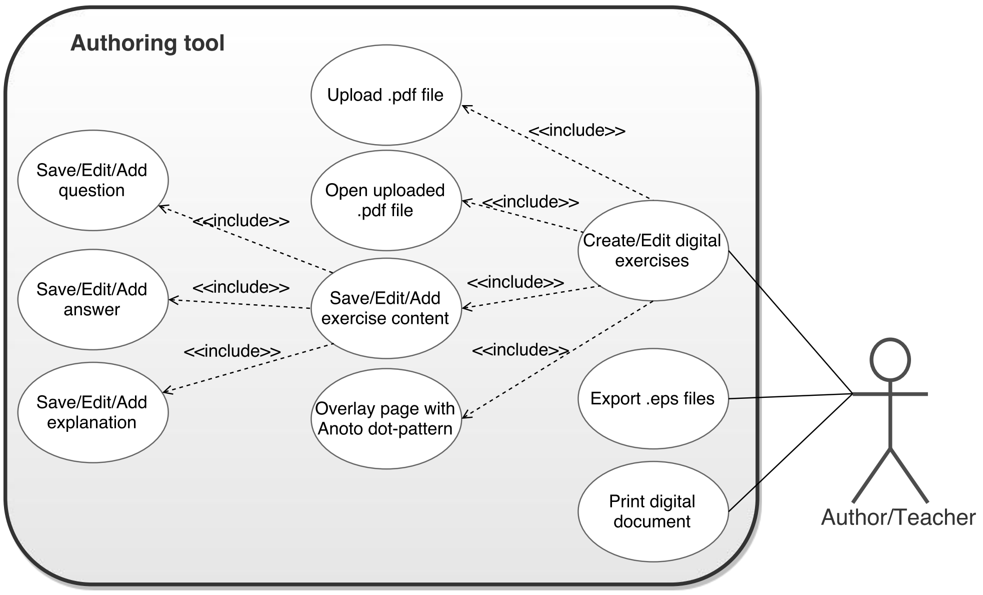


Figure 6.2: UC model

6.2.2 Functional Requirements

The functional requirements of the authoring tool are listed below, and they explain the expected behavior of the tool. Some associated usability requirement are also formulated.

1. Allow user to upload files (PDF format)
    - User should not wait more than 30 seconds to upload a file
2. Allow user to open uploaded files
    - User should not wait more than 10 seconds until the file opens
    - User should open the file just clicking on it
    - User should be able to edit previously saved exercise content (including question, answer and specified explanation or formula/s)
3. Allow user to enter correct answer
    - User should be able to enter correct answer
    - User should be able to edit or delete the content
4. Allow user to specify formula/s for entered answer
    - User should be able to specify formula/s
    - User should be able to edit or delete specified formula/s
5. Allow user to save any changes
    - User should not wait more than 10 seconds until all changes are saved
    - User should be able to cancel changes
6. Allow user to close the file (with or without saved changes)
    - User should receive a warning message in the case of exiting without saving
7. Provide an overview of all uploaded files
    - Files should be ordered alphabetically by name
8. Allow user to export file
    - User should not wait more than 30 seconds while file is exported
    - User should to be able to cancel the export process
    - User should be able to store the exported files anywhere he prefers

9. Allow user to mark a pen active area
    - User should be able to drag and drop the pen active area all over the active page
    - User should be able to reduce and increase the size of the pen active area
    - User should be able to delete the active pen area

6.3 Architecture

The information flow of the authoring tool can be seen in figure 6.3. The upper part illustrates the fundamental operations and how the information is send and retrieved between the parts of the system. The figure also shows that the connection between two components is different (the EPS file and Eclipse platform) due to the limitations of the *Livescribe SDK*. In the following section we explain all the relations between the authoring tool components.

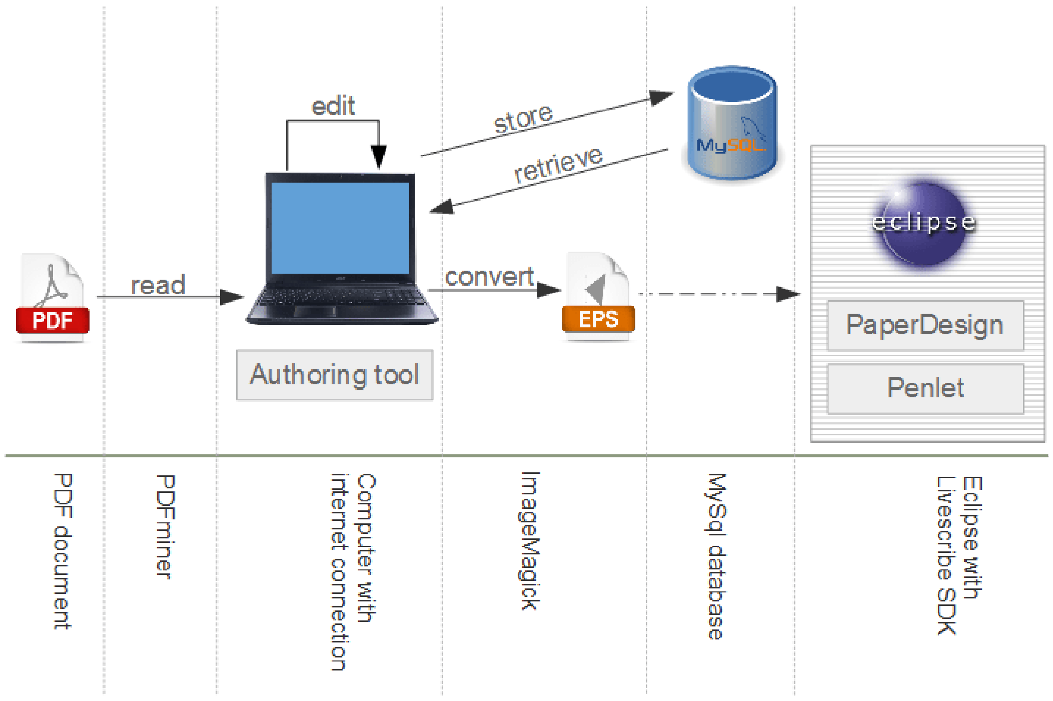


Figure 6.3: Information flow model of the authoring tool

6.4 Relation Between the Components

In this section we describe the working flow of the system, relations and information exchange between the components and we characterize each component and its role in the system. The following sections 6.4.1, 6.4.2 and 6.4.3 are written according to the scenario.

6.4.1 Scenario Server Setup

The authoring tool is a web browser application, therefore it needs to be hosted on a server with an Internet connection. Similar, the MySQL database should be stored on the server. As the application is using Python for the PDF parsing, the server needs to have installed the 2.4 version of Python or newer (the 3^{rd} version is not supported) (Shinyama, 2010). In addition to the installation of Python, the system environment variable *path* needs to be updated with the Python home folder. All the images that are read from the PDF file are encoded in PNG file format, to this end the server needs to have the *PyPNG* Python library installed.

6.4.2 Scenario Platform Setup

To start working with the authoring tool, a user needs ensure that everything is set up. For the environment to be set up, the user needs to have a workstation (laptop, personal computer, etc.) with a Windows operating system installed (Windows Vista or newer) and active Internet connection. On the Windows computer the Eclipse software (Ganymede or higher version (32 bit)) needs to be installed. Additionally, the JDK and Java SE Runtime Environment (JRE) need to be configured. For the Eclipse software to be able to present the EPS image file format the GhostScript needs to be installed and configured in the platform. The PDF document needs to be stored on the computer and in addition to that the machine needs to have an Internet browser installed.

6.4.3 Scenario Working With the Platform

After the environments are set up the user can start to interact with the application. To that end the user turns on his computer and opens the web browser. Active Internet connection is required. The user navigates to the authoring tool. When the application is loading two lists from the database are selected. The first list consists of files that are uploaded on the system but the content of them is not parsed and stored in the database (*uploaded files* *list*), while the other presents all files whose content is modified and stored (*stored files list*). After the application is loaded in the browser it shows the main screen, that presents the document upload form and both lists. When the user chooses to upload the PDF file, he clicks on the file upload widget, picks the document from the computer and clicks the button to upload it. The system triggers the event from the form, copies the file on the server and inserts the information about it on the database table *File*. After the PDF document is successfully stored on the server, it is also presented in the uploaded files list. The system is ready now to parse the document and present its content to the user on the screen.

When the user chooses to open the uploaded document, the system asks to the user to provide the phrase with which each exercise starts or leave it blank. After the user enters the phrase the system executes the PDFMiner code written in Python that parses the file. The name and the location of the picked document are provided to the PDFMiner. The PDFMiner detects and recognizes all written text and images. All find images are encoded in the PNG file format and are stored on the server. After the parsing, PDFMiner process returns to the system PDF file wrote in the string. Next, out of the returned string the authoring tool generates the array with *Page* objects that contains *Exercise* objects. Each image is allocated under the exercise object it belongs to. At this moment the *Page* object contains only the logic ordering number and the list of exercises, while the *Exercise* object stored images list, question, ordering number and the related page number.

Next the system presents to the user the read content of the file. The authoring tool shows each page separately. The exercises are visually separated and are presented in the same order as they appeared in the document. Next, when the user chooses the course and category for the page from the drop-down menu, the system triggers the event and stores the provided information in the current *Page*. All the actions that are related to the exercise content editing (question, answer and explanation) trigger the events that changes the *Exercise* object content to the actual one (the one from the screen). All the other actions such as 2 exercise merge, create or remove the exercise are fetch by the events that calls methods that are based on the provided action changes the hierarchy of the *Page* object. During the process of these actions the application screen is dimmed and a busy indicator shown, afterwards the system presents the feedback to the user by showing either a successful or error notification. After all the necessary modifications are done, the user can save the content. The system builds the insert statements with the foreign keys to related entities for each page, exercise and image. Next the system presents the main screen, but now the previously uploaded

PDF document is presented in the list of stored files.

When the user chooses to open a document from the stored files list, the system fetches all information about the picked document from the database and generates the *Page* and *Exercise* objects. After the objects are created the content of the file is presented on the screen. Similarly as with the uploaded file editing, when the user changes the text of the exercise (question, answer and explanation), the system triggers events that applies the changes to the related object content, as well as the process that sets the flag *changed* to true for the modified exercise. The merge, edit and delete actions of the exercise are processed differently as it is with the uploaded files. When the user picks the exercise to delete, the system hides the exercise content from the screen and sets the flag *isRemoved* to true for the removed exercise. Similar action happens when the user chooses to add a new exercise. The system shows the created exercise on the screen and sets the flag *isNew* to true. The user merge two exercises by dragging and dropping the exercise to the target one. After the user drops the exercise the authoring tool asks the confirmation. The user can cancel the merge action or confirm it by clicking *ok*. When the user confirms the merge process, the system triggers the content change event to the target exercise and marks the dragged exercise as removed. Similarly as for the uploaded files, the application shows that the system is in process and presents the notification after the successful or fail action. After the user finishes the content editing he can cancel or save. When the user choose to save the changes, the system generates the insert delete and/or update scripts. The flags on the *Exercise* object are now used to determine the correct action of the query. The flag *isRemoved* causes the delete query, *isNew* insert and *changed* update. After all the queries are successfully executed on the database the authoring tool presents the main screen to the user. The user at anytime can open either uploaded or stored PDF file and at any point in viewing or editing he can cancel (exit) or save the made changes.

After the modifications of the stored file the user can choose to convert the document to the EPS file format.fir this, he clicks on the *EPS* link next to the stored file name. The system triggers the action and starts the conversion process. To that end the application first converts each page of the PDF file to the EPS by using the ImageMagick command-line tool. After the conversion the EPS image is stored on the server. The following action is to append the page image (EPS) with the footer and button "I am ready!". This button is necessary for the Smart Study application and by tapping on it with the Smartpen the user notifies the application that he finished filling the exercises in the current page (Van Thienen et al., 2015). To do the append operation the command *compose* is executed for each page image. All the conversion and compose commands are executed from the PHP file inside the system. When all the images are ready, the system compresses them into the folder. While the conversion and compression processes are active the screen of the authoring tool is dimmed and the busy indicator is shown. When the processes are done the system streams the compressed file to the user. The user can choose to save them on his computer.

6.5 User Interface

In this section we present the UI design of the authoring tool. We describe and illustrate the mock-ups of the screens and forms.

The color schema and Cascading Style Sheets (CSS) components were created based on the open source Zurb project Foundation. This framework also provides responsive grid, button and from templates besides JavaScript (JS) extensions. (Zurb, 2015)

6.5.1 The Authoring Tool Screen Design

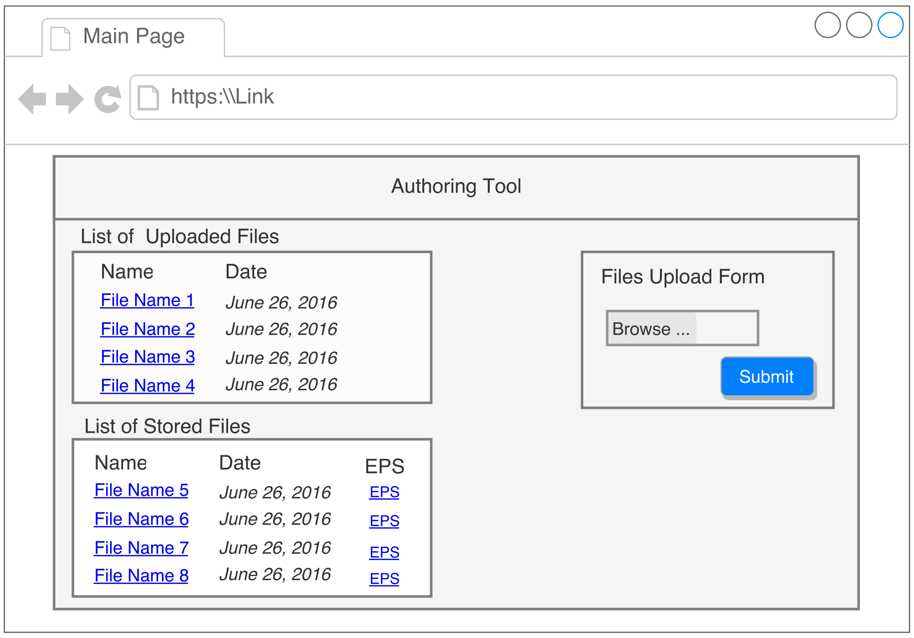
All the screens in the application have a similar background - image containing two main colors (blue and green) that symbolizes learning and intuition. The blue color itself represents stability, trust and intelligence, while the green color stands for harmony, safety and growth as well as it is characterized as emotionality positive and the best color for eyes to rest. These two colors are not only used for the application background, but also for the forms and buttons. The neutral and practical tint gray is also present in the authoring tool. (Olesen, n.d.)

Main Screen

The main screen contains tree major blocks: list of uploaded files, list of stored files and file upload form. Both lists of the files present the file name (title) and the date of the file upload. Besides that the stored files list contains the link to the *EPS* file download.

The file upload form consists of the form name and two inputs: the file browser and submit button. When the user choses the file to upload it is display in the form. After the user clicked the submit button the application uploads the file and by refreshing the screen illustrates it in the uploaded files list. (see fig: 6.4)

The top part of each view of the application illustrates the name of the authoring tool inside header component.



Main Page

https:\\Link

Authoring Tool

List of Uploaded Files

Name	Date
File Name 1	June 26, 2016
File Name 2	June 26, 2016
File Name 3	June 26, 2016
File Name 4	June 26, 2016

List of Stored Files

Name	Date	EPS
File Name 5	June 26, 2016	EPS
File Name 6	June 26, 2016	EPS
File Name 7	June 26, 2016	EPS
File Name 8	June 26, 2016	EPS

Files Upload Form

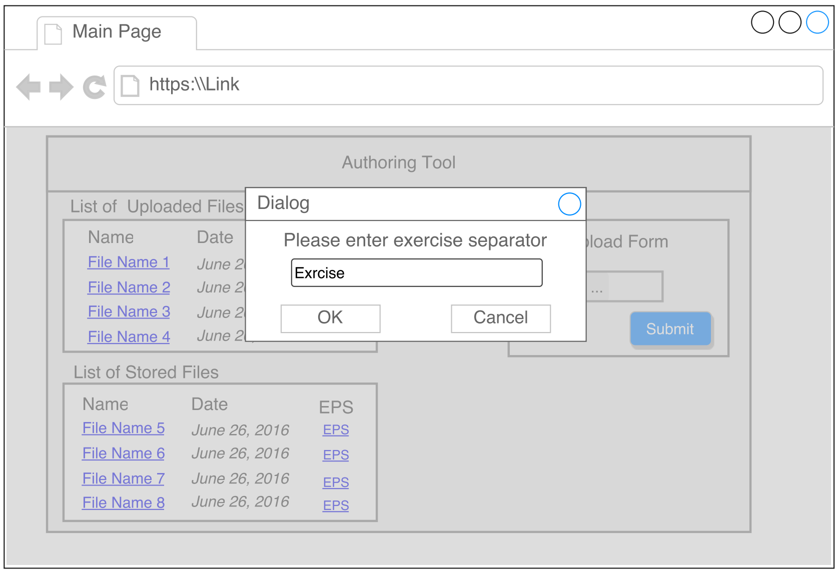
Figure 6.4: Man Page design

If the user chooses to download the file converted to the EPS, while the action is progressing, the screen is dimmed and busy indicator is displayed. Next the browser presents the file download form and the user can store the compressed folder in the desired location on his computer.

When the user clicks on the file name the system displays the dialog window. This pop up contains the text "Please enter exercise separator", input field and two buttons: *ok* and *cancel*. The user can cancel the file opening action by clicking button *cancel*. As well, the user can continue the file opening action, with the exercise separator or blank, by pressing the button *ok*. The dialog window is shown in the figure 6.5.

The screen of the application is dimmed and the busy indicator is displayed when the process of the file opening is started. The screen with the busy indicator is presented in the figure 6.6.

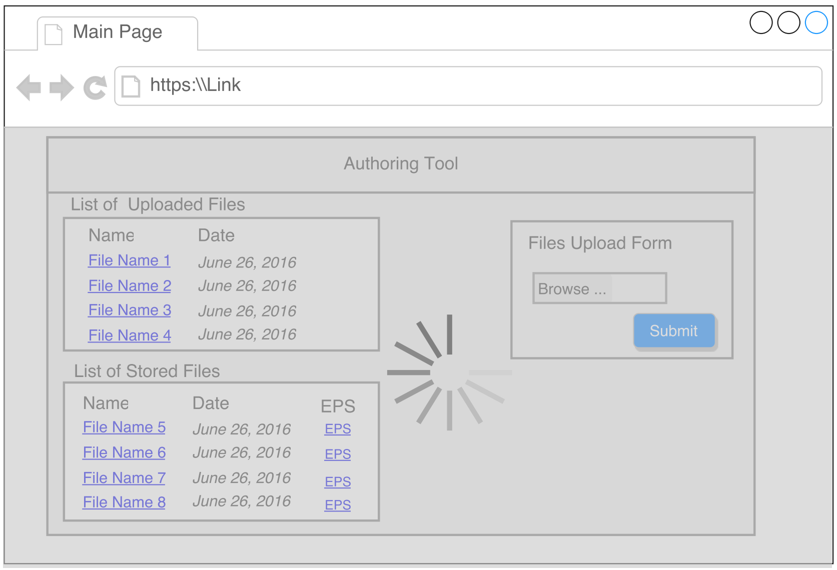
After the successful PDF opening process the system presents the file edit view. If the user chooses to open the uploaded file the system will parse the document and present its content on the file edit view. Similar as with the uploaded file, the content of the stored file will be shown in the edit view, but the data fetched from the database. (see fig: 6.4)



Name	Date
File Name 1	June 2
File Name 2	June 2
File Name 3	June 2
File Name 4	June 2

Name	Date	EPS
File Name 5	June 26, 2016	EPS
File Name 6	June 26, 2016	EPS
File Name 7	June 26, 2016	EPS
File Name 8	June 26, 2016	EPS

Figure 6.5: Main page with exercise separator dialog design



Name	Date
File Name 1	June 26, 2016
File Name 2	June 26, 2016
File Name 3	June 26, 2016
File Name 4	June 26, 2016

Name	Date	EPS
File Name 5	June 26, 2016	EPS
File Name 6	June 26, 2016	EPS
File Name 7	June 26, 2016	EPS
File Name 8	June 26, 2016	EPS

Figure 6.6: Busy Overlay design

Editing Screen

The figure 6.7 shows the file edit view. This screen consists of three major blocks: page header, footer and page content.

If the PDF document has multiple pages the system presents at the left side the button « to move to the previous and at the right side » to move to the next page. (see fig: 6.7)

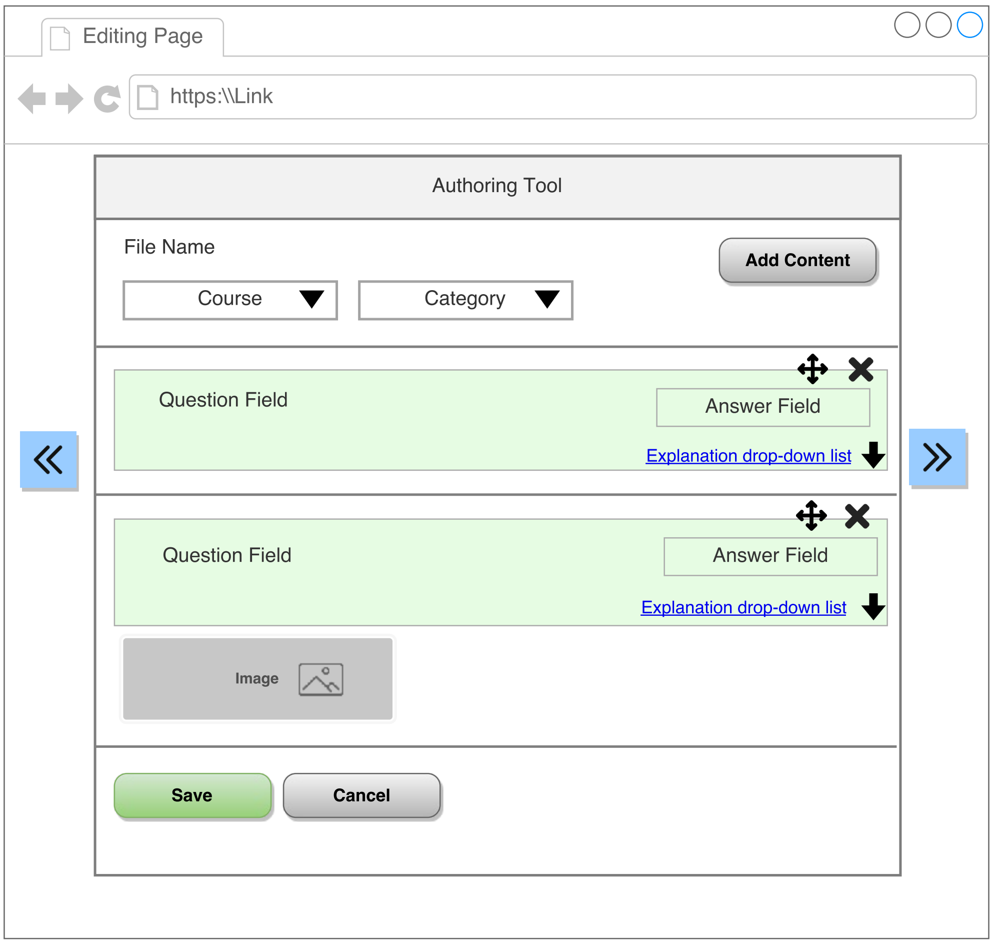


Figure 6.7: Exercise page design

The main block presents the page content. This part of the screen displays all the exercises and images from one PDF file page. The exercises are illustrated in the same order as they are in the document. All the images are presented under the related exercise content. For each of the exercises the user can define its question, correct answer and the solution. The answer field is placed on the right side in the exercise content while the explanation field is hidden, to open it user should press the explanation drop down list link, the same applies to close it.

The user can remove exercises by pressing the delete icon ✖. When the user chooses to remove the exercise, the system presents the alert window with the message "Delete?" and two buttons: accept the action (ok) and stop (cancel). If the user chooses to delete the exercise, the system removes it from the screen and presents the successful notification message on the top right corner of the screen. (see fig: 6.8)

To display all the notifications in the authoring tool we use the *Notify.js JQuery* plugin which provides custom notification messages (Pillora, 2014). All the notifications are displayed in the right top corner of the screen.

Successful message
Error message

Figure 6.8: Successful and Error notification message

The user can merge two exercise contents by dragging on of the exercise to the target one. To drag the content user should use the drag symbol ✣ that is located at the top right corner of the exercise. When the dragged exercise is dropped in the target one, the system presents the alert window with the message "Confirm exercise merge action" and two buttons: *ok* to confirm and *cancel* to cancel. If the user cancels the action the screen content remains the same, but if the user confirms, the system concatenates both questions and displays related images below the target exercise. After the merge process, the successful or error notification is displayed. An exercise merge process example can be seen in the figure 6.9.

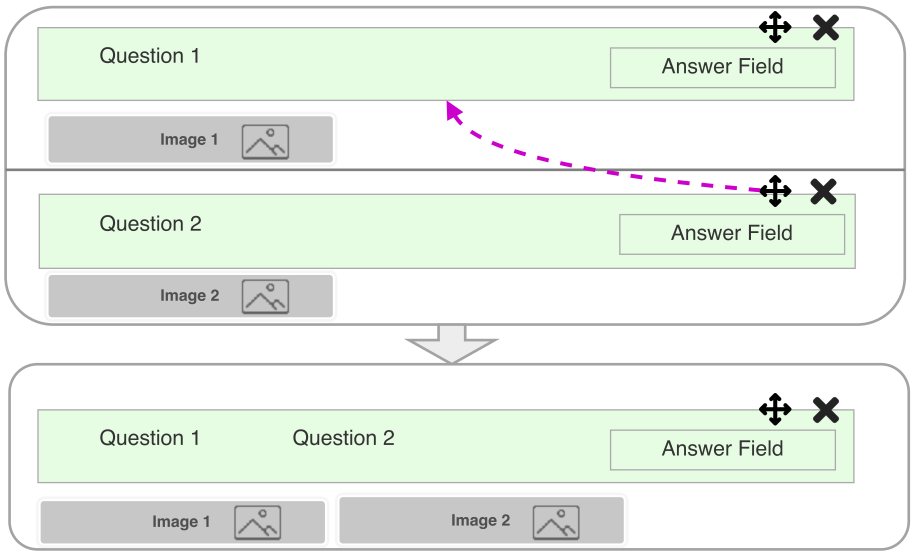


Figure 6.9: Merge two exercises

The page header block contains the file name, two drop down lists (*course*, *category*) and the "Add Content" button (see fig: 6.7). The drop down lists allows the user to characterize the page by giving it a course name and category. This characterization applies to each page individually. The design of the drop down lists is illustrated in the figure 6.10.

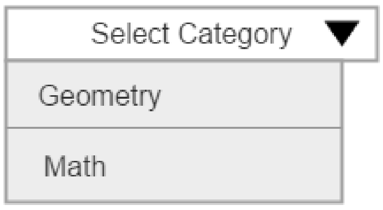


Figure 6.10: Exercise page design

When the user presses the Add Content button the system presents all available places where the new exercise can be added (see fig: 6.11). To cancel the action the user should press the Add Content button a second time. To add the exercise, the user should choose the desired location and click on the "add here" box. After the user clicks on the "add here" box, the system presents the new exercise at the chosen location and displays the notification message.

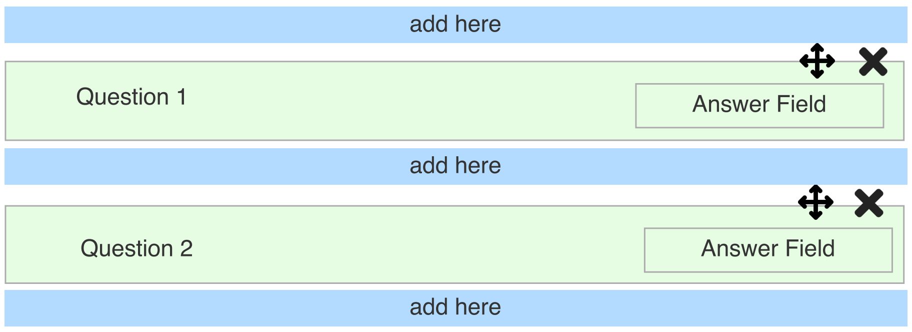


Figure 6.11: Add exercise design

The footer block contains two buttons: *save* and *cancel*. The *cancel* button always revokes all made changes and displays the main application screen, while the *save* button saves all modifications. The modifications include:

- add, remove or merge exercise
- changed exercise content
  - question

  - answer
  - explanation

- page course or/and category change

While the application is saving, the screen is dimmed and the busy indicator is displayed. After all modifications are saved the system displays the main application screen and the status notification.

The Screen navigation

Figure 6.12 presents the sequence between the *Main* and *File Edit* screens as well as the expected alerts and dialog windows.

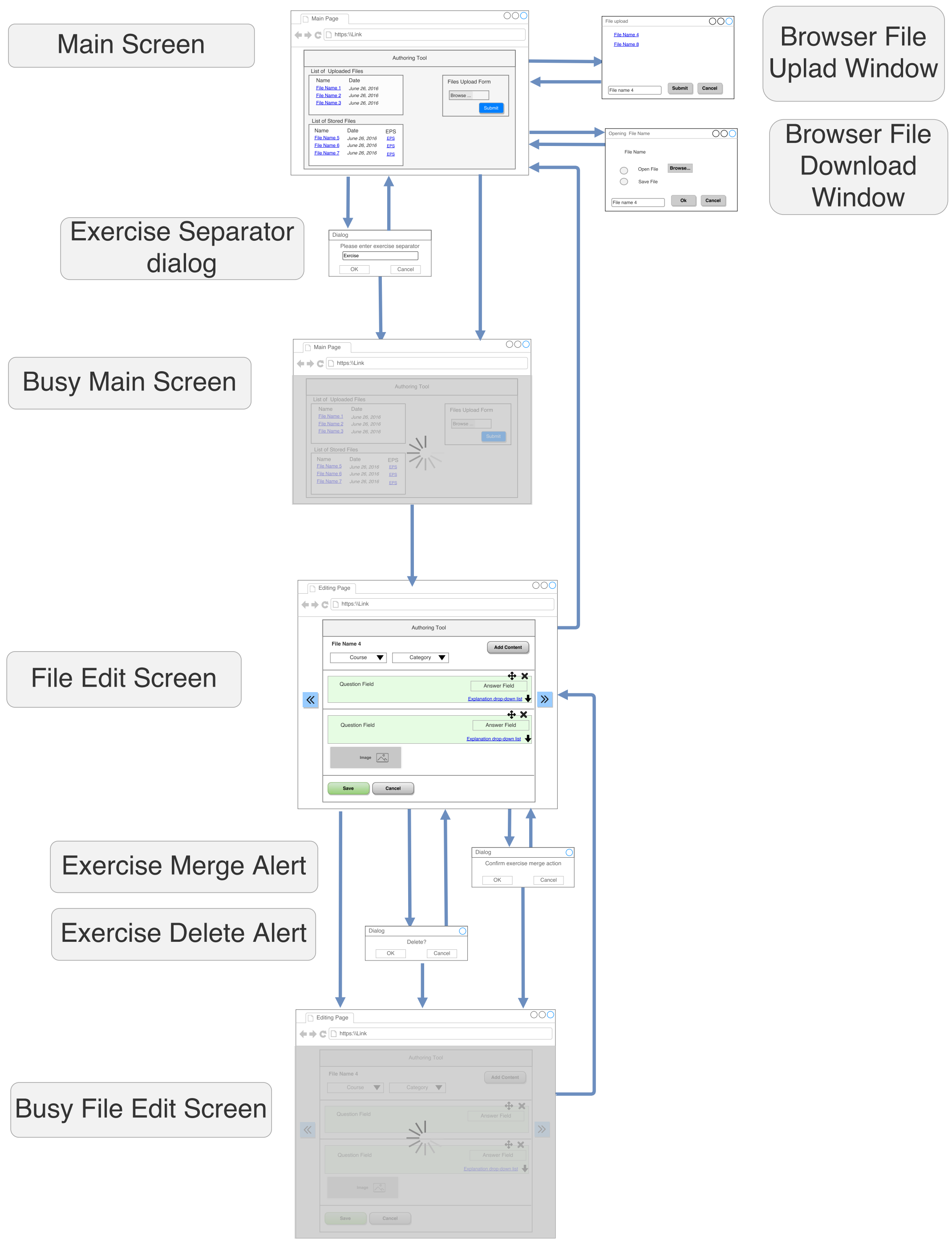


Figure 6.12: Screen and dialog sequence of the Authoring Tool

6.5.2 The Footer and Button "I am ready!" Template

When the saved PDF file is converted to the EPS file format, the application composes each page with the "I am ready!" button and footer with the logo of the university and reference to this thesis.

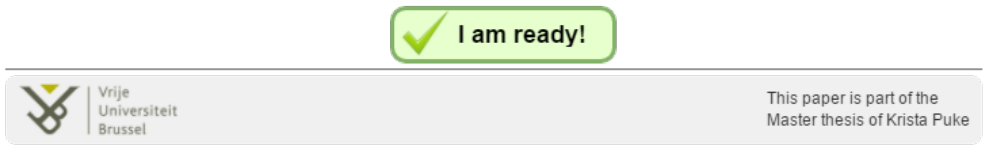

I am ready!
Vrije Universiteit Brussel
This paper is part of the Master thesis of Krista Puke

Figure 6.13: Footer for the converted PDF documents

6.6 Database Design

6.6.1 Object Role Model

Before designing the architecture of the database we started with the data modeling for which we used the Object Role Model (ORM). Figure 6.14 shows the ORM model for the database of te authoring tool. By using this modeling technique we ensure that all entities and attributes as well as their relations are known already before the database creation. There are 6 object types: File, Page, Course, Category, Exercise and Image. In our example there are no many-to-many relations between the objects, which is why the amount of tables will be equal to the amount of object types.

Next we build the database by applying a simple algorithm on our ORM model. This algorithm ensures that the database model will not contain redundancies. (Halpin, 2001)

6.6.2 Relational Database Diagram

The authoring tool uses the database, located on the server, to store and retrieve the information. The database is connected to the application over an Internet connection and users can reach the authoring tool through the web browser. Consequently, the application is accessible wherever users have an active Internet connection.

Mostly all the information that the application presents is gathered from the database. First it stores the information related to the uploaded files. Next during the PDF parsing process all the images from the file are stored on the server. When a user saves the parsed and modified document in the

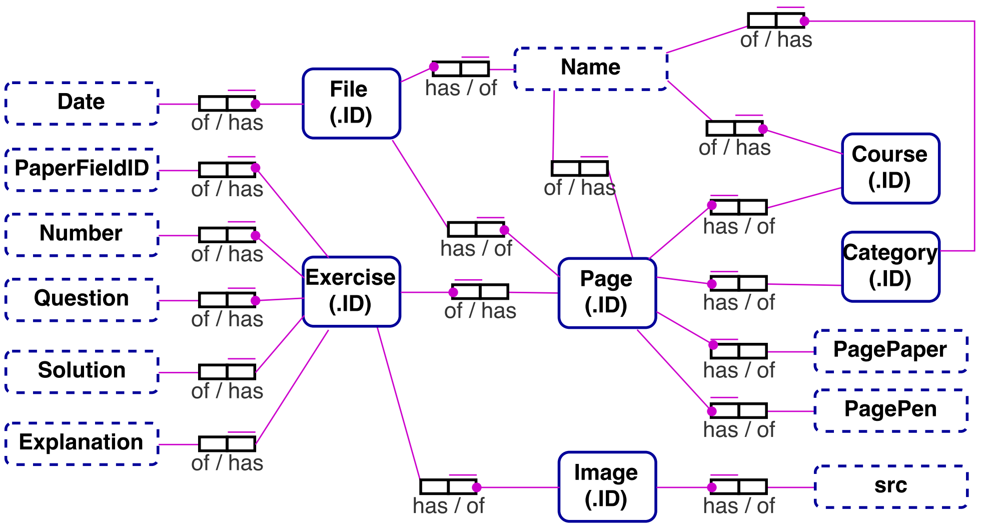


Figure 6.14: ORM model

application's editing mode, all the content of the pages (exercises, images, etc.) is stored in the database.

The database model of the authoring tool consists of 6 tables *(entities)*. An overview of the tables is presented in figure 6.15. As it is illustrated in the figure each of the tables has its own unique identifier (primary key); foreign keys are also indicated. All tables are presented in detail in the following sections.

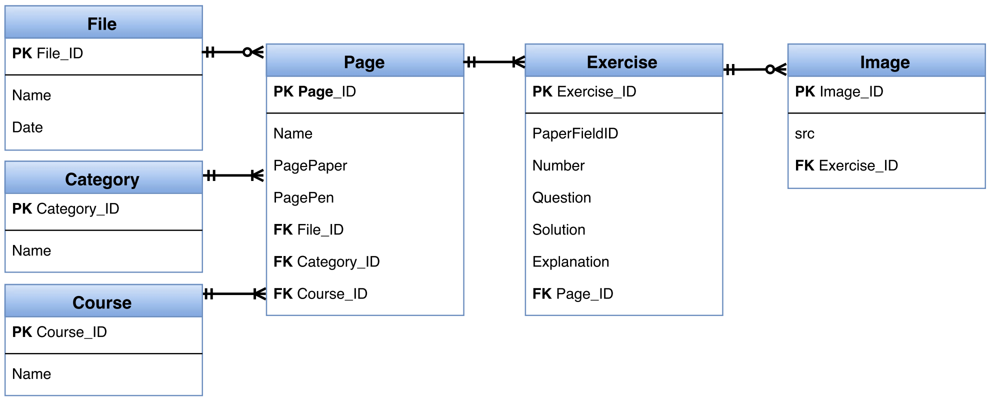


Figure 6.15: Database tables

File

This table stores the information about the all uploaded PDF documents. It contains the following attributes

- the *File_ID* is a unique identifier of the entity and has an auto increment functionality
- the *Name* is the name of the file
- the *Date* is the file uploading date

As the various functionality (example PDF conversion to EPS) in the application is using the file name it should be unique.

Course

The course table stores the information about the courses that are supported by the authoring tool. It has only two attributes, the unique identifier *Course_ID* and the name of the course *Name*.

Category

Similar to the *Course* table, the *Category* table also contains two attributes, the identifier and the name. Both tables are used for the drop-down lists in the application for classifying the page.

Page

This table stores all information about one page from a specific PDF file. As each page belongs to one file and each file can contain multiple pages, this table stores the identifier of the *File* table as a foreign key. Since the *Category* and *Course* tables are used to classify the page, they are used as foreign keys in the *Page* table, in addition to the attributes that describe the page. The attribute *PagePaper* describes the ordering number, while the *PagePen* attribute is used by the SmartPen to identify each printed page.

Exercise

Each page can have multiple exercises, that is why besides the other exercise attributes we also store the key to the *Page*. Other than this foreign key, the *Exercise* table contains the question, solution and explanation attributes as well the ordering number of the exercise in the page and *PaperFieldID* that is used for the SmartPen to distinguish each exercise.

Image

The image table stores the unique key and source of the images that are identified during the PDF file parsing. As each exercise can have zero or multiple images the key of the *Exercise* table is stored as foreign key in this database table. Based on our ORM model we can see that the image cannot exist without a relation to an exercise.

6.7 Summary

In this chapter we have shown the characteristics of the authoring tool design. We described user classes, and functional requirements. We also present the user interface with its accompanying design of all the views. Further we described the architecture of the authoring tool and discussed the relations between the tool components. Finally, we concluded the chapter with the object role model and the database model.

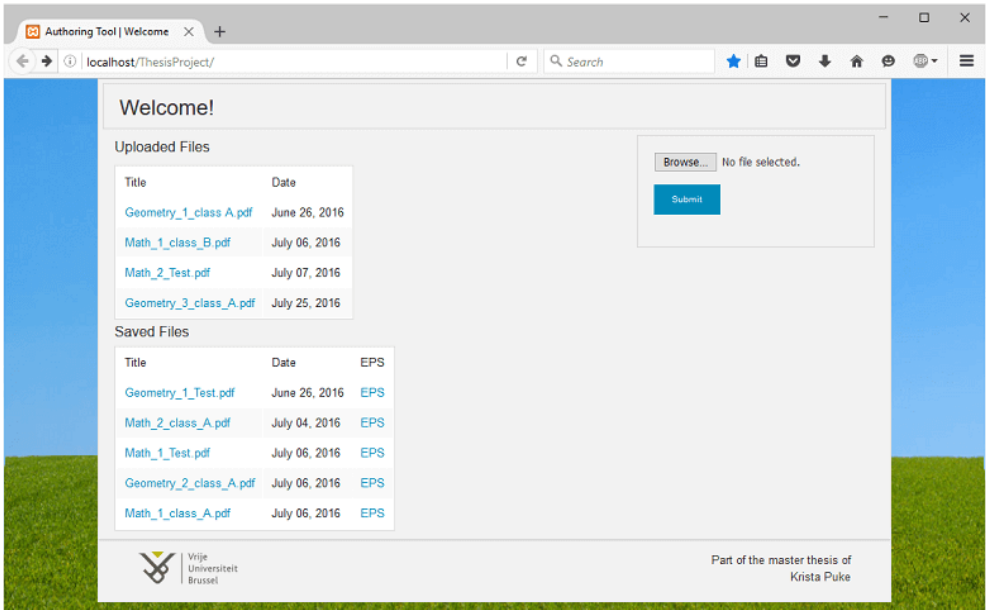
In the next chapter we present the Implementation, where we will present the implementation process of our authoring tool.

7 Implementation

In this chapter we explain how the relations between the authoring tool components are implemented. We also describe all the main methods, models and classes.

7.1 The Authoring Tool Application

The authoring tool is a PHP application that is developed using NetBeans IDE (v8.0.2) with HTML5 & PHP bundle. This bundle includes all the necessary tools to develop HTML5 and PHP applications (HTML, CSS, JS, integration with Apache, MySQL and other databases). The JAVA version 7 (or higher) is required for the IDE to work. (Oracle Corporation, 2016) The authoring tool is designed as a web application that ensures it is accessible on various devices that have an active Internet connection and web browser installed. The tool consists of two screens: the main screen (see fig: 7.1) and the file editing screen (see fig: 7.2, 7.3). Both screens are using various models and are supported by multiple controllers.



Welcome!

Uploaded Files

Title	Date
Geometry_1_class A.pdf	June 26, 2016
Math_1_class_B.pdf	July 06, 2016
Math_2_Test.pdf	July 07, 2016
Geometry_3_class_A.pdf	July 25, 2016

No file selected.

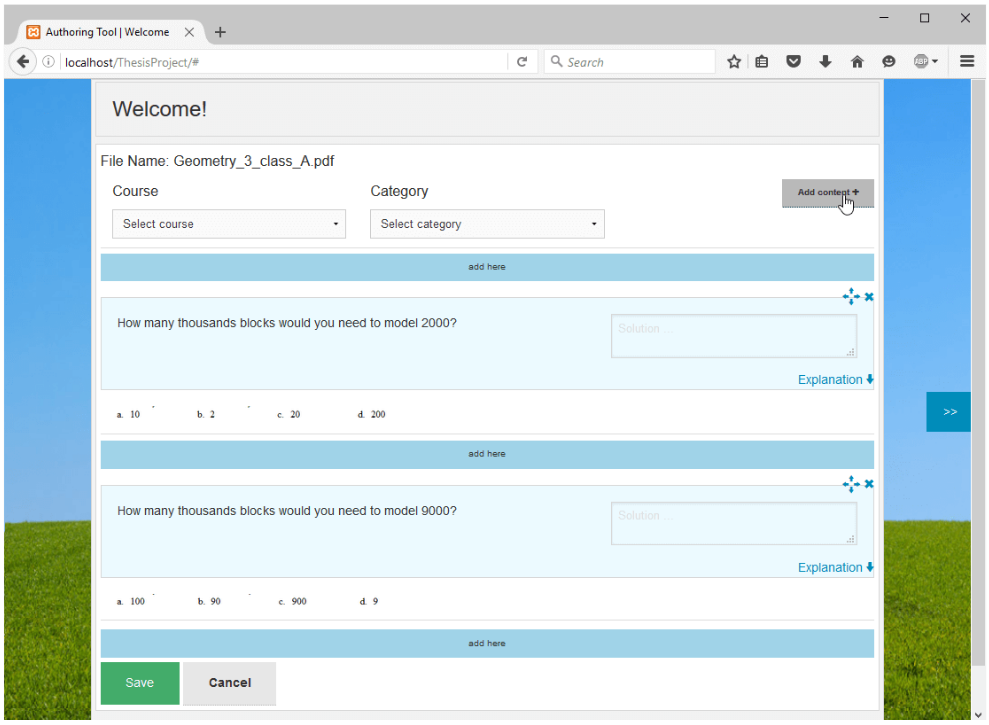
Saved Files

Title	Date	EPS
Geometry_1_Test.pdf	June 26, 2016	EPS
Math_2_class_A.pdf	July 04, 2016	EPS
Math_1_Test.pdf	July 06, 2016	EPS
Geometry_2_class_A.pdf	July 06, 2016	EPS
Math_1_class_A.pdf	July 06, 2016	EPS

Vrije Universiteit Brussel

Part of the master thesis of Krista Puke

Figure 7.1: Main screen of the authoring tool



Welcome!

File Name: Geometry_3_class_A.pdf

Course: Select course

Category: Select category

How many thousands blocks would you need to model 2000?

Solution ...

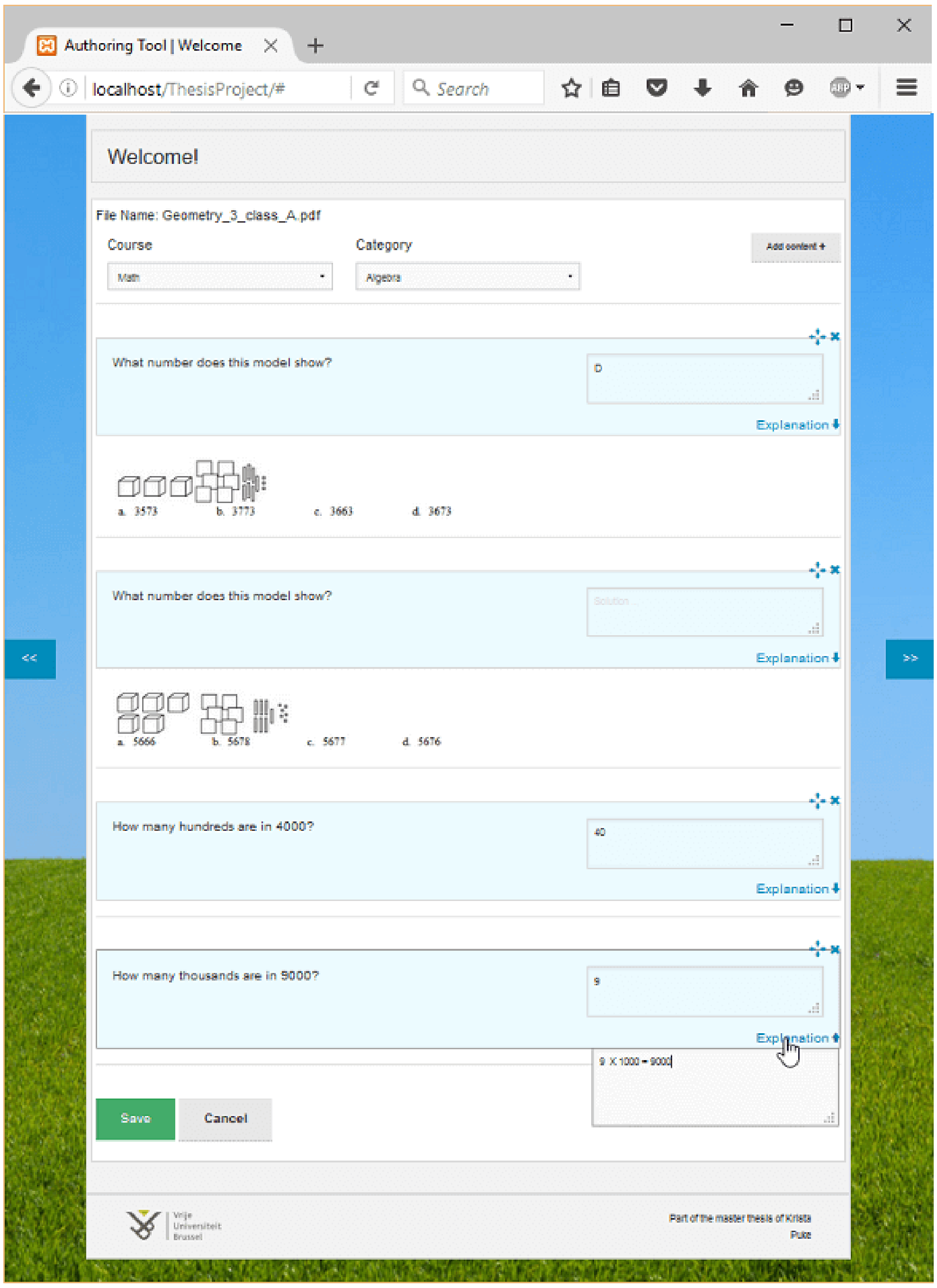
a. 10 b. 2 c. 20 d. 200

How many thousands blocks would you need to model 9000?

Solution ...

a. 100 b. 90 c. 900 d. 9

Figure 7.2: File edit screen of the authoring tool - A



Welcome!

File Name: Geometry_3_class_A.pdf

Course: Math

Category: Algebra

Add content +

What number does this model show?

D

Explanation

a. 3573 b. 3773 c. 3663 d. 3673

What number does this model show?

Solution ...

Explanation

a. 5666 b. 5678 c. 5677 d. 5676

How many hundreds are in 4000?

40

Explanation

How many thousands are in 9000?

9

Explanation

9 X 1000 = 9000

Save Cancel

Vrije Universiteit Brussel

Part of the master thesis of Krista Puke

Figure 7.3: File edit screen of the authoring tool - B

7.1.1 Files and Methods Used to Run the Application

Models:

File
1. Title: the name of the file
2. Date: the date when the file was uploaded
3. ID: the unique identifier from the database

Page
1. Page_ID: the unique page identifier
2. Page_Name: the name of the page
3. Page_Number: the logic number of the page in the file
4. Course: this is the foreign key to the *Course* entity
5. Category: this is the foreign key to the *Category* entity
6. ExercisesListObj: this is the list of *Exercise* objects that contains all contains all exercises from the current page

Exercise
1. Page_ID: the unique identifier of the parent object
2. Page_Name: the name of the parent object
3. Exe_ID: the unique exercise identifier
4. Question: the exercise question
5. Solution: the solution of the exercise
6. Explanation: the explanation
7. Images: this is list that contains images related to the current exercise
8. Flags: by default all the flags are set to 'no'
   - Changed: set to 'yes' if exercise content is changed
   - IsRemoved: set to 'yes' if current exercise is removed, this flag is used for the stored files, to know which exercises to delete from the DB
   - IsNew: set to 'yes' if exercise added, this flag is used for the stored files to distinguish to which exercises apply *update* and to which *insert*

The main screen is presented when the application is loaded. The data for the uploaded and stored files list are fetched from the database (tables *File* and *Page*). The file upload form is a HTML form, where the actions of the file upload is supported by the *fileUploadController.php* (see subsection: 7.2.2). Listing 7.1 presents the HTML file upload form.

Listing 7.1: HTML file upload form

```
<form action="controller/uploadFileController.php"
 method="post" enctype="multipart/form-data" >
 <input type="file" name="file" id="file" />
 <input type="submit" name="submit" value="Submit"
 class="tiny␣button"/>
</form>
```

After the PDF file is uploaded successfully the main screen is reloaded and the new document is listed in the uploaded files list.

Open and Edit Uploaded File

When the user clicks on the uploaded file name the system presents him the dialog window where the user can enter the exercise separator. Next the JS function *openUploadedPDF* is called and variables such as: *exerSeperator*, *fileName* and *fileId* are provided. Next we use the Asynchronous Javascript and XML (AJAX) to retrieve the data from the PHP file *getPDFdataFromPY*. This file sets the session variables (file name, exercise separator and file id) and gets the parsed PDF file content (see subsection: 7.1.2). Next, if the file parsing process was successful, the received file content is transformed in to the *Page* and *Exercise* object. These content transformations are done in the *setFileObectUploadedPDF* PHP file.

If the exercise separator is provided then the PHP file *seperateExercisesInPdfObject* is called. This file by using the *regex* and the exercise separator loops through all created *Exercise* objects in one page and searches for the match. Next based on the results some *Exercise* objects can be merged, if they previously were detected as separate ones. The result of the *setFileObectUploadedPDF* and *seperateExercisesInPdfObject* is returned to the AJAX code. Next from the AJAX successful branch the *printUploadedDivContent* is called. This PHP file is responsible to present the page header, footer and all exercises on the file edit screen.

File Edit Screen:

When the user changes the exercise question, answer or explanation text, the related *Exercise* object is updated. If the user removes an exercise, this *Exercise* object is deleted from the current *Page* object using PHP function *unset*. If a new exercise is added, the *Exercise* object of it is respectively added to the *Page* object, by using PHP function *array_splice*. If the user choses to merge exercises, by using drag and drop functionality, the dragged exercise object is removed from the hierarchy and the target exercise content(question and images) is updated.

Every time the object is removed from the hierarchy the order of the objects changes. To keep the sequence between all the keys we use PHP function *array_values* to re-index.

When the save action occurs the PHP file *exercisesInsertDB* is called (see subsection: 7.2.2).

Open and Edit Stored File

When the user chooses to open the file from the stored files list, the JS function *openSavedPDF* with the parameters: *fileName* and *fileId* is called. Next, we use AJAX to get the saved file content. The *getPDFdataFromDB* PHP file is responsible to query all necessary data from the database and generate the *Page* and *Exercise* objects (see subsection: 7.2.2).

Next from the AJAX success branch we call the *printStoredDivController*. This file is responsible to present on the file edit screen all information about the chosen PDF: file header(name, course, category), footer (save and cancel buttons) and the page content with related exercises.

File Edit Screen:

- If the user edits the text of the exercise question, answer or explanation the flag *Changed* is set to 'yes'. If the flag is set to 'yes' it means that this exercise records in the database will be updated.
- If the user choses to remove the exercise, the flag *IsRemoved* is set to 'yes'. This flag means that this exercise will be hidden from the screen and deleted from the database when the save action occurs.
- If the user chooses to add a new exercise, the new content appears at the place he chose. The flag *IsNew* on this exercise is set to 'yes' and when the save action will occur this exercise will be inserted in the database.
- If the user chooses to merge two exercises by using drag and drop functionality, the dragged exercise is marked as removed and the target exercise is changed. In the merge action both exercise questions are merged and all the images related to the exercises are presented below the target one.

For the database delete, insert and update functions see subsection 7.2.2.

7.1.2 PDFMiner Implementation

To use the PDFMiner, we downloaded and placed the library under the authoring tool project in the NetBeans platform. As a core structure to parse the PDF document we use Papathanasiou (2010) code. By modifying the given code example we found the solution for the authoring tool. The Python code is called from the PHP file *getPDFdataFromPY*. We use PHP function *popen(command,mode)* to open the pipe to the Python code. As a command variable to *popen* function we provide the link to the Python file, PDF document and its name. We set the mode to "r"; it means we will read only. Next while the connection is opened we gather all the data that are sent from the Python, by using binary safe file read function *fread()*.

Each page of the PDF file is parsed separately.

Page parsing:

1. It iterates through all the objects in the page
   (a) If the object is an instance of the *LTFigure* we look inside to knew if it contains an *LTImage*
   (b) If the object is an instance of the *LTImage*:
      i. we get the coordinates of it, to know where in the page it is located
      ii. if the image extension is PNG we store it on the server
      iii. if the image is other file format, we rewrite it to the PNG
   (c) If the object is an instance of the *LTTextBox*, we encode its text to the Universal Transformation Format-8 (UTF8).
2. Each object is added to the page in the order they are discovered (starting from the top)

When the page is completely analyzed and images are saved, the content of it is send back to the PHP file and the analyzing process of the next page, is existing, can be started.

7.1.3 ImageMagick Implementation

In the authoring tool we use ImageMagick to convert the PDF document to the EPS images and to compose each of the image with the "I am ready!" button and the footer (see subsection: 6.5.2). The PDF file conversion is called when the user clicks on the *EPS* link next to record in the stored files list. The PHP file *pdfToEpsConverter* is responsible to convert, compose images, compress them and push the file to the browser for the user to download it.

First the PDF file is converted to multiple EPS files. Listing 7.2 presents the PDF file conversion. The variable *$pdf_file* is the path to the PDF file and *$save_to_eps* is the result image name. If the PDF contains more than one page, at the end of the converted image the index of the page is added.

Listing 7.2: Comand to convert PDF file to EPS

```
exec(convert -density 250 $pdf_file $save_to_eps );
```

Next to get all created images we open the directory where they are stored and search for them. After all the necessary images are discovered we join them with the "I am ready!" button and footer (see listing: 7.3).

Listing 7.3: Compose Images

```
    foreach ($files as $name) {
        exec(composite -gravity South footbtn.png
            -density 250 tmp/.$name. tmp/.$name
        );
    }
```

Next we compress all created images related to the PDF file. After the archive is successful created we stream it to the user. The listing 7.4 presents how we set the browser headers for archive file download. In the first line we set the output type, in our case it is a *zip*. With the next line we give the name to the downloaded file. The command *readfile($path)* reads the file and writes it to the output buffer.

Listing 7.4: File download

```
    header('Content-type: application/zip');
    header("Content-Disposition: attachment; filename=$fileName");
    //required by some browsers to open the downlad window
    header("Content-length: " . filesize($path));
    //required for the IE
    header("Pragma: no-cache");
    header("Expires: 0");
    //cleans the output bffer
    ob_clean();
    flush();
    readfile($path);
    exit;
```

7.2 Database Implementation

The database architecture and design is presented in section 6.6. As we described in the section 6.3 the database is stored on a server. To simulate this architectural model we created a local web server by using the *XAMPP* service package (version v3.2.1). The tool provides us all the necessary packages to run a server and database, it is easy to install, cross-platform and the name stands for Cross-Platform, Apache, MariaDB, PHP and Perl. (Apache Friends, 2016)

We used the XAMPP control panel tool to switch on the Apache server and MySQL, with this setup we were able to access the database from the PHP files and *phpMyAdmin console*.

All the database operations are executed from the PHP files. We use one global PHP class to create and open the connection to the database. These connection settings are used all over the application. All the queries that are executed using this connection are typical CRUD operations: Create,Read, Update and Delete. All the PHP files that are containing the operations to connect to the database are explained in the following sections.

7.2.1 Database Connection

In this file we set up the connection to the MySQL DB. The PHP class is extended with the PHP Data Objects (PDO) that is one of the three APIs to set the connection between the application and the DB. First we set that char variables are coded in UTF8. It is important to keep all the text variables in the DB consistent. Next there is a connection object created and verified. For other PHP file to open the connection they need to create the new instance of this class.

7.2.2 Files and Methods Used To Communicate With Database

uploadFileController

This PHP file is called when the file upload process is started from the authoring tool main page. In this file we verify whether the DB table *File* does not already contain a record with the same file name. If the uploaded file name is unique we insert it in the *File* entity

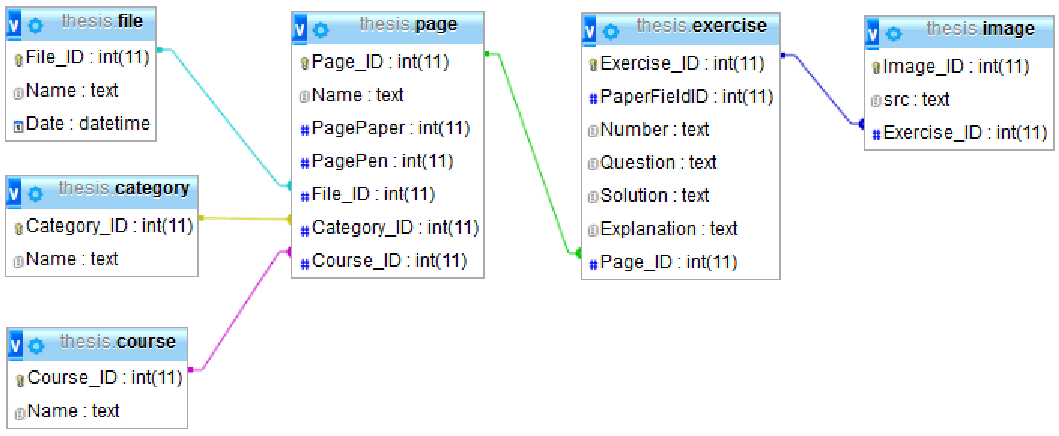


Figure 7.4: Database model

getFileList

This method is called when the application is preparing the data for the main screen. The query in this method selects all records from the *File* entity that are not connected with any record from the *Page* table. After the query is executed, each record from the fetched data is transformed to the *File* object and stored in the *$filesNew* list variable.

getFileListSaved

Similar as the uploaded files, the stored files are selected when the application is preparing the data for the main screen. This list of data is presented in the main screen stored files list block. The query selects all the records from the database table *File* that have the relation with the entity *Page*. Next each one of the fetched data rows is stored in the *File* object and added to the list *$filesSaved*.

getCategories

This method puts in the *$categoryList* all the saved categories from the DB table *Category*. Later this list is presented in the file edit view header the drop down list *Categories*.

getCourses

Similar as the list of categories, the courses list is presented in the file edit view header drop down list *Course*. The data for this list are fetched from the DB by selecting all the records from the table *Course*. Next each data row is transformed to the PHP object with the name and id and put in the list *$coursesList*.

exercisesInsertDB

This PHP file is called when the user chooses to store the uploaded file from the file edit screen. This file inserts all the information about the PDF document in to the database. As it is described in the section 7.1 the file array contains one or multiple *Page* objects and each *Page* contains one or more *Exercise* objects. Knowing this we follow the hierarchy and first object that is inserted in the DB is a page. After the page is inserted we use the PHP function *lastInsertId()* that as the result returns the last inserted identifier. Next the related exercises to the current page are inserted, as well as the foreign key to the *Page* entity is set. After each exercise insert we verify if the current exercise contains images, if yes then all images are inserted in the DB table *Image* and the foreign key to the exercise set.

All the identifiers of the DB tables has the auto increment functionality. We use getters from the *Page* and *Exercise* objects to fill the insert statements.

getPDFdataFromDB

This PHP file is called when the user choses to open the stored PDF to the file edit screen. Based on the given identifier all the information about the chosen file is selected from the DB. To fetch the data from the DB we use one query. The query selects all the exercises where the foreign key to the *File* table is equal with the given identifier, next it *LEFT JOIN* with the *Image*, *Page* and *File* tables. After the result is fetched from the database, the system generates the *File* and *Exercise* objects.

exercisesUpdateDB

This PHP file is called when the user chooses to save the stored file from the file edit screen. In this file we use three SQL functions: *update*, *insert* and *delete*. First each of the pages is updated with the current version of the changes. Next each exercise is updated and here the flags that are stored on the *Exercise* object helps to make a decision which SQL command to use.

First we delete all the removed exercises (flag *IsRemoved* is 'yes'), by first deleting everything from *Image* table where the foreign key to the *Exercise* entity is equal to the removed object identifier and next the exercise itself.

Next if the *Exercise* object flag *IsNew* is set to 'yes', we insert the new exercise in the database. Note, if there is an exercise removed or added, the logical numbering between all exercises in the one page is changed (the flag *Changes* is set to 'yes').

If the flag on the *Exercise* object is set to 'yes', the exercises question, answer, explanation, PaperFieldID and number is updated to the current version.

7.3 Summary

In this chapter we described the implementation of the authoring tool and the screen images. We also took a closer look at the methods behind the authoring tool screens and how the PDFMiner and ImageMagick platforms are implemented. We discuss the models and their attributes and flags.

Next, we have seen how the database is implemented and how the communication is built. We described the files and methods used to communicate with the database.

In the next chapter we present the Evaluation and Results where we will present and discuss the results of the authoring tool evaluation.

8 Evaluation and Results

In this chapter we present the evaluation of the authoring tool and the results. To evaluate the application we used different techniques. In the following sections we explain the setup of the evaluation, the methods used and the obtained results.

All the evaluation forms can be found in appendix A.

8.1 Setup

The evaluation was performed with three participants, all female, Latvian, primary school teachers, with ages ranging between 25 and 30. The participants were from two different schools and the classes they teach cover multiple topics. The communication was carried out through Skype[1] and in order to allow the participants to directly interact with our application we used TeamViewer[2].

[1] www.skype.com

[2] www.teamviewer.com

8.2 Methodology

The evaluation took place for all the participants at the same time. The opinion of each participant was gathered in three phases. Each phase used the methods that we found most suitable. The following section further explains all the used methods.

Phase 1

Teaching Material Creation and Usage Habits

We started with a focus group. We asked the participants open questions about their working material habits. During the discussion we focused on gathering answers to the following questions:

1. What kind of materials do you use in your classes?
    - printed (format of the stored materials: e.g. PDF, DOC);
    - copies from working books;
    - digital;
2. How often do you change your teaching materials?
3. How often do you need to edit them?
4. How often do you create new materials?
5. Can you re-use them (e.g., between different classes, or years)?
6. Is it possible to lose teaching materials? What you would do in this situation?

Computer Skills and Experience

Next we continue the open conversation by questioning the participants about their computer experience and available devices. We asked with what kind of computer devices they are familiar and if the school provides all the necessary equipment to create their teaching materials. We were also interested in how many hours per week participants spend working on the computer. The knowledge of the document (MS Word) creation and conversion to PDF file format was also inquired as well as the familiarity with the Internet and web browser.

Educational Software Usage and Experience

The final group of questions was about the experience using educational software to create teaching materials. Did the participants have experience with such software; usage habits of these tools; and the type of materials created. We were also interested in where a participant stores the materials (material creation tool, Internet or hard drive).

Phase 2

Each participant was given a prepared scenario to be performed, which covered all the provided features in the authoring tool. Before performing the task scenario, the participants were given a small presentation. During the presentation they were introduced to the main aspects of the user interface and functionality of the tool. The presentation was short to ensure that the results of the learnability and usability were accurate.

Next, each participant was invited to perform the scenario. While the participants were performing the scenario, we were taking notes about first impressions, feedback and comments.

Detailed Scenario

1. Open the application
2. Upload the prepared PDF file
3. Find the file in the uploaded files list and open it
4. Give the exercise separator
5. Move from page to page to verify that all the content is presented
6. Choose an exercise and remove it
7. Add a new exercise and choose the place for it
8. Merge two exercises
9. Set the course and category for the page
10. Save the changes
11. Open the saved file from the saved files list

12. Verify if all the changes you did are present (remove, add, merge exercise)
13. Remove another exercise
14. And add another exercise
15. Change some exercises content, by changing the question, adding the answer and solution
16. Save the changes
17. Download and store on your machine the compressed EPS images

Phase 3

To measure the usability of the authoring tool, we used the SUS survey (Brooke et al., 1996) with additional questions specific to the authoring tool.

SUS is a questionnaire that exists for more than 20 years. It is a popular and effective tool for the evaluation of the usability of various systems. The survey is using closed questions with a *Likert scale*, that provides a 5 (to 7) point scale to every question (statement): value 1 means *Strongly disagree* and value 5 *Strongly agree*, while the 3rd value is the neutral position. All the statements in the questionnaire are carefully selected and cover multiple aspects of the systems usability (e.g. learnability, complexity).

Besides the standard questions from the SUS survey we added some additional open questions that are listed below.

- What is your opinion on using educational software in the school environment?
  - To create teaching materials
  - For pupils to study or do tests/homework
- Could you suggest other possible uses for the tool?
- If the school would provide the authoring tool would you use it to digitize your teaching materials?
- What do you think of the content editing? (adding, deleting and merging exercises)

8.3 Results

Phase 1

Teaching Material Creation and Usage Habits

On the question about the type of materials participants are using, all the participants agreed that they use PDF and DOC (MS Word) documents, as well as copied materials from working books. Additionally they also exchange teaching materials with their colleges. The participants mentioned that they change their teaching materials at least once a week, but they do not edit them frequently.

All the participants indicated that they are teaching various subjects and they have more than one class to teach. They create new materials everyday as well as re-use previous materials completely or with some modifications.

The participants do not store the teaching materials in some particular place. If they lost them they would create new ones. From time to time they make a copy on the hard drive or email, but this happens if the materials must be printed and the printer is not accessible in the class room.

Computer Skills and Experience

All the participants have experience with computer devices, mainly personal computer, laptop and smart phone, but less with tablets. Every participant is familiar with a web browser.

Two of the participants received devices (personal computer and laptop) from their work place, while the third participant did not receive anything. Consequently, the third participant needed to use a personal device to create the teaching materials and was not happy about the lack of provided equipment.

As all the participants mainly use exercise books provided by the school, they spend 4 to 6 hours per week creating new exercises. Every participant knows how to save documents in a Microsoft Office[3] Word format, however one of the participants did not know how to save documents in the PDF format.

Educational Software Usage and Experience

The participants do not use any education software that would help them or guide them in the teaching materials creation process. All the participants

[3]www.office.com

mainly use Microsoft Office Word and Internet resources to create the teaching materials. None of the respondents backed up their teaching materials anywhere else other than the personal computer or laptop they used to create them.

Phase 2

We measured how much time it took for each participant to finish the given scenario, including the document conversion to the EPS file format. The average working time of all the participants was 8 minutes, while the maximum was 10 and minimum 6 minutes. We must note that the participants completed all the steps from the given scenario (see Detailed Scenario) during evaluation phase 2.

During this phase of the evaluation, participants were very concentrated to follow the given scenario. Therefore not a lot of remarks were given. The total amount of the remarks is 7 and they are listed below and grouped by participant.

First Participant remarks:

- Can I add an image to the new exercise (Functionality)
- Was hard to find EPS image download Link (User Interface)

Second Participant remarks:

- At the beginning of the evaluation: "I think it is impossible!" (Usability)
- Was hard to find how to add a new exercise (User Interface)
- If the program would be in Latvian it would be easier (Usability)

Third Participant remarks:

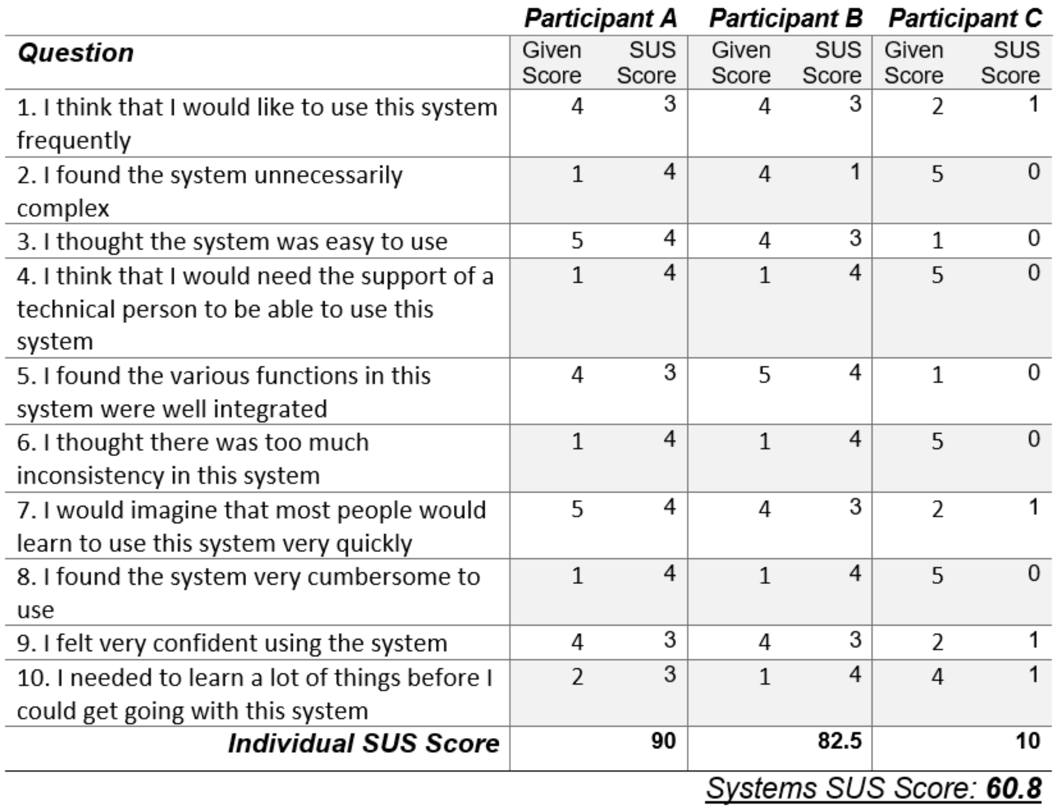
- It was easy (Usability)
- He saved himself! (Functionality)

Between all the given remarks we noticed three characteristic categories: Functionality, Usability and User Interface. We categorized each remark in its appropriate category. Four of them were split equally between user interface and functionality and three were related to usability.

Phase 3

Results from the SUS survey

All gathered SUS scores are presented in figure 8.1 as well as the calculated SUS score and the total score of the system. The survey contains positive and negative questions. To get the SUS score for each question per participant we subtract the given user score from 5, for the even questions, or subtract 1 from the user score, for the odd questions. To get the presentable results we sum up all the SUS scores per participant and multiply it by 2.5 and as a result we get a number between 0 and 100. (Bangor et al., 2009; Sauro, 2011)



	Participant A		Participant B		Participant C	
Question	Given Score	SUS Score	Given Score	SUS Score	Given Score	SUS Score
1. I think that I would like to use this system frequently	4	3	4	3	2	1
2. I found the system unnecessarily complex	1	4	4	1	5	0
3. I thought the system was easy to use	5	4	4	3	1	0
4. I think that I would need the support of a technical person to be able to use this system	1	4	1	4	5	0
5. I found the various functions in this system were well integrated	4	3	5	4	1	0
6. I thought there was too much inconsistency in this system	1	4	1	4	5	0
7. I would imagine that most people would learn to use this system very quickly	5	4	4	3	2	1
8. I found the system very cumbersome to use	1	4	1	4	5	0
9. I felt very confident using the system	4	3	4	3	2	1
10. I needed to learn a lot of things before I could get going with this system	2	3	1	4	4	1
Individual SUS Score		**90**		**82.5**		**10**

Systems SUS Score: ***60.8***

Figure 8.1: SUS survey results and scores

To get the average SUS result of the system we summed up all the individual SUS scores and divided this with the amount of participants. Figure 8.2 presents the SUS score per participant and the average system score. Two of the participants graded the system with over 80, while the third individual's SUS score was only 10. As a consequence the average score is 60.83.

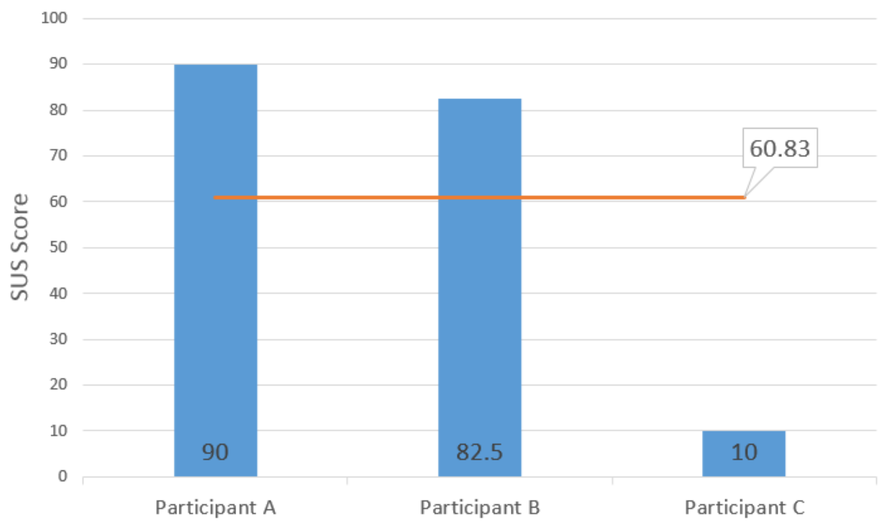


Figure 8.2: Bar chart presenting individual and the average SUS score of the authoring tool

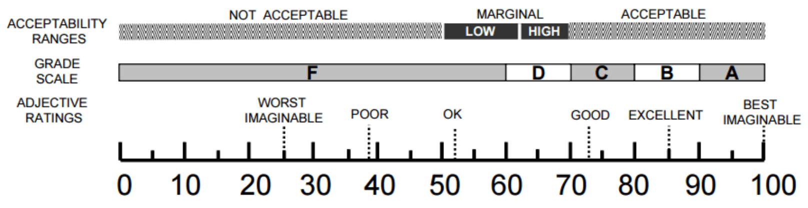


Figure 8.3: SUS score rank (Bangor et al., 2009)

Figure 8.3 presents the comparison between SUS score, acceptability range and grade scale, and based on the Bangor et al. (2009) research the average SUS score is 68. Bangor et al. (2009) also explains how to measure the learnability and usability of the evaluated system. Figure 8.4 presents the calculated scores of the authoring tool for the usability and learnability per participant. The highest score of 100 for learnability was given by participant B and the score of 90.625 for usability was given by participant A. The lowest scores for learnability and usability were given by participant C.

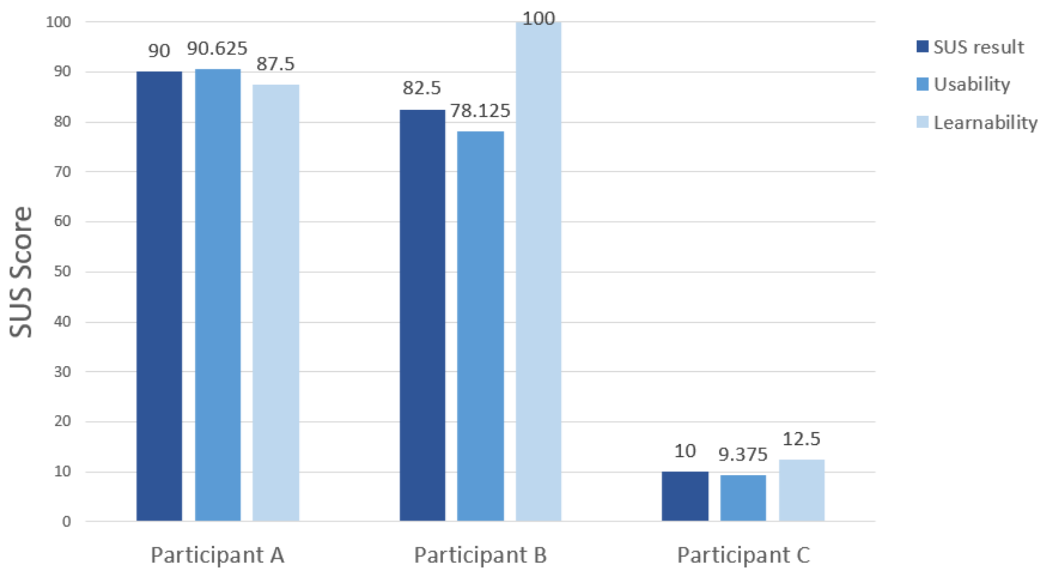


Figure 8.4: Bar chart illustrates the usability and learnability score based on the SUS survey

Results from the additional questions

What is your opinion on using educational software in the school environment?
The participants mentioned that all the responsibility to create interesting and attractive exercises interests the teachers. Consequently, all the participants felt it would benefit them to use additional education software that could help and guide them in creating more efficient and creative exercises and teaching materials.

Could you suggest other possible uses for the tool?
The first participant mentioned that the authoring tool could be useful not just for primary school teachers but also instructors that are preparing examinations. The second participant indicated that this tool could also be used like a cooperative exercise data storage, where colleagues could obtain information or already prepared exercises. The third participant mentioned that she could use such a tool to create tests for all classes rather than limiting it to math.

If the school would provide the authoring tool would you use it to digitize your teaching materials?
Answering to this question all the participants agreed that they would like

to use the program, because it is easy and useful, but they would require an additional software tool in order to create new documents from the modified exercises.

What do you think of the content editing?
All the participants stated that the provided functionality for managing the exercises is enough. One of the participants mentioned that the exercise merge is redundant, while others mentioned that for future use they might require additional functionalities.

8.4 Discussion

The results of the authoring tool evaluation were positive. Participants answered all the questions from the survey, gave concise answers to the open discussion questions and successfully followed the given scenario in the evaluation Phase 2. Participants mentioned that they are looking for educational software that would help them to collaborate, create and store teaching materials. As each of the participants has experience with computer devices and Internet, the approach to use web based education software was very attractive to them.

All the participants finished the given scenario in 10 minutes or less, that is a good result (the EPS conversion was included in the time spent). The scenario covered all the main functionality of the authoring tool and consisted of 17 steps. Based on this measure all the features were understandable and implemented in user friendly way.

The SUS score of the tool is 60.83 (see fig: 8.1 and 8.2), that is the average of the individual SUS scores. Bangor et al. (2009) mentions 68 as the required average SUS score for an acceptable level of usability. This means that we are under the average. However, we see big differences between the individual evaluations: two participants were very positive (high scores: 90 and 82.5 resulting in excellent and good usability), while one was negative (score of 10, meaning not usable). Because we only had three participants, the individual differences are influencing the overall result too much and therefore we cannot draw any definitive conclusions.

Comparing the results between the SUS and additional questions from the Phase 3 we can clearly see some inconsistency. On one hand participants, in the additional questions part, described the authoring tool as easy to use and the implemented functionality as sufficient. They also mentioned that for the future they might require additional functionality.

8.5 Summary

Throughout this chapter we have presented the results of the authoring tool evaluation. We characterized the participants and the setup of the evaluation. Afterwards, we described the phases of the evaluation and the outcome of each phase. Finally, we presented the results and discussed them.

In the next chapter we present the Conclusion, where we discuss the summary of the research and implementation and revisit the research question, presented in the section 1.1. We also discuss the limitations faced and potential future work.

9 Conclusion

In this chapter we provide a conclusion on the research conducted for the authoring tool. Additionally we discus limitations and possible future work.

9.1 Summary

Solid child education is key to their future, therefore we need to provide interesting, attractive and immersive environments that they can use to learn and increase their knowledge. Brooks (2014) mentions that children nowadays gain information by simultaneously using studying books and other smart technologies, such as television and other wide range computer devices. It would be favorable for educational institutions to change their materials to digital ones. Digital materials provide children with the most current version while the printed materials can contain already outdated information by the time they are being used.

Educational platforms are sometimes complex, hard to manage and some of them not affordable for all educational institutions or independent instructors. They provide various functionality that is partly used and often not needed for all institutions. As we saw in the evaluation we conducted, some teachers are creating their materials mostly by using Microsoft Office[1] Word

[1] www.office.com

or PowerPoint. To share their materials teachers often use emails or printed/copied versions edited by hand. Also they do not store their teaching materials on external storages, but rather on their computer devices, which may lead to loss of documents. With the developed authoring tool we tried to cover multiple paths of educational material creation. The tool allows to digitalize teaching material and content management as well as integration with the Smart Study educational platform. Though the pen application creation process is not automated, due to technical limitations described below, the tool provides the user with the possibility to convert the uploaded documents to the EPS format, which is required for the Livescribe Platform to generate the pen application.

9.2 Limitations and Future Work

During the research and development process we faced a couple of limitations that are discussed in this section. Besides that we also describe possible future work.

Heavy and complicated EPS

EPS image file format itself is heavy and requires specific tools to open it. The user should have knowledge of these tools (e.g., Adobe Illustrator, Inkscape) and vector graphic editing to be able to edit the EPS format images. In the analysis phase we did not find any open source tool that would allow conversion to the EPS format.

The creation of the Paper Project and Penlet

All the development tools provided from Livescribe have been discontinued since 2011, when Livescribe announced the end of the development program[2].

We also faced the issue that the Livescribe Paper Project AFD file can only be created and opened by using the eclipse IDE together with the Livescribe SDK.

Authentication

As the authoring tool is a web application and is available on various devices that have an active internet connection, an authentication process is required to keep the created author documents private. Authentication would allow the user to store, manage and overview documents that belong to the author.

[2]https://www.livescribe.com/errors/developer.html

DOC file format parsing and PDF templates

Currently the authoring tool application provides PDF document parsing. To make the options wider the authoring tool should also support DOC, ODT and other document format parsing. Besides that, a great feature would be to create completely new documents from provided templates, newly created or modified documents and provide the user the ability to export them as PDF.

Currently the authoring tool application provides PDF document parsing. It would be interesting to also support other document formats such as DOC, ODT. Besides that, a great feature would be to create completely new documents from provided templates, newly created or modified documents and provide the user the ability to export them as PDF.

Data Hub and Learning Analytics

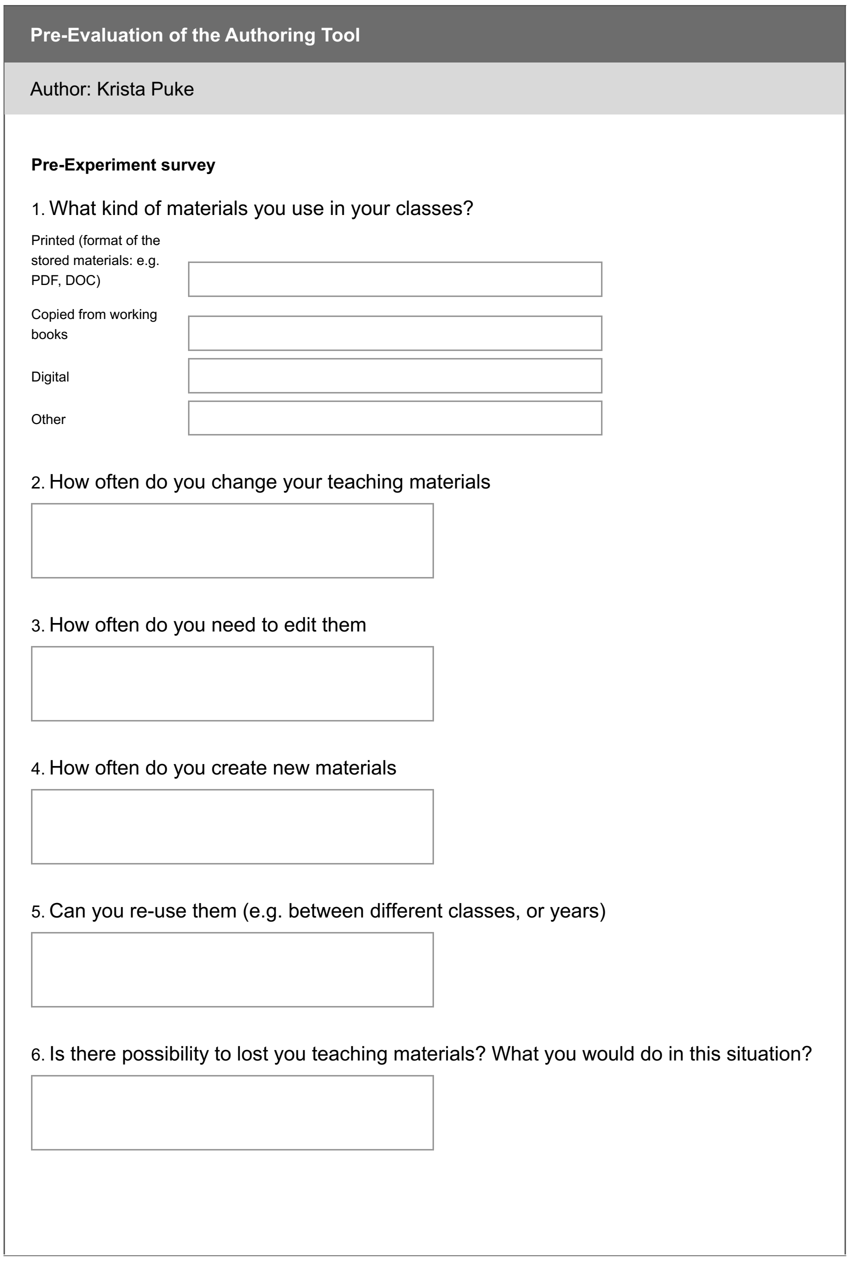
Our authoring tool and its database can be taken a step further and be incorporated as an instance of a data hub. Each unique component of the system can then share content and retrieve publicly stored learning objects from the hub. A great addition to this structure would be a learning analytics module for a better understanding and further optimization of the educational environment.

Part IV

Appendix

A
Evaluation Forms

A.1 Pre-Experiment Survey



Pre-Evaluation of the Authoring Tool

Author: Krista Puke

Pre-Experiment survey

1. What kind of materials you use in your classes?

Printed (format of the stored materials: e.g. PDF, DOC)	
Copied from working books	
Digital	
Other	

2. How often do you change your teaching materials

3. How often do you need to edit them

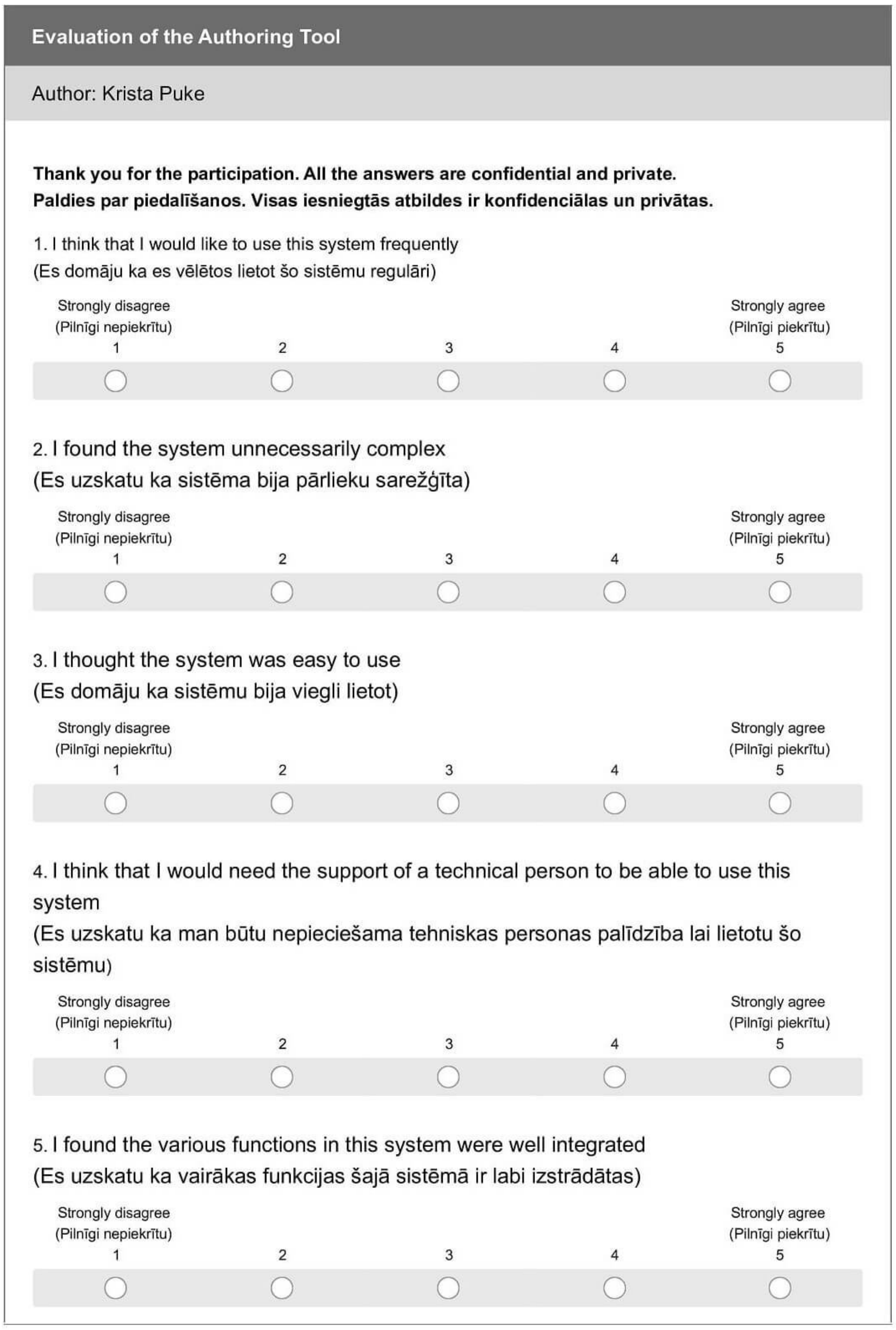
4. How often do you create new materials

5. Can you re-use them (e.g. between different classes, or years)

6. Is there possibility to lost you teaching materials? What you would do in this situation?

Figure A.1: Pre-Experiment Survey

A.2 Post Experiment Survey



Evaluation of the Authoring Tool

Author: Krista Puke

Thank you for the participation. All the answers are confidential and private.
Paldies par piedalīšanos. Visas iesniegtās atbildes ir konfidenciālas un privātas.

1. I think that I would like to use this system frequently
(Es domāju ka es vēlētos lietot šo sistēmu regulāri)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

2. I found the system unnecessarily complex
(Es uzskatu ka sistēma bija pārlieku sarežģīta)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

3. I thought the system was easy to use
(Es domāju ka sistēmu bija viegli lietot)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

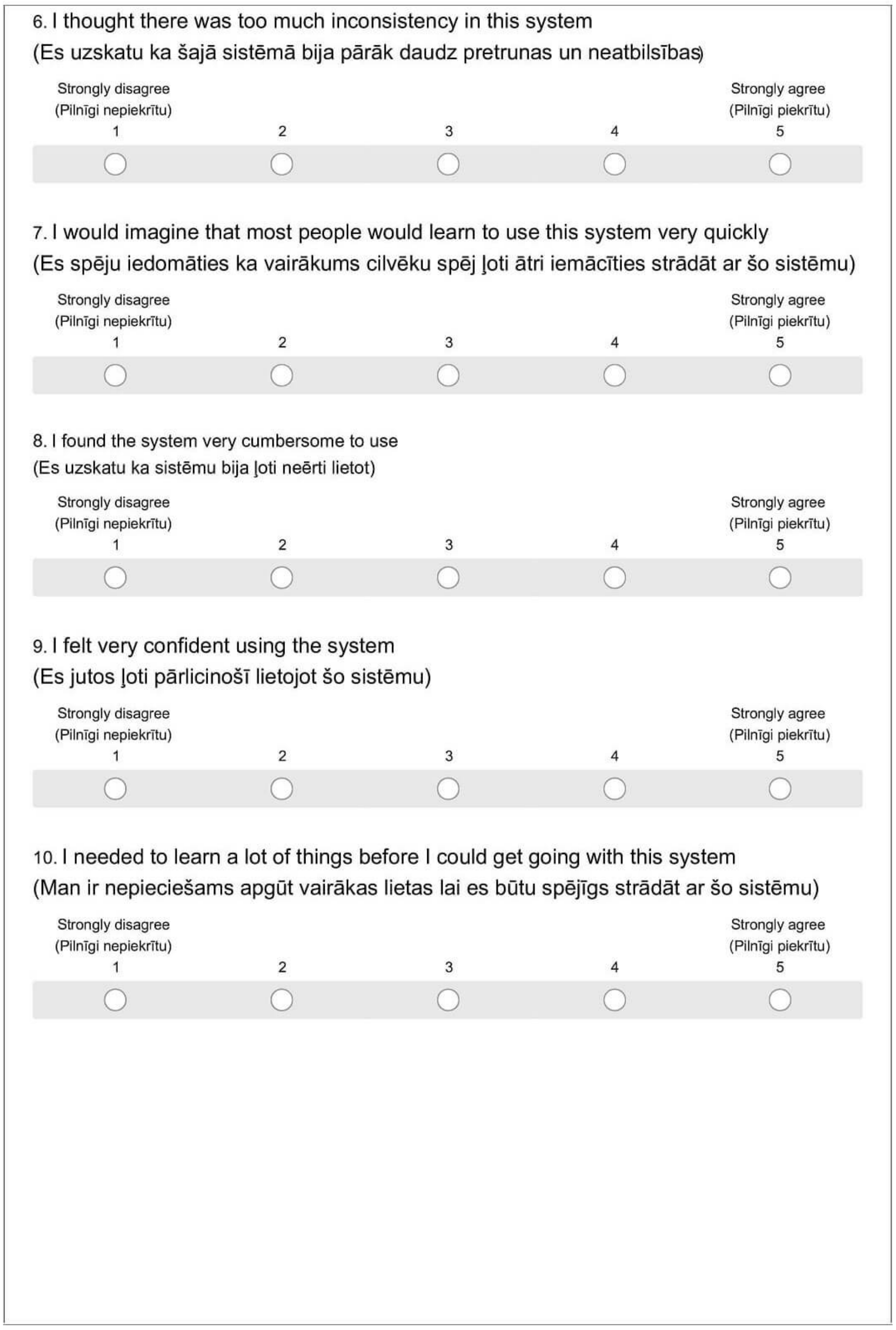
4. I think that I would need the support of a technical person to be able to use this system
(Es uzskatu ka man būtu nepieciešama tehniskas personas palīdzība lai lietotu šo sistēmu)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

5. I found the various functions in this system were well integrated
(Es uzskatu ka vairākas funkcijas šajā sistēmā ir labi izstrādātas)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

Figure A.2: Post Experiment Survey - Page 1



6. I thought there was too much inconsistency in this system
(Es uzskatu ka šajā sistēmā bija pārāk daudz pretrunas un neatbilsības)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

7. I would imagine that most people would learn to use this system very quickly
(Es spēju iedomāties ka vairākums cilvēku spēj ļoti ātri iemācīties strādāt ar šo sistēmu)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

8. I found the system very cumbersome to use
(Es uzskatu ka sistēmu bija ļoti neērti lietot)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

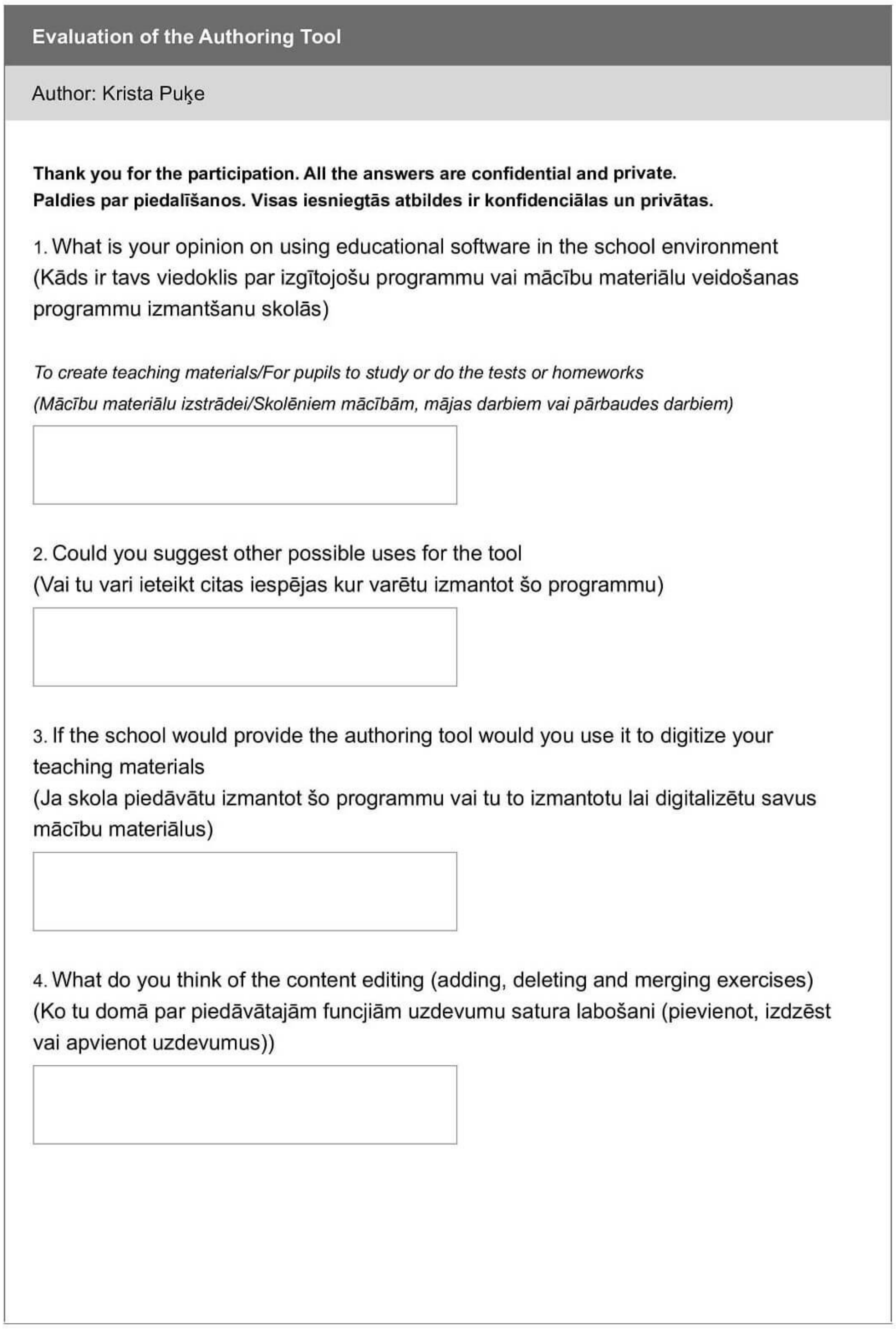
9. I felt very confident using the system
(Es jutos ļoti pārlicinošī lietojot šo sistēmu)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

10. I needed to learn a lot of things before I could get going with this system
(Man ir nepieciešams apgūt vairākas lietas lai es būtu spējīgs strādāt ar šo sistēmu)

Strongly disagree (Pilnīgi nepiekrītu) 1	2	3	4	Strongly agree (Pilnīgi piekrītu) 5
○	○	○	○	○

Figure A.3: Post Experiment Survey - Page 2



Evaluation of the Authoring Tool

Author: Krista Puķe

Thank you for the participation. All the answers are confidential and private.
Paldies par piedalīšanos. Visas iesniegtās atbildes ir konfidenciālas un privātas.

1. What is your opinion on using educational software in the school environment
(Kāds ir tavs viedoklis par izgītojošu programmu vai mācību materiālu veidošanas programmu izmantšanu skolās)

To create teaching materials/For pupils to study or do the tests or homeworks
(Mācību materiālu izstrādei/Skolēniem mācībām, mājas darbiem vai pārbaudes darbiem)

2. Could you suggest other possible uses for the tool
(Vai tu vari ieteikt citas iespējas kur varētu izmantot šo programmu)

3. If the school would provide the authoring tool would you use it to digitize your teaching materials
(Ja skola piedāvātu izmantot šo programmu vai tu to izmantotu lai digitalizētu savus mācību materiālus)

4. What do you think of the content editing (adding, deleting and merging exercises)
(Ko tu domā par piedāvātajām funcjiām uzdevumu satura labošani (pievienot, izdzēst vai apvienot uzdevumus))

Figure A.4: Post Experiment Survey - Page 3

Bibliography

Al-Ajlan, A. S. (2012). *A comparative study between e-learning features* (Dr. Elvis Pontes, Ed.). INTECH Open Access Publisher.

Apache Friends. (2016). *Xampp installers and downloads for apache friends.* `https://www.apachefriends.org/index.html`. (Accessed on 07/04/2016)

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114–123.

Blackhall, L. (2011). Educational content authoring tools. *A report for written for the college of engineering and computer science. The Australian National University.*

Bradford, P., Porciello, M., Balkon, N., & Backus, D. (2007). The blackboard learning system: The be all and end all in educational instruction? *Journal of Educational Technology Systems*, *35*(3), 301–314.

Brooke, J., et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Brooks, S. (2014). *Five reasons why education gets digitized - edtechreview™ (etr).* `http://edtechreview.in/e-learning/1144-five-reasons-why-education-gets-digitized`. (Accessed on 07/26/2016)

Dabbagh, N. (2002). The evolution of authoring tools and hypermedia learning systems: Current and future implications. *Educational Technology*, *42*(4), 24–31.

Dictionary.com. (2016). *Authoring | define authoring at dictionary.com.* `http://www.dictionary.com/browse/authoring`. (Accessed on 07/15/2016)

Dougiamas, M. (2010). *Moodle 2 overview.* http://www.slideshare.net/moodler/moodle-2-overview. (Accessed on 07/13/2016)

Farlex Inc. (n.d.). *Authoring - definition of authoring by the free dictionary.* http://www.thefreedictionary.com/authoring. (Accessed on 07/15/2016)

FinancesOnline. (2016). *Compare moodle vs blackboard 2016.* https://comparisons.financesonline.com/moodle-vs-blackboard-learn. (Accessed on 07/13/2016)

Gordillo, A., Barra, E., Gallego, D., & Quemada, J. (2013). An online e-learning authoring tool to create interactive multi-device learning objects using e-infrastructure resources. In *Frontiers in education conference, 2013 ieee* (pp. 1914–1920).

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

ImageMagick Studio LLC. (2016). *Imagemagick: Convert, edit, or compose bitmap images.* http://www.imagemagick.org/script/index.php. (Accessed on 06/24/2016)

Isotani, S., Mizoguchi, R., Isotani, S., Capeli, O. M., Isotani, N., & de Albuquerque, A. R. (2010). An authoring tool to support the design and use of theory-based collaborative learning activities. In *International conference on intelligent tutoring systems* (pp. 92–102).

Isotani, S., Mizoguchi, R., Isotani, S., Capeli, O. M., Isotani, N., De Albuquerque, A. R., ... Jaques, P. (2013). A semantic web-based authoring tool to facilitate the planning of collaborative learning scenarios compliant with learning theories. *Computers & Education*, *63*, 267–284.

iSpring Solutions, Inc. (2015). *Moodle vs blackboard – free vs paid.* http://www.ispringsolutions.com/blog/moodle-vs-blackboard/. ((Accessed on 07/30/2016))

Jisc Digital Media. (2016). *Introduction to elearning.* http://www.jiscdigitalmedia.ac.uk/guide/introduction-to-elearning. (Accessed on 07/12/2016)

Johari, M. (2013). *Digitization of education: Great change in teaching-learning trends.* http://scholarship-positions.com/blog/digitization-of-education-great-change-teaching-learning-trends/201307/. (Accessed on 07/23/2016)

Khademi, M., Haghshenas, M., & Kabir, H. (2011). A review on authoring tools. In *Proceedings of the 5th international conference on distance learning and education, ipcsit* (Vol. 12, pp. 40–44).

Kumar, S., Gankotiya, A. K., & Dutta, K. (2011). A comparative study of moodle with other e-learning systems. In *Electronics computer technology (icect), 2011 3rd international conference on* (Vol. 5, pp. 414–418).

Legault, N. (2014). *What is e-learning? - e-learning heroes.* `https://community.articulate.com/series/getting-started/articles/what-is-e-learning`. (Accessed on 07/12/2016)

Livescribe Inc. (2008). *Penlet (pen profile api reference).* `http://www.spiska.sk/manuals/penprofile-apidoc/index.html?com/livescribe/penlet/Penlet.html`. (Accessed on 07/05/2016)

Livescribe Inc. (2009). Developing Paper Products.

Markotic, A. (2015). *E-learning for primary school teachers - unite it: The e-inclusion network in europe.* `http://www.unite-it.eu/profiles/blogs/e-learning-for-primary-teachers-1`. (Accessed on 07/13/2016)

Mayo Clinic. (2015). *Aphasia - mayo clinic.* `http://www.mayoclinic.org/diseases-conditions/aphasia/basics/definition/con-20027061`. (Accessed on 07/11/2016)

Melia, M. (2012). *A solution to federated moodle management - enovation solutions.* `http://www.enovation.ie/a-solution-to-federated-moodle-management/`. (Accessed on 07/13/2016)

Mohan, P. (2004). Building an online course based on the e-learning standards: Guidelines, issues, and challenges. *SANDBOX-Canadian Journal of Learning and Technology/La revue canadienne de l'apprentissage et de la technologie*, *30*(3).

Moodle community. (2015). *Repositories - moodledocs.* `https://docs.moodle.org/31/en/Repositories`. (Accessed on 07/13/2016)

Olesen, J. (n.d.). *Color meanings - learn about colors and symbolism.* `http://www.color-meanings.com/`. (Accessed on 07/01/2016)

Oracle Corporation. (2016). *Netbeans ide - php development.* `https://netbeans.org/features/php/index.html`. (Accessed on 07/04/2016)

Papathanasiou, D. (2010). *Extracting text & images from pdf files.* `http://denis.papathanasiou.org/posts/2010.08.04.post.html`. (Accessed on 06/14/2016)

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.

Pillora, J. (2014). *Notify.js.* `https://notifyjs.com/`. (Accessed on 07/03/2016)

Piper, A. M., Weibel, N., & Hollan, J. D. (2011). Write-n-speak: Authoring multimodal digital-paper materials for speech-language therapy. *ACM Transactions on Accessible Computing (TACCESS)*, *4*(1), 2.

Piper, A. M., Weibel, N., & Hollan, J. D. (2012). Tap & play: an end-user toolkit for authoring interactive pen and paper language activities. In *Chi* (pp. 149–158).

Quark Software Inc. (2015). *The beginner's guide to content automation.* `http://www.quark.com/en/Solutions/Content-Automation/Content-Automation-Software.aspx`. (Accessed on 07/15/2016)

Rouse, M. (2007). *What is use case? - definition from whatis.com.* `http://searchsoftwarequality.techtarget.com/definition/use-case`. (Accessed on 06/21/2016)

Sauro, J. (2011). *Measuring usability with the system usability scale (sus): Measuringu.* `http://www.measuringu.com/sus.php`. ((Accessed on 07/19/2016))

Shinyama, Y. (2010). *Pdfminer.* `http://www.unixuser.org/~euske/python/pdfminer/`. (Accessed on 06/14/2016)

Shinyama, Y. (2014). *Programming with pdfminer.* `http://www.unixuser.org/~euske/python/pdfminer/programming.html#layout`. (Accessed on 06/23/2016)

University of Otago. (2014). *Blackboard helpsite for staff | information technology services.* `https://help.otago.ac.nz/blackboard/`. (Accessed on 07/14/2016)

Van Thienen, D., Sajjadi, P., & De Troyer, O. (2015). Smart study: Pen and paper-based e-learning. In V. L. Uskov, J. R. Howlett, & C. L. Jain (Eds.), *Smart education and smart e-learning* (pp. 93–103). Springer.

Virvou, M., & Moundridou, M. (2000). A web-based authoring tool for algebra-related intelligent tutoring systems. *Educational Technology & Society*, *3*(2), 61–70.

Vision One. (2015). *Interactive whiteboard 101 — a resource of activities for literacy instruction.* `http://www.visionone.com.au/interactive-whiteboard-101-a-resource-of-activities-for-literacy-instruction/`. (Accessed on 07/14/2016)

W. Caron, S. (2011). *Education world: Smart teaching with interactive whiteboards.* `http://www.educationworld.com/a_tech/smart_teaching_with_interactive_whiteboards.shtml`. (Accessed on 07/14/2016)

Zurb. (2015). *Getting started | foundation docs.* `http://foundation.zurb.com/sites/docs/v/5.5.3/`. (Accessed on 07/01/2016)