

UNIVERSITY OF OSLO
Department of Informatics

Standards - Access for
everyone. Converting
OOXML, ODF and
HTML.

Master thesis

Ingunn Elisabeth
Sundal Rønningen

3. August 2009



Acknowledgements

I would like to express my gratitude to those who have contributed to this thesis.

A special recognition goes to my advisors. Jo Herstad has been a great resource for guidance throughout the process. I have appreciated every session we have had, bringing me constructive criticism, encouragement and support. A big thank you to Håkon Wium Lie who presented me with this problem area, for inspiring me and for giving me helpful pointers and contributions along the way.

Thank you to the kind people at my student union PING, I appreciate the help and worthwhile discussions.

I would also like to show my sincerest gratitude to those who took time to give me feedback on the thesis; Trenton, Andy and Christian. Your contributions were greatly valued.

Lastly, I would like to give thanks to my fantastic family who always show their support, and to Christian, for being loving and understanding.

Abstract

It seems a feasible task for two people to share information in this modern age. But what happens when this information, in order to be conveyed, needs a distinct format, and this format needs a distinct application? In addition, that this application needs a distinct platform, and this platform needs a distinct set of hardware to convey this information appropriately. There is an obvious problem of having the right set of tools to view this piece of information. This paper will address this issue in the scope of the office document formats Open Document Format (ODF), Office Open XML (OOXML) and the publishing language for the web; HyperText Markup Language (HTML). The thesis will provide a possible solution to a part of the aforementioned problem, namely in the sharing of these type of documents.

A close look at the activity of sharing a document is conducted, along with its context. Looking at the formats specifications, it is clear that the structure of the content is the same. Regarding formatting, the open standards differ. With OOXML and ODF it is a mere difference in structure of the document. HTML, on the other hand need CSS in order to include equivalent presentation qualities as the other two formats.

A system was designed to perform conversions between the formats. The scope of this system was narrowed down to converting only text, without any presentational qualities. With this limited scope, it was natural to look away from the presentation and spreadsheets, and instead focusing on word processing. Hence, the attention was directed at DOCX from OOXML, ODT from ODF, and to HTML. This adjustment was done for the sake of locating a lowest common denominator to build further on, and to make sure the conversions between the formats were feasible. The resulting system had successful conversions between documents stored in the aforementioned formats containing text. There was a deficiency in the conversions from DOCX to ODT and HTML. However, the remaining set of conversions ran successfully; between HTML and ODT, from ODT to DOCX and from HTML to DOCX.

Even if the system was not carrying out complete conversions between all the formats, it shows great promise towards accomplishing this, hence affirming the possibility to convert between the three mentioned formats.

Contents

1	Introduction	1
1.1	Problem Definition	3
1.2	Problem Area	3
1.3	Definitions	5
1.4	Motivation	5
1.5	MOSCITO	6
1.6	Chapter Overview	7
2	Methods	9
2.1	Gaining the knowledge	9
2.1.1	Document analysis	9
2.1.2	Semi-structured interviews	9
2.1.3	Conferences	10
2.2	Development method	10
3	Theory	13
3.1	Communication	13
3.1.1	Possible problems	14
3.2	Human-Computer Interaction	15
3.3	Standards	17
3.3.1	What is a standard	17
3.3.2	Thought experiment	21
3.3.3	Evolution of standards	22
3.3.4	Who makes standards	23
	OASIS	24
	ISO	24
	Ecma	25
	W3C	25
	In common	26
3.3.5	Implications	26
3.4	Office documents	27
3.4.1	SGML	27
3.4.2	HTML	28
	Microformats	30
	MathML	32
3.4.3	ODF	32
3.4.4	OOXML	34
3.4.5	Commonalities	36
	Interaction	37
	Construction	37
	Overlaps	37
3.5	Affiliated terms	38
3.5.1	Open and free format	38

3.5.2	Open source	38
3.5.3	Free software	39
3.5.4	Open Source and business	40
4	Case	41
4.1	Initial planning	41
4.1.1	Research	41
4.1.2	The specifications	42
4.1.3	Structured documents	42
4.1.4	Existing converters	43
4.2	Developer Environment	44
4.2.1	Python	44
4.2.2	Apache2 HTTP server	45
4.2.3	PHP5 vs mod_python	45
4.2.4	MySQL	45
4.2.5	phpMyAdmin	45
4.2.6	Dropbox	45
4.3	Planning and setting requirements	46
4.3.1	Technical and functional requirements	47
4.3.2	Non-Functional requirements	48
4.4	Analyzing and designing	48
4.4.1	The website	48
4.4.2	The framework	49
4.5	Developing, implementing and testing	52
4.5.1	Website	52
4.5.2	Framework	53
	View	53
	Model	53
	Controller	54
	<i>The development environment</i>	54
	<i>The database</i>	54
	<i>The treatfile module</i>	55
	<i>The NewFile module</i>	55
	<i>The Mapping module</i>	55
4.6	Evaluating	59
5	Findings	61
5.1	Semi-structured interviews	61
5.1.1	Interview with Håkon Wium Lie	61
5.1.2	Interview with Ole Hanseth	63
5.2	Case findings	65
5.2.1	Experience	65
5.2.2	Potential	65
6	Discussion	67

6.1	Is it possible to convert between OOXML, ODF, and HTML? .	67
6.2	What is a standard?	68
6.3	Do open standards help to enable communication and universal design?	70
6.4	Is it possible to create a framework that eases conversions between overlapping formats ?	72
6.5	Can a framework that converts between overlapping formats be developed successfully using open source technology? . . .	76
7	Conclusion	77
7.1	Weak points of the thesis	77
7.2	Future work	78
	Appendix A: Consent forms	85
	Appendix B: Semi-structured interview questions	87
	Appendix C: Code	89

List of Figures

1	Intersection of HTML, ODF and OOXML	2
2	Iterative development.	10
3	Shannon's schematics.	13
4	Shannon's schematics applied in this context.	14
5	The task of communicating a document.	16
6	How many standards for communicating text.	21
7	Timeline of standards.	22
8	Different people see different parts of a situation.	23
9	Example matrix.	32
10	Matrix MathML markup.	32
11	Example markup of the content file of ODF.	33
12	Main components for Office Open XML.	35
13	Example markup of the content file of OOXML.	36
14	The converters website	49
15	The first sketch of the system.	50
16	The last sketch of the system.	51
17	Parsing the content files in the three formats.	52
18	Rule and Dependency tables.	54
19	The activities of the Mapping module.	55
20	The convertfile step in detail.	56
21	The ODT file with the HTML contents.	59
22	The dependency table.	73
23	Another possible database setup.	73

1 Introduction

Information flows are changing from being mostly paperbased, to being based on digital mediums. This development has provided both negative and positive side-effects, whereas we can share information across greater distances at greater speeds, but it has also increased difficulties in validating the vast amounts of data being shared. Included in this validation is the potential data loss in a document, in addition to the applications ability to correctly represent the transmitted data.

Andrew Updegrave tells about how the facilitation of Latin in the Western world and Arabic in the Islamic world led to the sharing of ideas across international borders [66]. Sharing a language enabled a new line of communication and sharing of information that was priorly inaccessible without a common way of communicating. Another great example for sharing knowledge, and that also enabled historical information to be accessible, is the art of printing. By putting signs and characters to print, humans learn our history from the printings in a cave wall, the Dead Sea Scrolls, and the present day history books. The information in these historical documents touch on every part of being a human. We have learnt a great amount of human history pairing up printed information with scientific information. An example of this is Darwin's theory of Evolution paired up with fossil findings, where the latest news is the believed uncovering of a fossil that could be one of our earliest ancestors, at the root of the branch into two species, ours being one of them[1]. The apperent flaw in historical documents is to ensure their validity, and to separate personal belief from science. A school science book in Kansas, USA might tell a different tale of human origins than a school science book in Oslo, Norway.

Our legacy for future generations might be the data-sets and documents created today, and having a way to access and understand them is therefore of great importance. In the same respect as language is a facilitator for sharing, open standards should work as a facilitators for sharing office documents. Include the World Wide Web in this picture, and you will have the opportunity to instantly share your information across the world the moment you publish it. This information will be easily accessible by anyone with a browser, and Internet access. Imagine the consequences of being able to instantly share the invention of the wheel across the world when it happened. Or at present time, sharing information about how space voyages are made, rapidly increasing the progress of space travel and understanding of space, with a larger set of scientists sharing a pool of information. In the IT industry, an entire field is focused on knowledge management. To avoid inventing the same artifact twice, and to learn from colleagues endeavors. Although the political aspects of sharing information across national and local borders is recognized, the scientific rewards are viewed much greater.

There is a need for knowing how to encode your information and what formats you should store your information in, in order to reach a wide audience. Or sometimes, you might need to know what a particular audience uses to make sure they get vital information. What if you change hospital and the data they send between them are corrupted by the transfer method? Or the software which is used at the different hospitals does not read your health information in an appropriate way?

The goal of this thesis, is to contribute to the knowledge sharing revolution. The piece of the puzzle this thesis will focus on is between two of the widely used office document formats, and the HyperText Markup Language (HTML) used to creating websites. The office document formats are Office Open XML (OOXML) by Microsoft, and Open Document Format (ODF) utilized by the OpenOffice.org application.

A system was created to address the aforementioned goal. The aim of the system is to convert between these three formats. Details of the system can be found in Chapter 4. The reason for including HTML is the easy access over the web, and the wide reach of this format. The standard version utilized for this will be the HTML 4.01 specification [67]. The final artifact will not be a full set of rules converting between these three formats, but instead a possible framework to ease conversion of the formats between each other, with the ability to convert a small set of equivalent elements.

Because the formats represent the same type of data, there is a notion of an overlap between the formats. As figure 1 illustrates, there is an anticipated intersect between the three formats, and this paper will investigate if the intersection is enough to create a converter between the forms.

In addition to easing conversion between the three aforementioned formats, the framework will be constructed in a way so that future overlapping standards could be added with ease to the conversion application. Hence, the possibility for it to be a generic converter for all sorts of overlapping formats is present. It will be released as an open source project, so that it can be a starting point for a complete open source converter that may be derived from my work, or even count as a contributor to a merger between these formats.

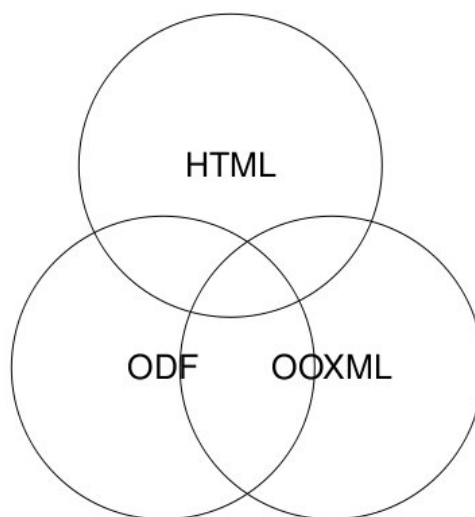


Figure 1: Intersection of HTML, ODF and OOXML

Important aspects like legal rights to documents and trust of data are recognized, but will not be regarded in the solution of converting between the formats. This is because it will have little implications on the actual conversions performed in the system created in this thesis.

1.1 Problem Definition

There is one main question, followed by some subdivisions branching into theoretical and technical questions.

The main inquiry is presented below.

- Is it possible to convert between OOXML, ODF, and HTML?

In the theoretical part, standards are in focus, touching in on Human-Computer Interaction (HCI) and universal design.

- What is a standard?
- Do open standards help to enable communication and universal design?

The technical questions are related to the framework being created, and the use of today's open source technology.

- Is it possible to create a framework that eases conversions between overlapping formats?
- Can a framework that converts between overlapping formats be developed successfully using open source technology?

1.2 Problem Area

HCI has gone through changes as the technologies have advanced, with focuses on different aspects of this interaction. A closer look at this term is found in section 3.2. In the first wave of HCI, where computers were used primarily for work purposes, the focus was logically on creating an efficient environment so that people could do their jobs optimally interacting with their workplace. As the second generation advanced, a broader notion of interaction with computers appeared, but still with its main focus on the workplace[46].

Susanne Bødker talks about the challenges for the third wave HCI when it reaches beyond the workplace[46]. She explains that in second wave HCI, the focus was mostly on work situations, but as technology advances, we are using technology in a greater extent with multiple mediators. Few homes

are without a computer, and leisure activities are growing in scale in interaction with them. Susanne claims that with this evolution towards the third wave HCI, where cultural factors are extending, and technology used in a broader sense, a broader angle of approach is needed within the field of HCI.

In parallel, office documents, such as text documents, spreadsheets, and presentations, are to be shared across technologies that extends the use from beyond the workplace. This paper will focus on the activities of writing and opening an Office Document, or a website on a computer. These documents are often communicated through a computer, and open standards play a big role in an applications ability to successfully perform these activities, particularly in the mentioned opening and writing, but also with the coupled activities of editing and saving a document. To illustrate, a few examples of possible situations that could be solved with a converter are listed below.

You are working on a project in a medium sized group, and want to share a text document with another group member. You have OpenOffice.org, and save your file with a .odt extension. You e-mail it to your group member, but he/she fails to open it, because she has Microsoft Word, and this office document package doesn't open this format. The solution often being that the group decides on either OpenOffice.org or Microsoft Word, as a standard to exchange documents in. The difference between the two being that OpenOffice.org is free, while you have to purchase MS Office.

You are working on a document at work using Microsoft Word, and want to finish it at home. When you open up the document in your home application, OpenOffice, and the document has lost its formatting and some images are lost. In order to get the same formatting you have to put in a lot of effort in changing the document to match the same formatting as you had at work, or get Microsoft Word at home.

A school child is writing an essay for a school assignment. He uses the application on the computer, called OpenOffice.org, and doesn't think much of it. When he saves the document, it gets a .docx extension. He mails it to his teacher, and considers his assignment delivered. The teacher uses Office 2007, and is unable to open the students essay. Two solutions possible outcomes for this unfortunate situation: the student now have to write the essay again using another application, or the teacher has to install the same application it was written in, namely OpenOffice.org.

These are examples are explained with just two applications, with two different open standards, with one office document type; text. Adding more variables would make the picture even more chaotic. Without a common format, these documents can not be shared.

These kinds of overlapping standards that are used in competing applica-

tions prevent efficient sharing of documents.

In the situations described above, a few main areas of interest are recognized. These being open standards, open source, communication, information which needs to be saved in a responsible manner, and a format to store the information in so that it is accessible without discrimination of hardware, software or person. This is the core of the problem area.

The user group is considered to be those who can read and write documents, ranging from children to the elderly.

1.3 Definitions

- Standard - A market driven specification that enables communication.
- Open Standard - A market driven specification that enables communication, that is freely available to use and implement.
- Open Source - Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. [2]
- Format - The arrangement of data for storage or display [3].
- Open format - Any format that is published for anyone to read and study but which may or may not be encumbered by patents, copyrights or other restrictions on use a specification for storing digital data[4].
- Namespace - A defined building block used for a single purpose, belonging to a defined tag-set of a format.
- File - A file is a named collection of related data that appears to the user as a single, contiguous block of information that can be retained in storage (e.g., a hard disk drive, CD-ROM and magnetic tape) and accessed by its file name[4].
- Formatting - The presentation of e.g. a file.
- Semantics - The meaning of e.g a file.
- Markup - A set of elements larger then zero.

1.4 Motivation

Having the ability to share information across borders, time and platforms is vital for exploiting the collective potential in the human race. On many occasions, the greatest inventions have been deduced or been the consequence

of prior explorations, made by different individuals or groups. Available and easy accessible documentation is therefore recognized to be a bottleneck for human innovation. The enabling of sharing being the main vision, it is equally important to have no discrimination of access to public documents based on the format it is stored in.

Office documents are often stored in the format the organization has decided to use for communication. This may result in difficulties when the organization is to share information to external entities, or if the employees work from home with a different environment. Additionally, there are difficulties if the format is not backwards compatible with prior versions of the application utilizing prior versions of the format.

In writing, you can not successfully open an ODF formatted file in a Microsoft Word application. The opposite is working to some extent, but with some deterioration in validity and formatting. More and more countries see the benefits of using open standards for their public information flows, and many are choosing it as a obligatory format for public documents. Norway legislated ODF, HTML and PDF to be the open formats in which all public documents are stored in. Microsoft have promised to support ODF in Office 2007 in the future, which is a great step towards not letting standards stand in the way of sharing information. This expansion of open standards is one the thesis wants to take one step further, by introducing HTML as the third format to perform conversions between, in addition to OOXML and ODF.

The system this thesis provides, not only facilitates conversions between the office document formats and HTML, but also serves as an aid to future conversions between overlapping standards, in both document formats and possibly other formats. This is because it is assumed that these overlapping standards are probably not the only occurrence of such a situation.

As mentioned above, the importance of accessing documents for both historically and present day is recognized, and is also a part of the motivation for this thesis. This also includes the safekeeping of present day historical documents for future generations.

1.5 MOSCITO

This thesis is a part of a project called MObilizing Social Capital in global ICT based Organisations (MOSCITO). A project trying to gain insights on how people communicate internally in organizations, and how the choice of those tools utilizes the social capital within.

This thesis takes a look at digital sharing of documents, which is a great part of communication both in organizations and out of the office, and tries to deliver a system that performs conversions between the focused formats.

1.6 Chapter Overview

Chapter one will give an introduction to the thesis, and what context it is in.

The second chapter describes the methods utilized in this thesis.

In the third chapter, we take a closer look at the theory studied and applied in the thesis.

A description of the case is found in the fourth chapter. Here we follow the construction of a conversion framework.

The fifth chapter contains the findings from the case, and the interviews.

In the sixth chapter, we discuss standards and the framework in the context of the findings and the theory.

Chapter seven contains the conclusion of the thesis, containing both proposals for further work, and pointing out vulnerabilities of the thesis.

2 Methods

In this chapter, the methods utilized in the thesis will be described. It is split in two, the first section describes the methods used to gain knowledge. The methods applied for development of the system itself is described in the second section.

2.1 Gaining the knowledge

There were three main methods used to address the problem area. Those were document analysis, attending conferences for informal talks and lectures, and semi-structured interviews. These methods will be elaborated in this section.

2.1.1 Document analysis

All documents touching on the problem area were of interest to the thesis. Two master thesis' from the University of Oslo were studied, to gain insight on universal design on the web. These were supplements to books, articles and websites written by people within the industry.

The information from the Internet was accumulated with caution, seeing that there is a potential occurrence of misdirection. Especially looking into the debates regarding OOXML and ODF, there is an excessive amount of material to study. It is sometimes hard to distinguishing a fact from a personal opinion.

2.1.2 Semi-structured interviews

The semi-structured interview is a qualitative interview method. This form of interview was chosen due to its balance between richness and comparability, where the conversation is guided by a script but accepts venturing deeper into discussed topics when seen fit[63].

These interviews were conducted to get information from people within the field of standards. The focus of these interviews were what standards are perceived as in real life, outside of books and academia, and what the interviewees experience with open document standards were. The findings of these interviews can be found under chapter 5.

The main topics for discussion in the interviews were standards in general, and office document standards. The script for the interviews are in Appendix B.

The semi-structures interviews were done with two people within the industry of IT-standards, in their workplace. With consent, the sessions were audio taped. The consent form can be found in Appendix A.

2.1.3 Conferences

Attending the largest Nordic conference regarding open standards and open source software called Goopen in Oslo 16-17 April 2009, and also the preceding year, was a great source of information and relations. The conference in 2009 is organized by Norwegian Open Source Center. Their aim with accommodating the conference was[5]:

To help and facilitate the use of open source in the public sector, to help and facilitate Norwegian and international vendors to use open source and open standards in their products and also to help and facilitate developers and investors to see business opportunities within open source.

In 2008 the conference was organized by Friprogsenteret, who's vision is "to encourage sharing, reuse and cooperation in order to provide a genuine free choice for everyone"[36].

2.2 Development method

The method used for development was the iterative development method.

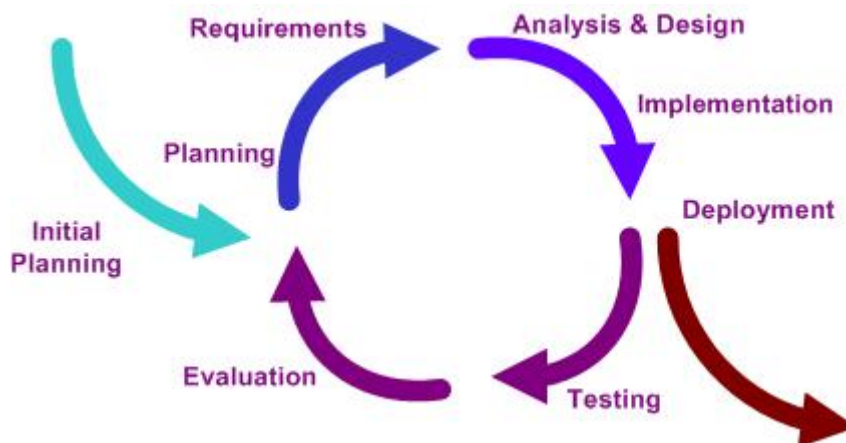


Figure 2: Iterative development.

As illustrated in figure 2 [6], the iterative development method is cyclic, repeating the steps of the development process as needed. This method is agile, with the aim to reduce development overhead in order to produce

software faster [64]. Since the development of a converter is a priorly unchartered area for the developer, the environment is anticipated to change, as having a consistent set of detailed requirements are unavailable.

Iterative development is a Rapid Application Development (RAD) process. The criteria of such a process according to Sommerville[64]:

- The processes of specification, design and implementation are concurrent.
- There is no detailed specification and design documentation is minimized.
- The system is developed in a series of increments.
- End users evaluate each increment and make proposals for later increments.
- System user interfaces are usually developed using an interactive development system.

All of these criteria fit the development process of the system in this thesis.

3 Theory

In this chapter, we take a closer look at the theories applied to the thesis.

3.1 Communication

To fully understand what part of the world the paper wishes to address, we take a deeper look into communication. Weaver explains that the word *communication* can broadly be defined as a procedure by which one mind may affect another [62].

The procedure being focused on in this paper, is the procedure of sharing a document. This certainly fits the broad definition, where the sharing of a document, may affect the receiver of that document. Hence, we will apply this task to a suggested communication system to elaborate on it. Shannon proposes a schematic diagram of a general communication system as shown in figure 3, including a noise factor in addition to Weavers proposed schematics[62](p 32).

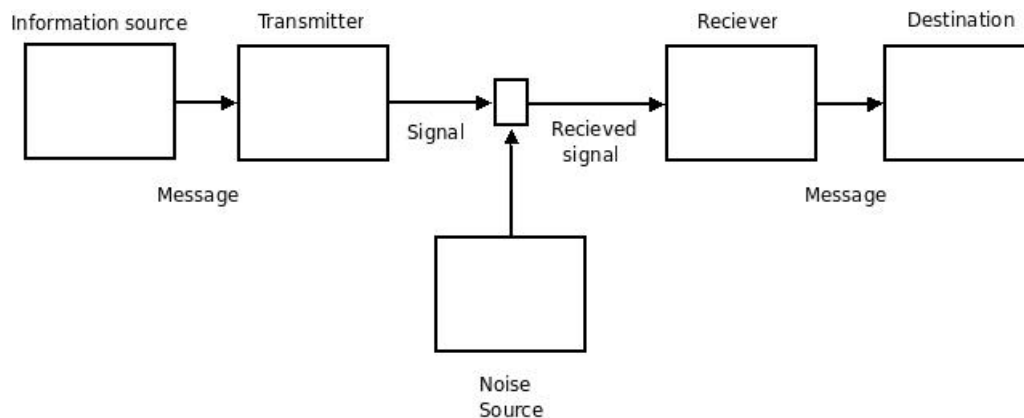


Figure 3: Shannon's schematics.

The *information source* is what produces a message or a sequence of message to be communicated. A *transmitter* operates on the message and encodes it to be suitable for transmission of the channel. A channel is defined as what transmits the signal from the transmitter to the reciever. The channel in our task would be accessing and retrieving information through the World Wide Web using TCP/IP. The *receiver* performs a decoding operation, reconstructing the message from the signal. And the *destination* is the person of object of which the message is intended. While there are many possible sources for noise, a noise source in this situation could be corrupt data, occurring

under transmission, or information loss about formatting or contents of the information source when the destination application opens the file.

An illustration of how the schematic for our task is found in figure 4.

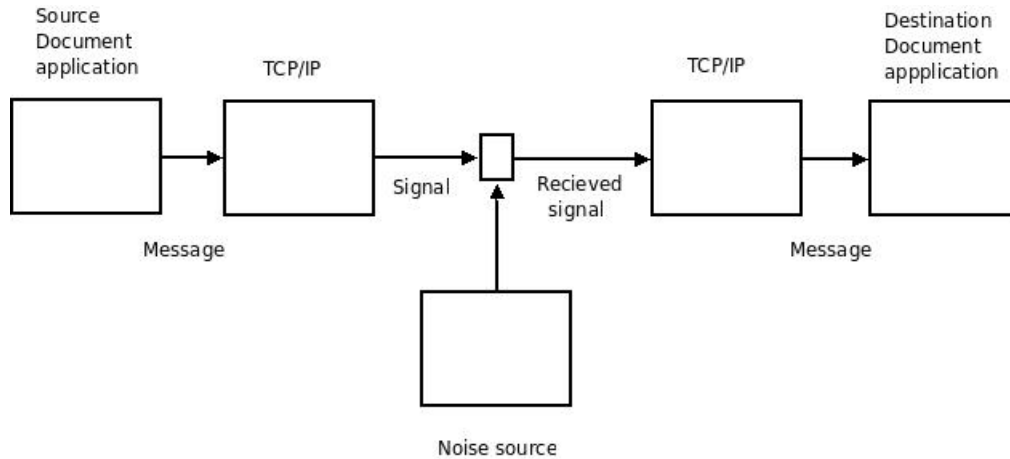


Figure 4: Shannon's schematics applied in this context.

This illustration assumes that the application is in a context where there is access to the Internet, and there is an ability to send the message on the channel. Also, that the document application permits reading, writing, editing and saving.

Weaver explains that the structure of communication is often statistically controlled. In communicating text or words, this means that your next word is often statistically decided by the probability of its occurrence after the current word. Much like after "in the event" the probability for "that" as the next word is high, and the probability for "elephant" is very low. This is called a Markoff process, or a Markoff chain [62]. Hence, the redundancy of communication is measured in how many words could be built from probabilities determined by the previous word.

3.1.1 Possible problems

Weaver recognizes three different levels of problems that can occur during communication. With each of the three problems, he attached a question to illuminate the corresponding problem. The three levels are[62]:

- A: The technical problem. How accurate the symbols of communication are transmitted.
- B: The semantical problem. Illuminating the question of whether the symbols conveyed carry the desired meaning.

- C: The effectiveness problem. Regarding how the received meaning affect conduct in the desired way.

All of these levels of problems apply to our task of communicating a document. The levels are all related and sometimes overlapping, where one depends on another.

In level A, the finite set of symbols being transmitted needs to reach the receiver in the same condition as they left the sender. The finite set of symbols being the set of graphic signs that make up the document.

The semantic problem, is perhaps the most difficult, where interpretation and desired meaning should be approximately the same to achieve a successful communication. More specifically to our task, the document being sent will try to communicate a desired meaning. For example we may consider heteronyms as a valid source of misinterpretation, where one word has different meanings. Consider this simple example: *After having a row, the couple went out on the river to row.* "Row" as a verb means to paddle a boat, while "row" as a noun means an argument. In this sentence, you have to interpret the meaning. To convey the desired meaning of a document, the receiver has to interpret the sentence in the way it was intended to be.

The effectiveness problem is concerned with the consequence of a successful communication, and whether it leads to the desired conduct. It is closely related to level B where the desired interpretation of the communication will lead to the desired conduct. The simplest example of this is propaganda or commercials.

An enabler of communication, is universal design. This is much grounded in what may be the main principle of universal design, which is equitable use. Equitable use is when the system provides the same means of use for all users, identical whenever possible, or equivalent when not [7]. This contributes to the creation of both open and closed standards that helps to perform tasks in such a way that everyone can participate, without discrimination. In relation to our task, these standards offers a valuable grounding for successfully communicating messages by using a common channel like the World Wide Web, and also a common document application for viewing and writing documents.

3.2 Human-Computer Interaction

Since computers are involved in the communication activity, and the interaction between humans and computers is a great part of the communication being performed, we touch the field of Human-Computer Interaction(HCI).

An article from ACM, named ACM SIGCHI Curricula for Human-Computer

Interaction written by Hewett et al., explains that there is no agreed upon definition of the range of topics which form the area of HCI [54]. However, the article attempts a working definition:

Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.

Hewett et al. narrows the focus down regarding computer science to be interaction between one or more humans and one or more computational machines.

In his book, Johnson defines HCI as the study of the interaction between people, computers and tasks [56], and catalogues it as a sub discipline of computer science. He places its concern with all issues of computer science and sees the main challenges to be situated within the input and output of a system. Having the output make sense to the user, and the input make sense to the computer.

These two definitions overlap each other in the sense that they both describe a context to comprise of one or more humans and computers, and a tasks being performed within that context. This seems to be a recurrence in most attempted definitions of HCI. In correlation with our task of sharing a document, we can see the obvious conjunction with the interaction between the human and computer to perform that task of communicating a document over the Internet.

To clarify the interaction of sharing a document, we perform a shallow task analysis, as shown in figure 5. This is to analyze the human behavior of the task, and identify actions, objects and procedures according to Johnson's guidelines [56]. The actions included are writing, saving, reading, sending and receiving. The objects in this context is the document being communicated. The procedures above are in order if the task is for a human to use the computer to send a document using the A markings, but procedure 1 and 2 must be reversed if the computer is used to receive a document, following B markings [56](p177-189).

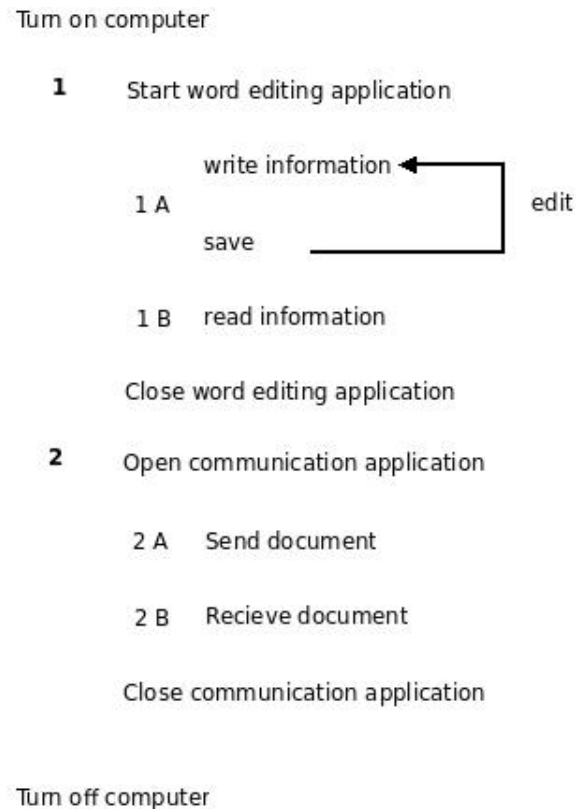


Figure 5: The task of communicating a document.

3.3 Standards

This section will be split in three parts. The first part will look into what a standard is perceived as. Secondly, we will look into who makes the standards focused on in this thesis. In the last section we go in deeper detail regarding the history of the standards focused on in this thesis.

3.3.1 What is a standard

It appears that the notion of a standard is fairly subjective matter. Although medium like wikipedia and encyclopedias attempt to form a definition, these definitions are not recited, but rather the subjective understanding of the term. In this section, we take a closer look at what a standard is, and go in deeper detail in open standards.

There were conducted semi-structured interviews with people within the field of standardization to dive into the reality of this question. The questions can be found in Appendix B, and the results can be found in chapter 5. Below, the term standard is explained through commonly referred source, namely Ken Krechmer, accompanied with a thought experiment of how many standards really are in use in a given situation, and an overview of the evolution of the standards related to the thesis.

There is a range of definitions made for what a standard is. All from the simple and general definitions like the European Telecommunications Standards Institute suggests[8]:

A set of rules for ensuring quality.

To the more elaborate definition from MIT Laboratories[9]:

A prescribed set of rules, conditions, or requirements concerning definitions of terms; classification of components; specification of materials, performance, or operations; delineation of procedures; or measurement of quantity and quality in describing materials, products, systems, services, or practices.

The lowest common denominator in these definitions seems to be that a standard is an organized collection of guidelines or rules.

Ken Krechmer defines a standard as “a representation of common agreements that enable communications, directly in the case of Information technology (IT), and indirectly in the case of all other standards”[58]. Combining the definitions, we end up with a definition that states a standard is an organized collection of guidelines or rules that enable communication.

The part of the standards world we will concentrate most on in this thesis

is open standards, acknowledging that the focused standards are all open standards.

Krechmer sets a clear difference between standards and open standards. Recognizing that an open standard, in the IT-industry, is created through an open process, and is free to use and implement, while a standard not necessarily have these properties. For example, the predecessor of DOCX; DOC is not part of an open standard. Constituting the possibility to freely use and implement DOCX which is part of the open standard OOXML, while this is not possible with DOC. Correspondingly, the same distinction can be found between a format and an open format, where the open format is free to use and implement, while the proprietary formats are controlled by private stakeholders. More about formats and open formats is in section 3.5.

Similar to a standard, there are many definitions for what is required from a standard in order for it to be open. We will take a look at Ken Krechmers requirements for a standard to be open, as it is viewed the most strict and including list of requirements.

Krechmer acknowledges the wide range of definitions, and in addition points out the different economic needs in the groups that create, use and implement open standards. He states that the creation of open standards is driven by potential market development and control. The implementation of open standards is driven by production and distribution and cost efficiencies. While the use of standards is driven by the potential in improving efficiency both in a political and financial manner[58]. This provides three groups of stakeholders; namely creators, users and implementers. He states the different interests are important to echo in the definition of standards, to make sure the definition is not bias in any direction. Krechmer goes on to explain the ten criteria he has gathered to be in place for a standard to be labeled open. They are[58]:

- 1. Open Meeting**

All stakeholders may participate in standards development process. The largest barrier for this is considered to be the economic barrier to be a member of a standard setting organization(SSO). Many SSOs open for any stakeholder to pay to become a member, and to lower the economic barrier, accept cost to join on a per meeting basis. However, the possibility to join is enough to consider this requirement fulfilled.

- 2. Consensus**

This is defined differently in different SSOs. All interests must be discussed, and an agreement must be found, without any domination. A requirement for a purposeful consensus, is that all stakeholders must be represented equally.

3. **Due Process**

This is also a criteria described differently in different SSOs. In order for an open standard process to be a due process, is that all participants are informed of all actions and inactions, and have the ability to turn to an appeal mechanism if needed to reach a resolution.

4. **One World**

This is the principle of wanting one worldwide standard for a single purpose. The World Trade Organization supports this requirement to prevent technical barriers to trade. Some cultural and political difficulties are recognized in this criteria. The open world criteria is assessed by identifying the geographic reach of each SSO. The larger area covered, the better it will meet this requirement.

5. **Open IPR**

The holders of Intellectual Property Rights(IPR) related to an open standard must grade their IPR on a Reasonable and Non-Discriminatory (RAND) scale for its implementation. This scale is used for this purpose in most SSOs and consortia.

6. **Open Change**

This requirement ensure that necessary system changes or upgrades made to the open standard also needs to be equally open, to avoid monopoly through upgrading. All of the changes made should be presented in a forum supporting the five requirements above.

7. **Open Documents**

This requirement is regarding the ability to see any documents from the standards process, be it a work in progress, or a final document. There are three main levels of transparency:

- (a) Work in progress documents are only available to committee members, and the standards are for sale.
- (b) Work in progress documents are only available to committee members, and the standards are available for little or no cost.
- (c) Work in progress documents and standards are available for little or no cost.

8. **Open Interface**

Krechmer acknowledges that this requirement is a fairly new one. It points at a systems talent to integrate, and having the ability to be backwards compatible, as well as compatible with future systems that share the same interface. An open interface gives less control to the maker of the standard, and supports migration. This proposed requirement might be more significant when applied in SSOs.

9. Open Access

Open access is a requirement regarding accessibility. Conformance to both safety and non-discriminatory requirements are highlighted. This to ensure that people with disabilities have access in an equal manner as people without disabilities. And also to meet with safety, environmental and health certificates like the European Commissions CE mark, or the worldwide UL mark for safe use of equipment[10]. Krechmer identifies two levels of containing conformance regarding open access; one with emphasis on the implementer, where interoperability is the goal, and the second with emphasis on the user, where the universal design principle is addressed.

10. On-going support

The user of a standard has a specific interest in this requirement, where they want standards to be supported until user interest ceases instead of when implemented interests ceases.

Krechmer points out that the first three requirements are addressed and largely resolved in most SSOs and consortia. However, beyond the first five requirements, most SSOs do not fully address the ten requirements listed above. The reason for this is that the five first requirements are those who matter the most to the creators of standards. While the other five remaining requirements are largely centered to the users and implementers, only separated with on-going support not necessarily being a part of the implementers interests.

These ten requirements for an open standard, have a very broad view of what a standard is, and it is extended far beyond the IT-industry. In his article, Krechmer explains that few if any SSOs meet all ten requirements. He sees the argument that with a narrow view of what a standard is, fewer requirements may be needed, but claims it is not a sufficient argument, seeing that the consequences of giving up some of the ten requirements can have larger implications than initially foreseen.

The working definition of an open standard in this thesis is simplified to being a market driven specification that enables communication, that is freely available to use and implement, as stated in section 1.3. This abridged definition is fitted for the problem area of this thesis, expanding on the definition of a standard uncovered earlier. It does not contradict Krechmers requirements, since none of them are excluded just by narrowing down the definition. Krechmers requirements are believed to be a good goal for the future, and a solid and detailed definition of what is required in order to be open. This is much needed in the industry, noting the number of definitions that exist at present time, and the differences in them.

3.3.2 Thought experiment

Figure 6 shows a thought experiment where we think about how many standards actually interplay in the communication of a single document. The activity depicted in the figure also includes the activities required in order to communicate a documents, whereas writing and reading are some of those activities. To make the picture less chaotic, the subsets of all the standards are not included. For example in the layer of task at hand, where the user is centered, opening to read a document has a large subset of open standards in place for achieving this activity. This subset contains among other sets, accessibility standards to enable visually impaired or people with other disabilities to view the information.

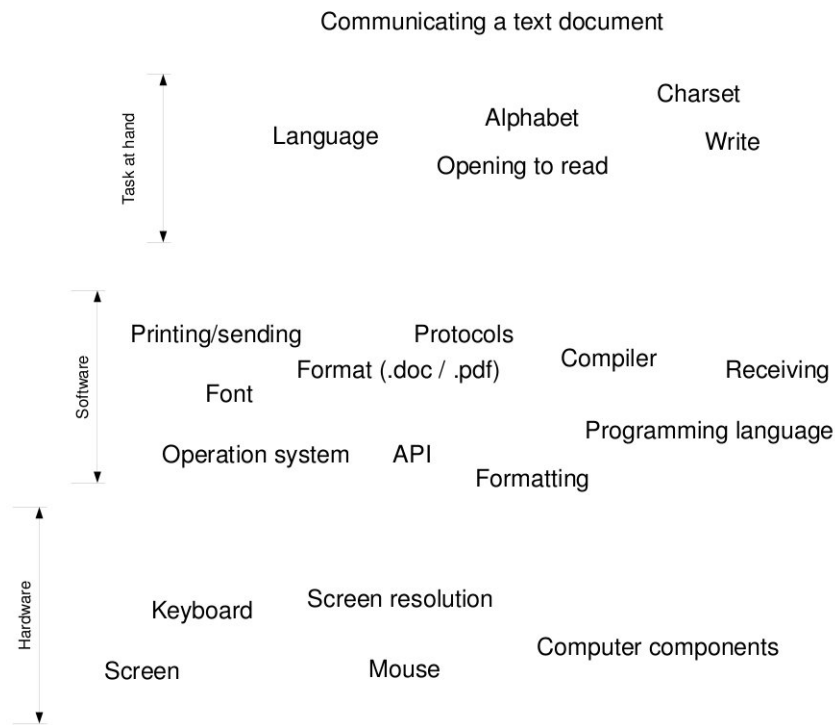


Figure 6: How many standards for communicating text.

Contemplating the amount of time and effort went into all of these standards and specifications, only to be able to successfully communicate a document, it could be considered a wonder that they interplays with a favorable outcome, but this is also the reason for their existence. It visualizes how simplifying it is to have the same standard for a single purpose, when shifting out one of the standards in figure 6 will have great consequences. The worst case being failing to communicate the document.

3.3.3 Evolution of standards

The timeline in figure 3.3.3 shows the creation of standards related to this thesis from the creation of the World Wide Web until microformats in 2004 [40]. A clear trend can be spotted that the formats try to become more human readable as time goes. The aim at the start of the computer era was to make the computer understand, and the amount of competent people being able develop and interact with these standards was fewer, and with more specific knowledge.

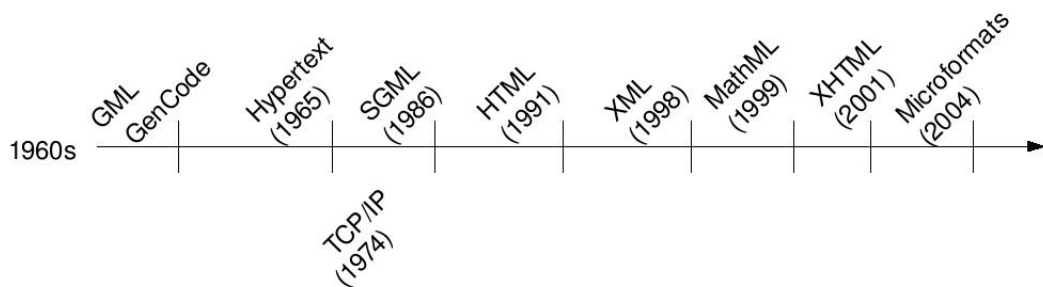


Figure 7: Timeline of standards.

As we strive towards a human readable, and computer processable format, more and more people are able to comprehend the knowledge needed to utilize the technology at hand. This is a result of good standards, and creates a diverse community of users and developers.

3.3.4 Who makes standards

In theory, anyone can make a specification for something. But no matter how great that specification is, it will not do any good if nobody uses it. In order to reach a greater group of people, even in nation and global extent, there are established standard setting organizations (SSO) and consortiums.

This grounds the purpose of standards, where having one standard in common for a large group of people leads to easier sharing of information. Enabling this for different platforms and applications, in addition to doing so without discrimination. And also, the chance for a standard to be a good standard increases with the amount of people working on it, seeing that the entirety of a situation is more probable to be recognized with many people, illustrated in figure 8. The reason for the acceptance and reliance of big

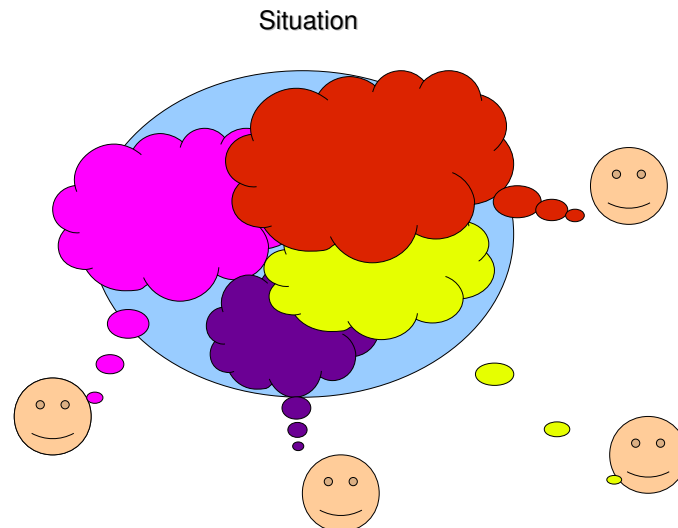


Figure 8: Different people see different parts of a situation.

organizations to set standards is dual. One is to get consensus of in-depth knowledge of the group of people from various background to get the best standards as possible as mentioned above, but an additional consequence with having a large userbase where sharing is easier, the business side must also be considered, where more money comes your way when your userbase is large. The committees are in these groups to ensure the standards are well suited for their purpose.

Below, we look at organizations or consortiums that specializes in office document and web standards including the focused formats. The difference between a consortium and the standard setting organizations are that the

consortiums are not recognized standard setting organizations[58], meaning that they only have the authority to recommend and not constitute. Many of the organizations described below have close relationships with each other, as would be expected when covering some of the same area of interest.

OASIS Organization for the Advancement of Structured Information Standards (OASIS) defines themselves as[11]:

OASIS is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

The most significant standard development in OASIS with relation to this thesis is ODF. They submitted the ODF standard to ISO for approval, and it was published as an ISO/IEC international standard in 2006.

The way for a specification to become a standard in OASIS, according to their website[12], is to have it approved within an OASIS committee, submitted for public review, implemented by at least three organizations, and finally ratified by the consortium's membership at large.

OASIS is a consortium, with 5000 participants over 600 organizations and members in 100 countries[13]. Their leaders are democratically elected, and are independent of financial or political standings.

They have an active technical liaison relationship with ISO, and a few other organizations. In ISO they participate in the technical work in e-business and also in standards development work in the subcommittee for document description and processing languages.

ISO International Organization for Standardization (ISO) is an organization devoted to creating standards for government use, and also business and society [14]. ISO develops standards for different industries. It is not owned by any government however, so it is not swayed in any direction for financial or political benefits.

This organization has a great influence over what standard is broadly used, being the largest developer and publisher of International Standards. Organizations often submit their standards to ISO for approval, seeing ISO is the largest SSO with the longest reach.

The standards development in this organization that is significant for this thesis is the work on publishing OOXML and ODT as open standards.

In ISO, a standard is made according to some main principles. One of those is that the views of all interests are taken into account, this being everything

from research organizations to manufacturers. Also, the standard needs to satisfy industries and customers worldwide, and not cause restrictions. They also develop international standards based on voluntary involvement from all interest in the market place. Further explanation of the standards process can be found in their website [14]. ISO has the vision that a standards should meets the requirements of business and the broader needs of society.

Ecma Ecma international is a non-profit standards organization, that has the aim to contribute with development of standards who facilitate and standardize the use of Information Communication Technology. [15].

They approved OOXML in 2005, which came to be a published standard with ISO in 2008[16].

The structure in this organization is flat. In order for a standard to become an Ecma standard, it needs a formal document prepared by an Ecma Technical committee and approved by the Ecma General Assembly. A majority of at least two-thirds of all the ordinary members are required for approval by vote. The standards are submitted twice a year to a General Assembly for approval and publication. More detailed information can be found in their website [15].

Ecma occasionally submits their standards to ISO/IEC to use their approval procedures.

W3C World Wide Web Consortium (W3C) is an international consortium. Their mission is [17]:

To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the web.

The standard they develop, with significance to this thesis, is HyperText Markup Language (HTML). This is the publishing language of the web, invented by Tim Berners-Lee in 1989. He is still involved today, as a director of the W3C [18]. Originally, HTML was not sustained by W3C, but was a published ISO/IEC standard in the year 2000. Now, W3C maintains and develops the specification with the mission of ensuring long-term growth for the web as mentioned above.

They have done great work in providing services for validation of websites, and guidelines for constructing websites. This aids the casual homepage-maker in constructing a valid website, as well as giving a quick debug tool for programmers, and ensures that a website will show as intended in most browsers.

HTML is considered a standard due to its wide reach and world wide adaptation. But it also scores high on Krechmers rating of openness (As the only one of the standard setting organizations or consortia mentioned in this rating), only lacking in the requirements of Consensus, Due Process, Open World and Open Change.

In W3C, specifications are decided upon by the member organizations, a full-time staff, and the public. They do not apply a majority rules voting system, and in the end, the director will have the final decision.

In common The standard setting organizations and consortia explored above, all have in common that they have published or recommended the standards significant to this thesis. Secondly, these open standards for documents are derived from SGML. More about this in section 3.4.

3.3.5 Implications

By looking at how standards are perceived, and adding it with what is the ideal purpose of a standard, we get a view of the implications of a standard. Regarding this thesis, we can understand that the document format standards have the goal of enabling the writing, reading, and sharing of a document. However, we see that when formats are overlapping, it might create a situation that differs from its intended implications.

On another hand, the ideal implications are not the only ones to be considered when recognizing standards impact on the world. Public governments have started to see the benefits in using open standards to store their public documents and information. This is partly due to its low cost, and ease of access. The Norwegian government legislated that from 1st of January 2009, all document published by the government must be stored in open formats. They chose ODF, PDF and HTML as the three formats to be used for different purposes. HTML for publishing public information online, PDF when they want to maintain the original formatting and looks of a document, and ODF on documents you are supposed to change the document after you have downloaded them [39]. This is not the only government that has decided on using open standards for their public information. Vietnam uses ODF as their national standard for documents, as well as Hungary [19]. As more countries adopt open standards for their public information, this has large implications on what tools the public uses to view information.

This decision on using open standards for public documents, will influence a lot of lives and businesses, including how information is handled in the public. Just the fact that you can, without extra cost of proprietary applications, view the public documents is a great improvement, but this also

promotes the open landscape of the IT-industry in a great way. It challenges those with proprietary formats to change to ensure a viable longterm business model that is prepared to compete with the trend of open standards and open source.

3.4 Office documents

This section contains in-dept information about the open standards that are being concentrated on, namely the office document standards OOXML and ODF, and the publishing language of the web; HTML. We start with SGML, which is the predecessor to all three formats. Next, we go into detail about HTML, succeeded by OOXML, and ODF. In the end of this chapter the overlap of the formats will be outlined.

3.4.1 SGML

Before we look at the three open standards being considered in this paper, we take a look at their origin; Standard Generalized Markup Language (SGML).

Dan Connelly et al. writes that in the 1960s, there was two separate companies who saw the need for a format to handle the structured documents on computers [47]. The need spurred from either within the company, or as a business enhancement. Two tag-sets were created, namely the GenCode and GML, but they were not generic enough. This led to a branches and diverse formats, suited for specific needs. These different formats were the first template formats we call Document Type Definitions (DTD), which we still have today, used primarily to validate the documents, and make the parser aware of what tagset it is trying to make sense of..

With DTDs, there became a clear difference between a document being “well formed” and “valid”, whereas a document is valid if it uses the document structured in the DTD’s specification, and it is well formed if the document is achieved when it fulfills the basic grammatical rules of the format, such as all elements being properly nested.

SGML first became an ISO standard in 1986, and contained definitions for portable document formats, and was formal enough to allow proofs of document validity. ([47]). It was the first attempt at a structured way of handling documents. But it was too ambiguous on some points, and covered too broad an area to have an overview of the format. Plus, there were too many optional features that confuses the users of the format. SGML didn’t have the possibility to represent special characters such as mathematics and chemistry related annotations and various expressions. It played a role as

an interchange language to communicate and manipulate text documents [47].

An example of the SGML grammar:

```
<mytext>A text example</mytext>
```

It depicts what we may recognize as the common structure of any of today's eXtensible Markup Language (XML) based formats, which is derived of SGML. It consists of the element name in plain text, surrounded by angled brackets. The start tag `<mytext>` tells us where to start to look for the contents of this element, while the end tag `</mytext>` identified by the `/` symbol, representing the closing of the element.

The element name in the start and end tag must be identical. In between these tags is the mytext elements content "A text example". The whole structure with contents, start and end tag, is referred to as the element. These are the basic syntax rules, in addition to the requirement of proper nesting of elements. To illustrate this nesting, a new example is found below:

```
<root>
  <parent>
    <child>
    </child>
  </parent>
</root>
```

Each element must be opened and closed within the same scope, in order for the document to be valid. These are the basic rules of how this format and also the formats concerning this thesis is constructed.

Even though the three focused formats differ in many ways, all three share the syntactical rules described above.

XML, which is derived from SGML, is considered a neutral language where it is independent of both platform and application. Its hierarchical structure and freedom to choose own tag names is also favorable attributes to this language [57]. Paired up with a suitable DTD, lies the freedom of constructing information in any way you desire. This freedom is utilized in the office document formats, as well as the HTML format, and will be further elaborated below.

3.4.2 HTML

HyperText Markup Language (HTML) was originally defined by Tim Berners-Lee and Robert Caillau in 1989 [20]. The idea of hypertext however, dates

back to 1945 when Vannevar Bush introduced the memex, he also anticipated a way to link a trail of documents together in future machines [45]. The idea not too distant from the way the web works today. The World Wide Web Consortium [17] now maintains and evolves the specification. Grammatically, the rules are the same as in SGML, which HTML conforms to.

Like SGML, HTML by itself cannot represent special characters or formulas with origin in mathematics and chemistry related annotations. However, paired up with microformats and MathML, which is further explained below, it gains that possibility. In this way, HTML can display the equivalent of an office document format.

The design of the first HTML specification are however not completely overlapping with SGML [59], but as Tim Berners-Lee writes, it was compliant with its ideals[42]:

It is required that HTML be a common language between all platforms. This implies no device-specific markup, or anything which requires control over fonts or colors, for example. This is in keeping with the SGML ideal.

In the HTML 4.01 specification[67], we can read that:

HTML 4 is an SGML application conforming to International Standard ISO 8879 – Standard Generalized Markup Language [ISO8879], and further that they define an HTML document to be an SGML document that meets the constraints of the HTML4.01 specification.

The story about HTML is somewhat different from OOXML and ODF. It was an outcome of the creation of the web, and not a complete tailored language for a long existing need. HTML is the publishing language of the web, and is perhaps the open format that has the fastest growth among any publishing language, growing as much as one billion pages per day [21].

A great difference from the OOXML and ODF formats, is that HTML does not include presentational markup language, it was intended for structural and semantic markup [40]. However, if paired with CSS, it gains presentational and formatting qualities. With CSS, HTML is an elaborate enough language to even publish papers online, as Håkon Wium Lie did with his PhD paper[59].

HTML is the basic common denominator of web-languages, and covers the basic needs of web development, such as text and tables. For presentations and formatting, HTML pairs up with CSS. If additional structures are needed to represent a document, supplementary markup is needed, and there are several created with different purposes. MathML includes math-

ematical expressions in websites, which have the benefit of being recommended and developed in W3C, ensuring no conflicts arise between HTML and MathML. Microformats can be used to embed the the hidden information in office documents such as formulas to websites. More about these additions to HTML below.

This format enables rules beyond the basic rules covered in the chapter above. HTML has a defined tag-set, and there are some tags that needs to be in the document in order for a browser to read the page correctly. These rules are defined by DTDs specific for HTML, defining the legal building blocks of the document. The most common DTDs for HTML are the strict and transitional ones. An example of the minimal tag-set you can use to construct a valid HTML document is shown below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Title of website</title>
<body>
Contents
</body>
</html>
```

Attributes in elements are used extensively in HTML, among the uses are setting language attributes and character set of the document, and accomodate forms. The correct syntax for the use of attributes in HTML and other XML formats is shown below.

```
<html lang="en"/>
```

These are the basic rules of HTML that this thesis will concern itself with. There will be paid little attention to the visual qualities of HTML paired up with CSS.

Microformats A possible way to add structured semantic information to a website, is to use the already existing attributes and elements that are native to HTML, with a new purpose. The idea is to use attributes like the "class" attribute to achieve this. The data used in this manner provides a standardized way of publishing or consuming data [22].

The definition of a microformat according to microformats.org is[22]:

Designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards. Instead of throwing away what works

today, microformats intend to solve simpler problems first by adapting to current behaviors and usage patterns (e.g. XHTML, blogging).

It is used today for formatting user specific information. One of the most widely used microformat is hCard, for marking up contact information for people or organizations. Among other sites, the photo sharing site called Flickr uses hCard to mark up profile details on its profile pages[40]. This microformat is elaborated below to clarify what a microformat is, and how it can be applied to the thesis.

The hCard format is based on a standard called vCard, and uses a 1:1 representation of it, so in the example below of how to embed hCard to HTML, the class attribute name is vCard.

```
<div class="vcard">  
<span class="fn">Ingunn Rønningen</span>  
</div>
```

The equivalent vCard would be:

```
BEGIN:VCARD  
VERSION:3.0  
FN:Ingunn Rønningen  
END:VCARD
```

The *span* element is used because it is likely to be an in-line element and not a paragraph or block element like the hCard itself is probable to be, hence the *div* element is used for that purpose. The attribute of the span class called *fn* is an abbreviation for *formatted name* and explains how the name should appear[40].

This structure may be used for adding information from office documents to a website, that is not supported by an already existing element, or by a possible union with MathML. Consider the formula in a spreadsheet to be a matching example of application of microformats. Imagine using the newly developed value class pattern in the vCard standard, in a little different way achieve this [23]. An example of this pattern in the hCard fragment would be:

```
<span class="tel">  
<span class="type">Home</span>:  
<span class="value">+1.415.555.1212</span>  
</span>
```

With the vCard equivalent: TEL;TYPE=HOME:+1.415.555.1212

In a similar way, we can picture a markup like below for a formula B1+B2:

```

<span class="cell3">
<span class="type">Formula</span>:
<span class="value">B1+B2</span>
</span>

```

To clarify, there is no desire to use the hCard for this purpose, but instead create a similar new standardized microformat with the solemn purpose of informing websites about office document elements that does not have an equivalent element mapping.

MathML is a released recommendation from W3C dating back to 2001. It is developed by a W3C Math working group, giving it the benefit of complementing HTML rather than conflicting. Alas, not all browsers support this yet, but a large amount of them do, and you have the possibility to configure a browser to display MathML[24].

$$A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

Figure 9: Example matrix.

Like the other focused formats of the thesis, MathML divides and presentation elements, and the specification can be found at W3C's website [24].

A simple example of the MathML markup of a matrix shown in figure 9, which will have markup as shown in figure 10.

```

<mrow>
  <mi>A</mi>
  <mo>=</mo>
  <mfenced open="[" close="]">
    <mtable>
      <mtr>
        <td><mi>x</mi></td>
        <td><mi>y</mi></td>
      </mtr>
      <mtr>
        <td><mi>z</mi></td>
        <td><mi>w</mi></td>
      </mtr>
    </mtable>
  </mfenced>
</mrow>

```

As we can see, the grammatical rules are followed, and while the elements will not be detailed here, we can see that they follow a logical pattern. To pair this markup language with HTML, we would be able to show the unique mathematical expressions from office documents on websites using the conversion system.

Figure 10: Matrix MathML markup.

3.4.3 ODF

The Open Document Format is created in a collaboration consisting of a set of organizations concerning standardization. Sun Microsystems originally devel-

oped the format, while OASIS developed the open standard. The most known implementer of this open standard is OpenOffice.org who created the XML format on which ODF is based on [25].

This format has been in the making since 1999, and became an accepted ISO standard in 2006 [26]. The format can be used to represent text, spreadsheet, charts and graphical elements [44].

The ODF file is not a standalone file, but instead a packages collection of files, where subdocuments for contents, appearance, style, meta-data and settings are found, all with defined criteria for root elements and belonging tag-set. The file of concern to this thesis is the *contents.xml* file, which describes the contents of the document.

Since it is a format based on XML, it has the same structural integrity as that format. And like HTML, it has a defined tag-set. The specification is published to the public in PDF, and is a 738 pages long document. The smallest tag-set for an empty text document differs greatly from HTML. It has a lot more namespace bindings in the start of the document, declaring the use of a family of reserved attributes and elements. An example of the ODF syntax is found in figure 11, illustrated by the body element, with the text “ An example text” as its contents.

```
<office:body>
  <office:text>
    <text:sequence-decls>
      <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
      <text:sequence-secl text:display-outline-level="0" text:name="Drawing"/>
    </text:sequence-decls>
    <text:p text:style-name="Standard">
      An example text
    </text:p>
  </office:text>
</office:body>
```

Figure 11: Example markup of the content file of ODF.

As we can see, there is a set of default tags in order for the markup to be valid. The element names do share similarities with the tag-set of HTML, or other XML based formats. The *office:* text in front of the remaining tag name, is a prefix to the element name, and is a part of the set of reserved namespaces in ODF. The *office* prefix describes common pieces of information that are not contained in another, more specific namespace [48]. The second prefix used in the example is *text*, which is described as the used prefix for elements and attributes that may occur within text documents and text parts of other document types, such as the contents of a spreadsheet cell [48]. This format also has a defined structure of a document, where the root element in the content.xml file, has to be *<office:document-content>*.

As previously mentioned, this format is getting increasingly recognized worldwide as the answer to promoting equal and free access to critical documents published by governments. An alliance has formed with the mission to work on a global scale to contribute to the implementation of this standard as a legal alternative in all public services[61].

To ensure ease of use, Openoffice.org has created a validator for this open standard, which can be found on their website [27].

3.4.4 OOXML

Office Open XML (OOXML) is an open format used to represent text, spreadsheets, presentations and chart documents. This format was developed by Microsoft, who delivered the specification over to Ecma in order to be developed as an open standard. It was submitted for approval at ISO, and became a voted ISO standard in 2008. The most known implementer of this open standard, is Microsoft, in their Microsoft Office application [16].

The format was meant as a successor to the Office 2003 XML file format. Backwards compatibility was not included in the new format, causing some difficulties especially since the Office 2007 package had a free trial, leaving you with no other option then to buy the Office 2007 package if you wanted to be able to open your files written prior to trial expiration [37]. All difficulties aside, OOXML is one of the most used open standards at present day, due to its related application and operating systems vast success.

Based in the XML format, OOXML share the same principles as ODF, with namespaces and defined tag-sets for various defined purposes. However, the namespaces and tagsets diverge in naming and use. The specification is 6045 pages (not counting the example files), and has a different packaging structure[49]. The specification used in this thesis is the ECMA specification for the standard from 2006, since the ISO 29500 open standard for Office Open XML File Formats, is only available at the ISO store for 352 dollars.

Another difference is that OOXML has a set of markup languages for different purposes. The primary markup languages are:

- WordProcessingML for word-processing
- SpreadsheetML for spreadsheets
- PresentationML for presentations

These markup languages share some language materials like markup for math and drawing. The main components of OOXML are illustrated in figure 12[28].

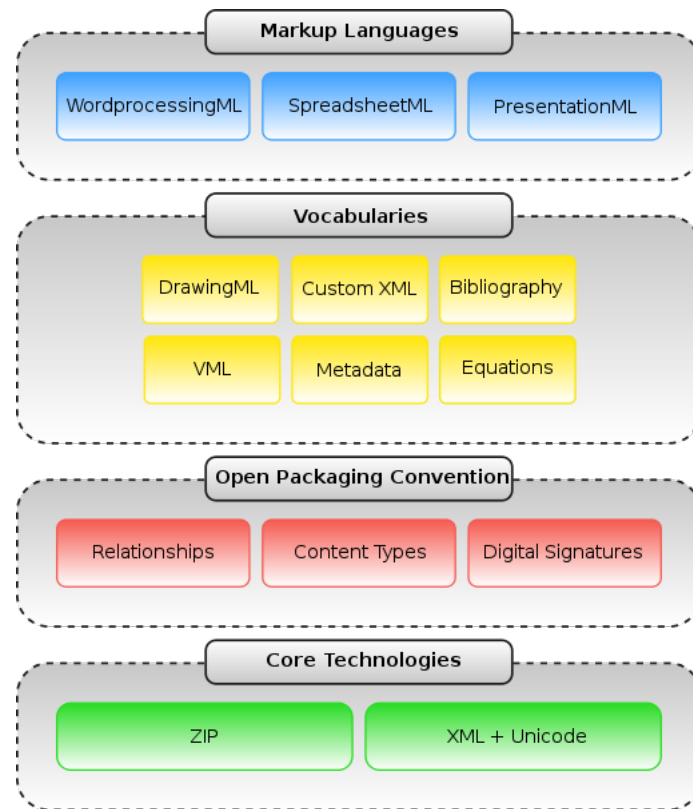


Figure 12: Main components for Office Open XML.

In figure 12, the complexity of OOXML is outlined, with a family of markup languages and ability to customize your XML. The open packaging convention sets the relations between the files, content types and declares digital signatures. The core technologies used, as in ODF, is zip and XML+Unicode.

Like ODF, the OOXML file is not a single standalone file, it also has a set of sub documents describing settings, meta-data, contents, and styles. The thesis will concern itself primarily with WordProcessingML, observing that this markup language is used for word-processing. The file of concern to the thesis is the content file called *document.xml*. This file is claimed to define the minimum content for the Main Document part[49](p31). The body element of the document only containing the text "An example text" is shown in figure 13.

As we can see there are some meta-data in this file as well, setting the language attribute, and a prefix *w* instead of the text or office prefix we saw in ODF, due to the use of WordProcessingML. The element names are not as simple to understand as ODF's element names at first glance. The *p* tells of a paragraph, while the *t* element tag is used for text. The *rPr* and *pPr* elements are style related for the document text [49](p667), such as font and heading type. The various attributes starting with *rsid* are session specific,

```

<w:body>
  <w:p w:rsidR="00577D1E" w:rsidRPr="00BB7D9B" w:rsidRDefault="00BB7D9B">
    <w:pPr>
      <w:rPr>
        <w:lang w:val="nb-NO"/>
      </w:rPr>
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:lang w:val="nb-NO"/>
      </w:rPr>
      <w:t>
        An example text
      </w:t>
    </w:r>
  </w:p>
  <w:sectPr w:rsidR="00577D1E" w:rsidRPR="00BB7D9B" w:rsidSect="00577D1E">
    <w:pgSz w:w="12240" w:h="15840"/>
    <w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="708" w:footer="708" w:gutter="0"/>
    <w:cols w:space="708"/>
    <w:docGrid w:linePitch="360"/>
  </w:sectPr>
</w:body>

```

Figure 13: Example markup of the content file of OOXML.

the specification[49] states that:

All *rsid** attributes shall indicate that those regions were modified during the same editing session.

Keeping in mind the automatic backups and saving possibilities in office document applications, we find that this is the way OOXML solves this issue.

There are also some elements and attributes giving some page formatting like size, margins, headers and footers representing the default values of these elements in a docx element. Hence, OOXML does not completely separate presentation and contents. The *sectPr* element contains the final section properties. All other section properties is usually is a child element of its section, but the final one has to be a child element of the body, in order to have a default property for the document [49](p654).

3.4.5 Commonalities

This part looks into how the standards above relate and what commonalities there are to build from.

First of all, it is important to remember that even though all of the focused formats are based in XML, they differ in their purpose. XML was developed to transfer and store data, while the HTML, ODF and OOXML formats were first and foremost developed to display data. This is a rudimentary technical relation between the three formats.

Interaction The set of interaction activities are also similar regarding the three open standards. They are all used in a setting involving a computer and a person. Regarding office documents and documents on the World Wide Web, the set of activities being performed are usually sharing documents, reading, saving and writing or editing a document.

Construction The three formats share some common rules of construction, like a defined way to start and end a document, and a defined set of tags to use for word processing with a distinct behavior. An example of this is the body tag for defining where the contents of a document is found. In HTML this is `<body>`, while in OOXML the same distinct behavior is found in the `<w:body>` element, and in ODF the equal element name is `<office:body>`, only differing in the prefix.

The basic syntax rules are the same for all three formats. Inherited from SGML, they all have proper nesting of elements, and construction of matching start tags and end tags. The syntax of elements attributes are also handled the same way. All three also have a document type definition(DTD) to notify the rendering application of how to read the document, and thus how to use the defined tag-set connected to the format. The connection to SGML is stronger with HTML, where it in 1992 was formally specified an SGML DTD [68], while ODF and OOXML doesn't have a straight connection but rather through XML.

Formatting of contents is handled in a similar way in all three formats, seeing that it is largely found in a separate file. ODF and OOXML have defined placement for this formatting in the file structure. HTML is usually formatted with the use of Cascading Style Sheets (CSS). Although there are some exceptions regarding OOXMLs default style values, and HTMLs opportunity to use the inline style attribute.

Overlaps The two open standards that are overlapping in a substantial manner is OOXML and ODF, recognizing that HTML has a different position as the publishing language of the web. Logically, there are three feasible outcomes for the overlapping standards in the future. One of the open standards becomes the open standard with the most users, and takes over the market for office documents. The second feasible outcome is that the formats merge. This will be probably be the most cumbersome and complex outcome. The third outcome is that a new format takes over the market for office documents entirely, and obsoletes both OOXML and ODF. In either of these three cases, access to the documents made today are still vital, in addition to historical reasons.

3.5 Affiliated terms

As the terms free and open are increasingly used in the IT-industry, there is some confusion around these terms. Regardless of this confusion, the terms are frequently used, perhaps due to their positive and trendy associations. Under any circumstance, the definitions of the terms vary in the IT-industry. In this section we look into the terms often used in relation to open standards, namely open and free formats, open source and free software and how open standards are situated among them.

3.5.1 Open and free format

As briefly explained earlier in 3.3.1, the difference between a format and an open format, is that you are free to use and implement the open format, which may not be the case with closed formats. The difference between an open format and a free format has about the same distinction as the open source software and free software terms, which will be elaborated further below. An open format is commonly used to mean free format. However, they originated differently, and have different definitions.

The free formats are made freely available for anyone to read and study but for which there may be restrictions on its use[4]. The Linux Information Project (LINFO) defines the free format as[4]:

A free file format is a file format that is both (1) published so that anyone can read and study it in its entirety and (2) not encumbered by any copyrights, patents, trademarks or other restrictions so that anyone may use it at no monetary cost for any desired purpose. Such specifications are usually maintained by a non-commercial standards organization.

This definition is the most purposeful regarding this thesis, where a file format is defined as a specific way of encoding digital data to create a file. An open standard thus have to be an open format, but not the opposite. The open format does not have to be market driven, or used by a large group of people, while the open standard requires those abilities. If we only look at the technical aspects of these terms, an open format and an open standard are close to identical. All three of the focused formats of this thesis are free file formats.

3.5.2 Open source

Open source, is a term that defines a process. This process is ongoing from the birth of a program, to the implementation of it.

The Open Source process encourages developers to develop systems in collaborations. The source code is readable, editable, downloadable and is often linked with a online community in which to discuss and deliberate whilst developing.

The Open Source Initiative (OSI) claim to be the stewards of the Open Source Definition, and have made an annotated list of ten criteria which you must comply with in order to call your software open source software. These can be found in their website [2], and in short explains the regulations that may apply after the program is finished regarding derived work and licencing, and rules to eliminate discrimination during the development period.

Open standards fit into the Open Source process. If Open Source Software use open standards, it will theoretically ensure that when the software is used, the integration to the users life will be cheaper then with a proprietary format. The open source initiative doesn't try to make an own definition for an open standard, but instead simply states that if you can not implement an open standard under an open source license, it is not open enough for them [2]. This means that Open Source Software has to use open standards fitting the working definition of an open standard of this thesis.

In order to call your software Free Software however, there are more restrictions, more about this in the following section.

3.5.3 Free software

When you first hear the term free software, it is common to assume that it is free of cost. This is not necessarily the case, but instead it is software that has certain criteria fulfilled. Richard Stallman explains that there are four criteria for software to be free[65]. These four being :

- Use - The ability to use the software freely
- Study - Study the source code
- Distribute - Share a copy non-commercially
- Contribute - Modify the source code at own leisure and share it

To ensure free software is not mistaken for anything else, licences are usually utilized. The most common is perhaps the GNU General Public License, covering all the four criteria above and also making sure that any derived software will be published under the same license, this addition to the four criteria is called Copyleft. This way it will remain free software after the software has been modified.

Open standards can without conflict be used in free software, as long as the licences are compatible.

3.5.4 Open Source and business

Attending the Goopen conference both in 2008 [37] and 2009 [38], it was clear that the use of open standards and open source programs are being extended. This seems to be the case for free formats and free software as well, considering governments increasingly pick open standards to be their way of communicating with the public.

In correlation of the growth in interest, new companies based on open source software has spurred, and existing companies are converging to using open source or free software and open standards in their business model. There is commonly a difference in licencing between commercial use, and private use. Where commercial interests have more restrictions then private users.

Freedom of choice with open source and free software might be seen to conflict with standards. One may be inclined to think that with the commonalities that enable the communication of documents dissapears, and it may be the case when we see the situation with how OOXML and ODF interact as exemplified in 1.2. However, this is not necessarily the case, where a standard does not decide what application to be put in, but rather the other way around. Different applications does not automatically mean different standards.

It is worth noting that since the code is free to examine and copy, most of the profitable business models surrounding open source and free software focus on support and maintenance[38].

4 Case

The case in this thesis is the construction of a system that will perform conversions between overlapping formats. The focus is to enable the conversions between ODT, HTML and DOCX.

This section will contain information about how the development process progressed. As mentioned earlier, the method applied to the case was an iterative method. First we will take a look at the initial planning for the system. Following this description of the initial planning is the remaining steps of the iterative development cycle. The code for the system can be found in Appendix C.

This development process had three iterations. The first two iterations were fairly short, with the first was quicker than the second. However, they were by far the most educational. In these two cycles, the concern was to understand what needed to be done, and how to do it efficiently and properly. After learning from these first iterations, uncovering flaws and probable improvements, the third and most embrative of iterations was engaged. Although the development steps in the last cycle were not followed firmly, jumping back and forth between the development stages as called for by the progression of the project. In the sections below, there is a short description of the first two iteration, followed by what happened in the third iteration.

4.1 Initial planning

The grounding for the development of the system was laid in the initial planning. This initial planning is described in this section, and is in two parts. The first part named research tells about the theoretical knowledge acquired to enable the development process. Secondly, the technologies used in the project are explained.

4.1.1 Research

Initially, the idea was to create a system that treats conversions between the formats to work somewhat like a dictionary. In dictionaries, one word in one language corresponds to one word in another language, but share the same meaning. This *mapping* between the two words create a *rule* of conversion between the two languages. These same kind of mappings exist between OOXML, ODF and HTML, where one element in one format will correspond to an element in another format, and share the same meaning. For example the body tag in all three formats share the same meaning; here is the contents of the document, but the element name differs slightly.

This way, the converter would be a rule-based system, where adding and removing rules as specifications change is easy, making it a dynamic system to handle changes that surely will come. To make this happen, a web-interface had to be made, to input these rules to some structure that would contain them.

As the paragraphs below will show, to understand the basics of the formats, and to wrap my head around how to construct the framework in a manner corresponding to what explained above, I started looking into the specifications and structure of the formats, and what it would mean to be a structured document. After that, a closer look at how to address HTMLs shortcomings was done, resulting in microformats and MathML being a promising candidates. Finally, with some basic knowledge of what concerns the framework, other existing converters were analyzed, in order to understand how they work.

4.1.2 The specifications

The HTML 4.01 specification was found at W3C's website [67]. It was easy to navigate in, and easy to find. The specification was written in HTML, but available in Portable Document Format (PDF).

The specification for the Open Document Format was fairly easy to locate from the OASIS website, and was available in the OpenOffice.org format, in addition to PDF[48].

The OOXML specification however was trickier to locate. ISOs publication of OOXML named ISO/IEC DIS 29500 cost money, and with student finances, that was not feasible. The ECMA publication of the OOXML standard from 2006 was located and utilized instead [49]. The second edition publication from 2008 was not available in the start of writing of the thesis, but is available through Ecma [49]. The differences between the two editions did not have much influence on the system created in this thesis[50]. This specification was the largest, and perhaps the most confusing at the first glimpse. The specification contains 5 zipped separated parts, who all hold different information about the standard. The written specification was in PDF, while the attached files with information Schema, Style and definition documents were in various file formats.

4.1.3 Structured documents

All of the formats to be used in the conversions are structured documents. Håkon Wium Lie defines structured documents in his PhD paper as:

A digital document consisting of hierarchical elements containing text and other content. The elements primarily represent the logical roles of the content rather than the presentation of the content[59].

Further, he states that HTML and XML are seminal structure systems [59], and explains that:

Style sheet languages and structured document formats are mutually dependent on each other. Without style sheets, structured documents cannot be presented, and without structured documents there is nothing for style sheets to present.

Judging by this, OOXML and ODF are also partly structured systems according to the definition above. They differ from a structured document where presentational qualities are also a part of the document structure. However, the specifications for both OOXML and ODF separates presentation and contents to a great extent, where they have one file in the structure dedicated to contents in the hierarchy of documents that create the file in the destined format [49] [48].

The fact that HTML is not sufficient on its own to display all the information that may be in the two document formats, had to be addressed. Mathematical formulas and other special signs and graphics from chemistry are indeed displayed on the web, due to browsers supporting markup such as MathML[24]. A natural course of action concerning this issue, is therefore to look into MathML, and see if it in union with HTML is sufficient to create the rules needed to create a full mapping between the two office document formats. In supplement or perhaps replacing MathML, the use of microformats is considered, to aid HTML in representing the same information as the office document formats.

4.1.4 Existing converters

To get acquainted with the workings of existing systems that perform conversions between formats, a set of existing converters were tested and analyzed. The majority of the converters studied were open source software, or free software.

Some were located through a search on the web, but also in websites like sourceforge.net, that among other open source projects, supports projects concerned with converting one format to another format.

There was a vast variation in quality of the converters found online. As expected, a moderate amount of converters were found that claimed to convert between ODF and OOXML. Apart from Microsofts published project

description from 2008, there were not many converters between OOXML and HTML. A couple of converters were found between ODF and HTML. However, none of the located converters could convert successfully with both formatting and content intact, between all three focused formats.

A choice was to be made whether to try to merge some existing converters, or follow the initial idea of the mapping and rule-based framework described above. Since most of the converters were written in different programming languages, and offered various solutions to converting, although all through XML in some form, the structure differences would not make an integration seamless. The tipping weight of abandoning the merging approach, was that the research in a larger sense would occur when trying to make two or three existing systems interoperable, and not while developing and learning about the three focused formats.

Although the existing converters were abandoned, some knowledge was extracted from the projects that were analyzed:

- The step through XML seemed to be an essential step in converting the formats.
- Developing in communities like Sourceforge.net or code.google.com, gives you advantages where other people can help you in the development process.
- It's hard to locate working converters, and understanding how to operate them.
- This converter seemed a larger task then originally perceived.

4.2 Developer Environment

The tools and technologies used in the project are highlighted in this section. The requirements for these tools and technologies, was that they were free of charge, and that they were free to use and implement as seen fit. They also had to fit with the operating system available to the developer, namely Ubuntu.

4.2.1 Python

Python was chosen as the programming language for the converter.

The abilities of this scripting language combined with a very intuitive structure, allows easy and swift progression with writing code. It was first introduced through a course in the University of Oslo [29]. The many flexible traits of the language was learnt from this course. In addition, Python is

a great language for processing XML, with a great variance of dedicated libraries [57].

4.2.2 Apache2 HTTP server

To run the web interface, a web server was needed. The choice to use apache was easy, its a well tested, free, and a well documented web server[30].

4.2.3 PHP5 vs mod_python

To enable Python to work with apache2, an extension needs to be installed. This extension is called mod_python. PHP however, will work with apache2 out of the box, leaving the interface programming to be dealt with straight away. However, wanting to keep the system in one language, an effort would be done to get Python working with apache2.

4.2.4 MySQL

A database was chosen over a file-system to retrieve and store the mapping information between two formats. The system uses a relational database, with SQL as a query language. The benefits being a quick lookup and reliable storage, following the ACID properties of data transaction[53].

4.2.5 phpMyAdmin

To ease the handling of the MYSQL database, a tool called phpMyAdmin was utilized[31]. This tool allows an overview of your database, and eases manual transactions to the database, otherwise this would have been done with the MySQL command line alternative.

4.2.6 Dropbox

To keep a backup and to have version handling of both the system and the thesis, an online service called Dropbox was utilized.

With this service, you can link several machines together to write and read from the same repository kept by Dropbox. This repository keeps deleted files, and revisions of your files. If you are on a machine without Dropbox installed, you can still access your files with a password and username on their website[32].

Furthermore, it provides a simple online interface to interact with your repository, and a lucid overview of recent changes.

4.3 Planning and setting requirements

The mantra consistently chanted while setting the requirements and further planning, was to keep it simple.

The overall aim is defined as “Creating a system that eases conversions between OOXML, ODF and HTML”. The loose requirements set prior to the two first cycles were:

- Make an interface on the web that will input the rules that makes the mappings between three formats
- Make the framework in python that converts between the formats.

The last cycle lasted around three months, where composing the structure of the framework was the most difficult task to finalize, hence the most time consuming. This event was speculated and applied in the timeplan, although time was hard to estimate considering the many uncertainties of the project. The timeplan was a rough plan, shown below:

- 1 week - Get familiar with context and tools, start coding a little.
- 3 weeks - Set up environment, get familiar with it and code a bit more.
- 3 months - Programming and testing in center. Analysis, planning and designing as seen fit.

The online part of the system was decided to be developed top-down, and very basic, since the focus would be on developing the framework. A bottom-up approach was chosen for developing the framework, seeing that all the major decisions would have to be made there, and would lead to handle the predicted changes with less overhead.

The time spent on the first two cycles were in the vicinity of four weeks, investigating and uncovering a wider perspective of what the system should contain and how its structure should be. And perhaps most importantly, get a better feel for what sort of challenges working with the formats would bring, as well as potential issues that required addressing.

Although these requirements were still valid despite of their vague nature, the last cycle, added on a more detailed set of requirements, both technical and functional. At this point, the scope of the project was also narrowed down, seeing that the whole specifications would be too much to envelop in this project. Initially, the new framework would only deal with converting text, and attributes are not considered.

There are three terms used from this point on; *framework*, *website* and *system*. The *website* signifies the online website, while the *framework* signifies where the handling of the conversions occur. The *system* addresses both the framework and website.

4.3.1 Technical and functional requirements

Functional requirements of the system was identified, defining the specific behavior of the system. These are listed below starting with the website:

- The website where you can input rules is to be written in PHP.
- The website will insert mappings to the database.
- The website should communicate an input file and retrieve a converted output file from the framework.

The website is a part of the system to ease user interactions with the database and framework. The former when you enter rules of mapping, the latter when you want to convert a file. In addition to this, a website is included to the system due to the extensive reach of the web.

Further, we look at the technical requirements of the framework:

- The framework should convert text. Starting with the tags necessary to create a text document, along with HTMLs `<p>` and the other formats equivalent tags.
- The framework should be object oriented, using python classes.
- The framework will convert the formats through XML.
- The framework will build the new file XML with XML and string manipulation.
- The XML library used for parsing in the framework is the ElementTree library.
- In the framework, separate the start and end tag elements from the rest of the contents, due to very different ways of starting and ending the content document.
- Have the framework that performs the conversions work with command line first.
- Expand the development environment to include temporary structures for unzipping, testing and filehandling.
- In the command line, let the user know what is happening, and where to find the new converted file.

- The framework should have a MVC structure, separating view, control and model.

The ElementTree library was chosen due to word of mouth of its ability to process XML. It also has a check for grammatical errors when building the tree. So files that normally wouldn't compile with its normal file structure, would not even be considered for conversions.

4.3.2 Non-Functional requirements

The non-functional requirements of the system were also identified, specific to the operations of the system, and are listed below:

- The responsetime for the system should be quick.
- The system should be stable and not crash. This should be addressed through exception handling and error checking.
- The data should be secure. The database is protected by a password and username, and security measures should be taken on the website when public.
- The system should be easy to use.
- The system should be easy to find.
- The system should be platform independent.
- There should be tests for every piece of code.
- Configuration files to separate the necessary hardcoded information from dynamic code.

4.4 Analyzing and designing

This section will describe the analysis and design choices made during the lifespan of the development process.

4.4.1 The website

The contents of the website was designed to meet the basic requirements set for it. You are supposed to be able to enter a rule of conversion that leads to a stored mapping between two formats in the database, and you are supposed to be able to enter a file, to get your converted file back.

The design of the website was very basic, and only changed to include dependencies after the second cycle. With uncertainties regarding its final appearance, it was considered best to keep this part to the bare necessities until a more finite structure was in place. Its simplistic appearance is shown in figure 14, as produced in the last cycle.

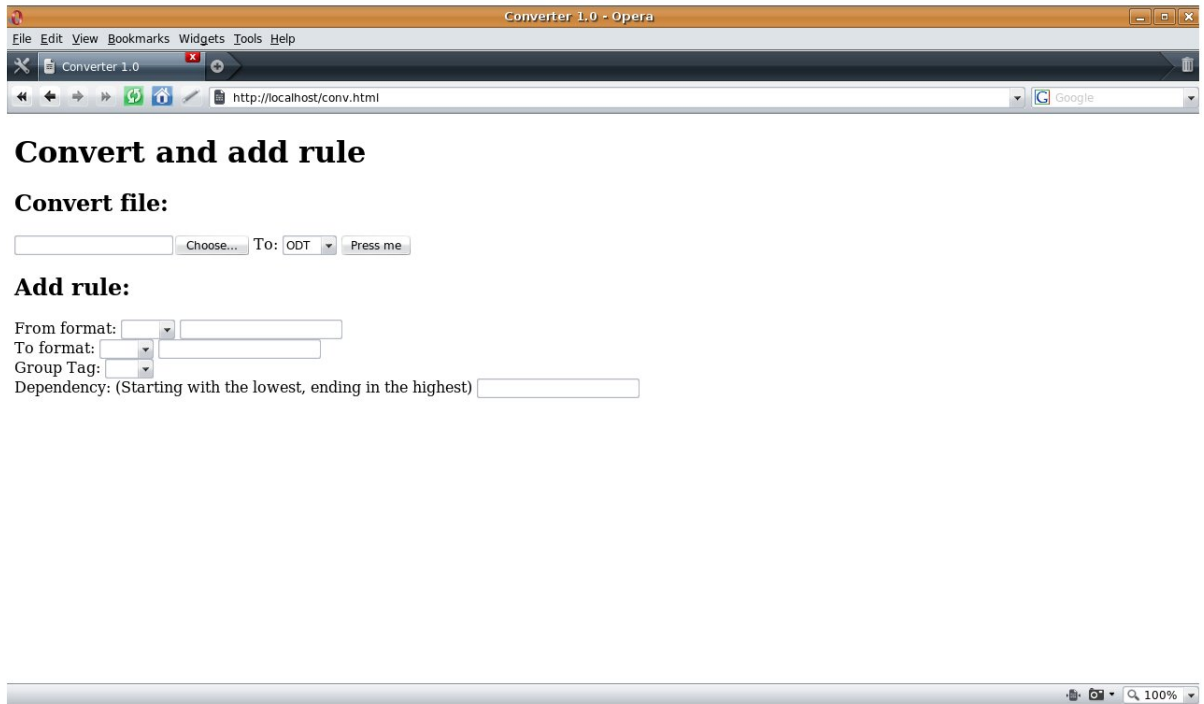


Figure 14: The converters website

As we can see, there are no CSS applied to the website, but merely the form options that are necessary to execute the desired operations on the database. It also has some descriptive headings, a title, and a set DOCTYPE.

4.4.2 The framework

The first design sketch of the systems flow was made during the first cycle, using pen and paper. This was a low detail and low resolution sketch. It is shown in figure 15.

The structure of the framework was roughly guessed on basis of the information acquired during the initial planning, and the requirements originated from the prior development step. The MVC structure was outlined, and some of the building blocks of the system were defined.

The sketch turned more sophisticated as it embeds the lessons from prior cycles. The last representation of the systems flow was made late in the

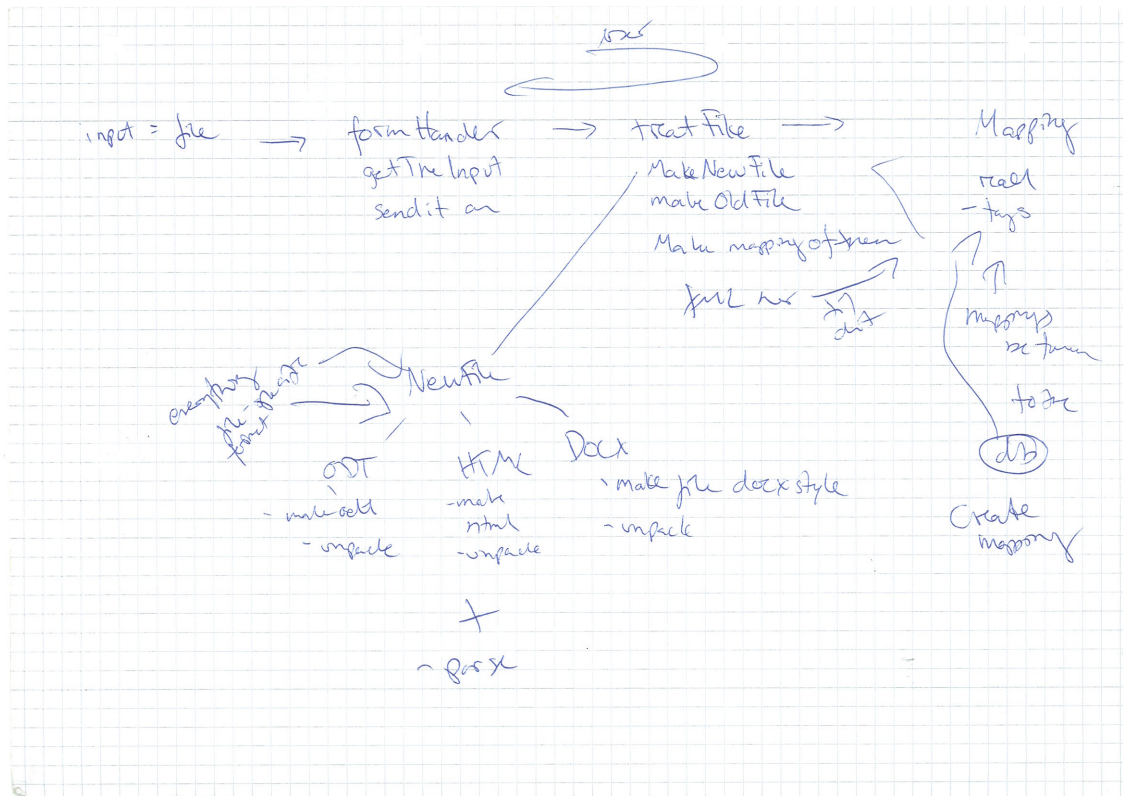


Figure 15: The first sketch of the system.

last development cycle. This representation is shown in figure 16. This is far more detailed, showing the actual flow of the system, and the building blocks of it. It includes some comments on special cases and distinct circumstances to acknowledge in the development process.

The framework separates the view, control and model. In the view layer it presents the user with a way to interact with the system, in the model layer it handles the input from the view layer, and sends it to the controller. The controller processes the input, and responds to them.

In the flow of the system, the treatfile, NewFile with subclasses, and Mapping classes form the controller, where all the processing of data is performed. The formhandler is the model, and the view is represented by the website and the command line.

Each format has its own subclass, containing only the attributes that separates the formats, sharing the equivalent information in the parent class; NewFile. The Mapping class contains the controllers communication with the database, and performs the conversions. Further details of these building blocks will be found in section 4.5.

Some extra attention was given to the tags that did not map 1:1, but rather

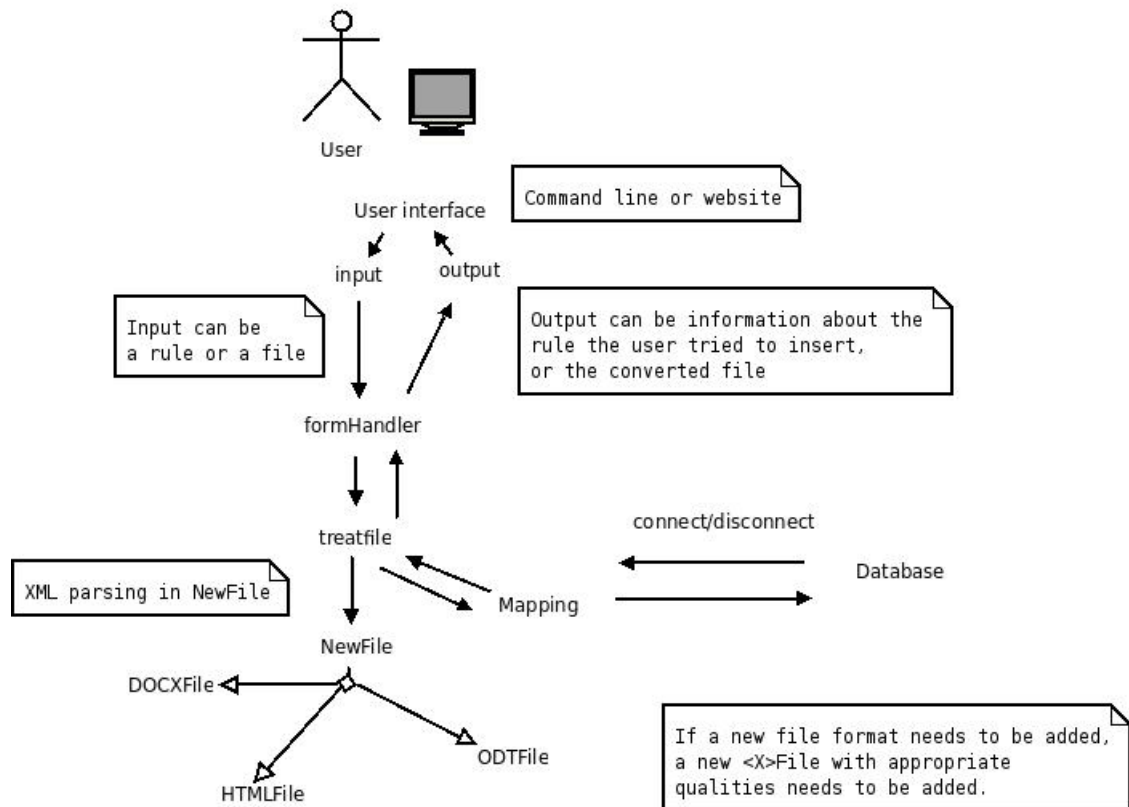


Figure 16: The last sketch of the system.

1:many. For example the text element in HTML can be shown as `<p>` while in ODT the equivalent to this HTML element are two elements; namely `<office:text>` `<text:p>`. The solution for this is explained in detail in section 4.5.

After analyzing the structure of the formats, a more detailed design on how to convert between the formats was formed. An illustration is shown in figure 17.

As figure 17 shows, the files with the actual text content was picked out of all three formats, to be used in the conversion. Consequently, when converting to a format, with text only, it would be sufficient to enter the new content file in the suitable place for the formats structure, and if needed, zip it back up, creating a converted file. Many data structures were considered to achieve this design, but the tree structure was preferred from an early stage. Primarily due to its advantages regarding parsing of XML, having the parent and child structure for the nested elements.

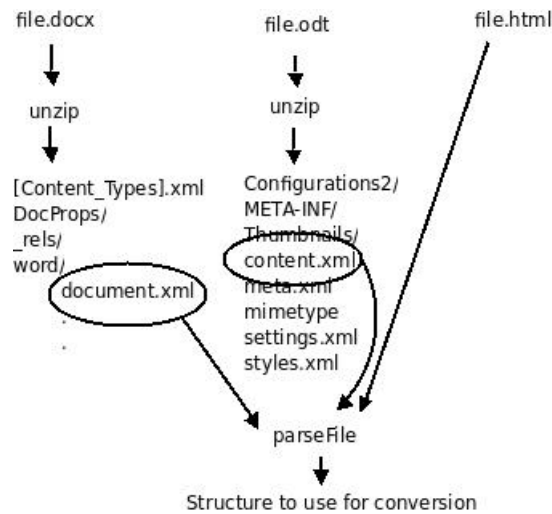


Figure 17: Parsing the content files in the three formats.

4.5 Developing, implementing and testing

This section will concern itself with the development, testing and implementing of the system.

The only implementation of the system is on the developers laptop, using the tools and technologies described earlier. The system has not been implemented on any other device, although it is theoretically feasible with the same surrounding environment, to have it perform as normal under any device.

Overall testing was carried out differently then originally intended. Manual testing was performed constantly throughout the development. Alas, no unit tests or functional tests were written. Some comments on this will be found in the discussion section of the paper.

We take a closer look at the development of two main building blocks of the system below.

4.5.1 Website

In the first cycles, there was a strong desire to facilitate the use of Python in this part of the system. However, this was not within the developers capability. After fighting with symbolic links, making the symbolic links executable, and mod_python for a much longer time then expected, it was discarded.

In order to adapt to this situation, falling back to programming this part in PHP was inescapable, and executed in the start of the last cycle.

The website is made of one HTML file, combined with a PHP script that handles the form posts and submits data to the database. The script lets the user know whether it was a success or not. It is very basic and simple code, wishing to keep the focus on this part to a minimum, while still having it work as intended.

Inserting rules with dependencies to the database from the website worked perfectly, albeit only on localhost, not prioritizing putting this online before it is ready for it. However, inserting a file through this interface was never finalized, due to prioritizing having the framework work with command line before the website.

4.5.2 Framework

This part will describe the development done in the framework.

View - Since the user interface on the web ended up being separated from the framework, the only user interaction with the framework came from the command line. However, the framework was intended to have an interface written in Python. The output was also directed to the command line.

Model - The transporter from the view to the controller in the framework is the formhandler. This module is programmed to make sure the user uses the system the way it is intended.

It also checks the users input to the framework. If the user gives faulty input, he or she gets information about how to use the framework from command line. In addition, it checks that the input file exists, and that you are not trying to use the conversion framework to convert between identical formats.

The intent of the formhandler did not change much during the cycles. Based on the website difficulties in the first cycles however, the range of input did change from being from the website and command line to being from command line only. This restriction is reflected in the output of the system, where it returns the location of the converted file.

The formhandler passes the valid information on to the controller part of the framework. This valid information is the location of a file, in a supported format. These formats being ODT, HTML, and DOCX. Specifically, the formhandler passes the information on to the controller's organizer, the treatfile module which will be described further below. The output from the treatfile module is caught by the formhandler, giving the location of the converted file to the user interface.

Controller - The structure of the controller changed substantially during the development cycles. However, the main outline remained the same. Following, the details of the modules in the controller is described in its final stage of the last cycle.

The development environment surrounding the main modules of the controller contained several folders for various uses:

- unzipped/ - A folder to unpack the incoming files, in order to extract the content data.
- testfiles/ - Contains testfiles with and without text, in all the focused formats.
- docxcontext/ - A folder containing all the information needed to construct a DOCX file, except its document.xml file, which hold the written text contents.
- odtcontext/ - A folder containing all the information needed to construct a ODT file, except its content.xml file, which hold the written text contents.
- outputFile/ - A Folder to keep the converted file in.

The database consists of two tables, detailed below in figure 18:

Rule	Dependency
+id: INT, AUTO_INC, PK	+id: INT, AUTO_INC, PK
+fromSpec: VARCHAR	+ruleid: INT, FK
+toSpec: VARCHAR	+thetags: VARCHAR
+fromSpecData: VARCHAR	
+toSpecData: VARCHAR	
+groupTag: VARCHAR	
+dependency: INT, FK	

Figure 18: Rule and Dependency tables.

In the Rule table, the size of the varchars of the data attributes were chosen fairly big, knowing that particularly the start tag of some formats could be very extensive. In the queries, the data was fetched considering both from-to and to-from relations seeing that this doesn't make a difference. To separate the start and end tags, the groupTag attribute is utilized. This is done simply by setting the groupTag as either "start" or "end". The dependency foreign key is the id in the Dependency table described next.

To meet with the issue of the 1:many mappings of elements, the Dependency table was created. The attribute `thetags` simply contains a comma separated collection of tags, that the tag in Rule needs in order to map completely with the other format.

The `treatfile` module was intended to keep the communication to the model layer, and also organize the conversion, binding the Mapping and NewFile modules together. This module creates the NewFile-objects needed for the conversion, and sends them to the Mapping module to have the conversion process performed. The `treatfile` module then cleans up the framework environment, by removing the content files from the respective surroundings, preparing for the next conversion to take place. It returns the location of the new converted file to the formhandler module.

The `NewFile` module imports the element tree library, done in a portable way to ensure it works across platforms. This class also parses the XML and sets the common variables of the formats such as its filename, what format it is, what it should be converted too, and its parsed XML

The focused formats are all subclasses of this class. In addition to the variables already stored in NewFile, the subclasses locates the content file of the format (unpacking if necessary), reads and stores it, then sends it to be parsed in the parent class.

The `Mapping` module holds all the conversion intelligence. The activities of the module is depicted in figure 19.

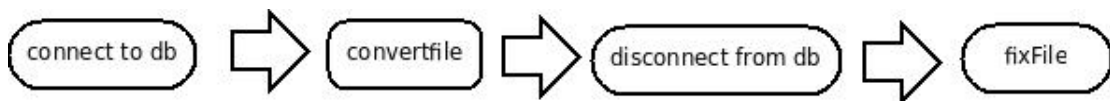


Figure 19: The activities of the Mapping module.

The superficial flow is quite simple, and easy to understand. The Mapping module contains the connection information to the database, and methods to arrange for both its connection and disconnect. Below, we go in further detail about the convert-file step, and the fix-file step.

Figure 20 shows the detailed `convertfile` step. Recalling the separation between the start and end of a document, this is reflected in the code. Both `getStart` and `getEnd` queries the database for the respective start and end `groupTag`. After some confusion regarding how the element-tree library iterated the XML tree, the `getMiddle` part calls a method named `makexml`

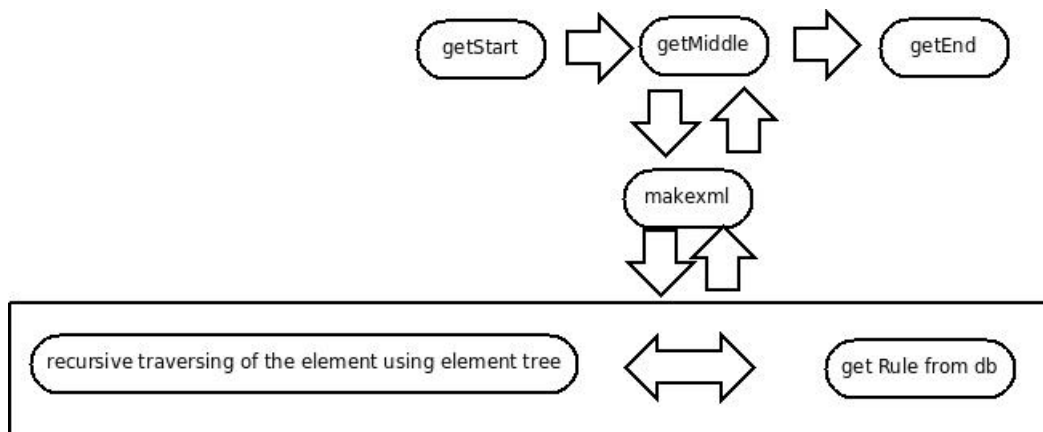


Figure 20: The convertfile step in detail.

with the root of the tree. The makexml method traverses the tree of elements recursively, making sure the nesting of elements and the contents is conserved while building the new files XML in a string. The contents where the mappings are 1:many are located using a recursive method, traversing the tree to identify the element that holds the text content.

After the XML of the new file is ready, it is sent to the fixfile method, where the XML is written to a destination decided by its format. ODT and DOCX needed to be zipped with its docxcontext or odtcontext folder files in order to compose a complete file. With HTML this was obviously unnecessary, and the file was directly written to the outputFile folder, where the treatfile module could locate it.

At the end of the time period dedicated to the last cycle, successful conversions were done between ODT and HTML, from HTML to DOCX, and from ODT to DOCX. The remaining conversion possibilities provided an empty file, due to a shortcoming in the code that will be elaborated in chapter 6. An example using the ODT to HTML conversion is shown below.

```
> ingunn@pinktop: /Dropbox/utvikling/control$ python formHandler.py testfiles/solskinn.odt html
> the file is: testfiles/solskinn.odt
>
> Processing the file
> unpacking
> Archive: testfiles/solskinn.odt
> extracting: unzipped/mimetype
> creating: unzipped/Configurations2/statusbar/
> inflating: unzipped/Configurations2/accelerator/current.xml
> creating: unzipped/Configurations2/floatar/
> creating: unzipped/Configurations2/popupmenu/
> creating: unzipped/Configurations2/progressbar/
```

```

> creating: unzipped/Configurations2/menubar/
> creating: unzipped/Configurations2/toolbar/
> creating: unzipped/Configurations2/images/Bitmaps/
> inflating: unzipped/content.xml
> inflating: unzipped/styles.xml
> extracting: unzipped/meta.xml
> inflating: unzipped/Thumbnails/thumbnail.png
> inflating: unzipped/settings.xml
> inflating: unzipped/META-INF/manifest.xml
> HTML is the target format
> In Mapping
> fromFile is : <ODTFile.ODTFile instance at 0x824fcec>
> newFile is : <HTMLFile.HTMLFile instance at 0x824fe2c>
> Connecting to DB
> Building file
> Killing connection to DB
> File is at outputFile/solskinn.html
> ingunn@pinktop: /Dropbox/utvikling/control$ more outputFile/solskinn.html
> <html><body><p>solskinn</p></body></html>

```

As we can see, the HTML document will not validate due to the lack of the head and title element, but nonetheless it will show the content of the ODT document in a lenient browser. The lack of the head and title element is due to this is defined using headings, and these are not considered yet.

Here is a successful conversion using 1:many mappings, from HTML to ODT, using an HTML file with the following markup:

```

<html>
<head>
<title></title>
</head>
<body>
<p>
solskinn
</p>
</body>
</html>

```

The conversion ran as shown below:

```

> ingunn@pinktop: /Dropbox/utvikling/control$ python formHandler.py testfiles/file.html odt
> the file is: testfiles/file.html
> ** * * * *
>

```

```
> Processing the file
> ODT is the target format
> In Mapping
> fromFile is : <HTMLFile.HTMLFile instance at 0x824fd0c>
> newFile is : <ODTFile.ODTFile instance at 0x824fe6c>
> Connecting to DB
> Building file
> Killing connection to DB
> adding: Configurations2/ (stored 0%)
> adding: Configurations2/toolbar/ (stored 0%)
> adding: Configurations2/progressbar/ (stored 0%)
> adding: Configurations2/popupmenu/ (stored 0%)
> adding: Configurations2/accelerator/ (stored 0%)
> adding: Configurations2/accelerator/current.xml (stored 0%)
> adding: Configurations2/statusbar/ (stored 0%)
> adding: Configurations2/menubar/ (stored 0%)
> adding: Configurations2/floater/ (stored 0%)
> adding: Configurations2/images/ (stored 0%)
> adding: Configurations2/images/Bitmaps/ (stored 0%)
> adding: META-INF/ (stored 0%)
> adding: META-INF/manifest.xml (deflated 83%)
> adding: Thumbnails/ (stored 0%)
> adding: Thumbnails/thumbnail.png (deflated 61%)
> adding: content.xml (deflated 72%)
> adding: meta.xml (deflated 59%)
> adding: mimetype (stored 0%)
> adding: settings.xml (deflated 84%)
> adding: styles.xml (deflated 80%)
> File is at outputFile/file.odt
> ingunn@pinktop: /Dropbox/utvikling/control$ ooffice outputFile/file.odt
```

The final command leading to the screenshot shown in figure 21.

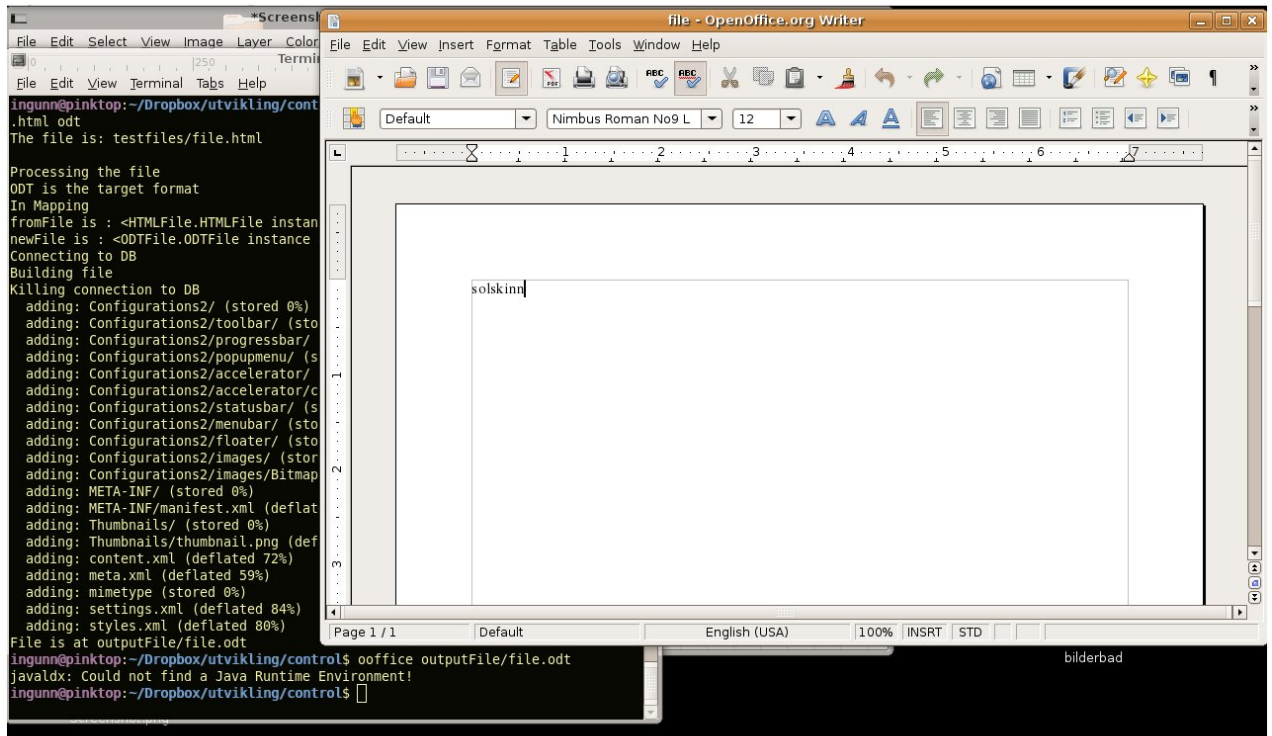


Figure 21: The ODT file with the HTML contents.

As we can see, the ODT file shows the HTML contents. This is because of the handling of the tag dependencies in the code. However, the algorithm was in need to be improved to deal with the shortcomings of some of the conversions. There was no time to make these alterations, but the origins of the problem was located. This is further explained in chapter 6 along with possible improvements to the system. Nonetheless, the working conversions were successful in showing that it is possible to use the framework to convert text content.

4.6 Evaluating

The evaluation at the end of the first two cycles was the most structured ones. The last cycles evaluation was done simultaneous with developing.

The main question when in the evaluation mode, was: *Will the system do what it is intended to do when designed and programmed this way?*. The main properties sought after in the system was efficiency, simplicity, and order.

There was a notion of a Rule module after the first cycle, and a class was constructed for the purpose of handling these. This was discarded and con-

sidered redundant after evaluating its necessity, and its code was embedded in the Mapping module in the second cycle. A few other dead ends were hit, but usually caught while evaluating based on the forementioned question.

The framework does successful conversions between:

- ODT -> HTML
- ODT -> DOCX
- HTML -> DOCX
- HTML -> ODT

The markup of the converted file is sufficient to show the textual contents. The two conversions that did not run successfully however were from DOCX to HTML and from DOCX to ODT. The conversions ran, but the resulting files had no contents. This will be further addressed in chapter 6. The framework does not completely fulfill the requirements set in the first development steps. However, even with these unsuccessful conversions, the framework shows great promise.

Evaluating the need to enter the file through the website was also considered here, resulting in downprioritizing this part. The system fulfill its main goal without it, which is convert between three formats, even though the file is only submitted through the command line.

While the answer to the question posed at the start of the evaluation usually was negative, the evaluation process was valuable. It allowed reconsidering the design, planning and programming of the system. Based on the successful conversions, even though the system did not fulfill the goal of converting between all formats, the possibility for it to complete all conversions is not far ahead.

5 Findings

In this section, the findings from the interviews and the case are presented.

5.1 Semi-structured interviews

In the semi-structured interviews, the conversation was divided in two main topics; standards in general, and the open standards focused on in this these; OOXML, ODF and HTML.

In these interviews, some deeper insights regarding standards were made. The two interviewees shared some commonalities and differences concerning what a standard is defined as, and what a standards implications and purpose is.

The findings from the semi-structured interviews are found below, separated into two sections, one for the interview with Håkon Wium Lie, the other section dedicated to the interview with Ole Hanseth.

5.1.1 Interview with Håkon Wium Lie

Håkon Wium Lie is chief technology officer of Opera Software in Oslo, Norway; before joining Opera, he was the style sheet activity lead at the W3C, where he proposed the CSS concept in 1994. He is also the external advisor in this thesis.

Knowing the main inquiry of this thesis, Håkon gave a definition of a standard narrowed down to the problem area:

A standard is a specification that describes formats used for storing or transferring.

When asked to explain what a standard is, he emphasized the importance of *“what that crosses the line”*. What goes on inside or outside a computer is not important until you are crossing the line to connect the two, and some sort of common ground needs to be set, basing the need for using standards. Further, he pointed out the tension between political and technical aspects of standards.

The purpose of a standard was recognized to be communication and interaction leading to exchange of information and trade.

Using an example from trade industry, he went in detail on what he considered a standard to be. It was viewed as a common measuring unit for a practical use, mentioning the well known story of how the wheel-width of

the old horse carriages decided the present day standard railway width for the rail tracks.

Checking that standards are being followed to intent on the Internet was viewed difficult, at least compared to the aforementioned railway track. He noted that even though you are not following the specification of a standard the way it is intended, you still use the building blocks of that specification, hence you are using that standard.

Further we discussed if there was a need to adapt standards to the third HCI wave described earlier, where the interactions between humans and computers are spreading beyond the workplace, taking a more social place in our lives. Håkon expressed that he did not see the need for to expand standards with this progression.

Regarding overlapping standards, he claimed that this competition is positive and legitimate. This way you have many choices regarding what standard to use, and the solid and good standards often come out on top. He noted that commercial software usually does not support standards for various reasons, one of them being that it is too expensive in either time, money or resource.

Håkon spoke warmly of W3C and HTML when asked about whether the organizations publishing standards today are protecting the purpose of standards. He said HTML is a simple standard, freely available, and both a de facto and de jour standard. A de jour standard is a standard by principle, while a de facto standard, is a standard in practice.

He noted that all the people being able to share over the Internet using HTML has lead to an anarchy which often leads to a lack of following any standards. However, this is not the case with HTML. Without any additional cost then your ISP gives, you can connect to all parts of the world, and use HTML markup to share your information, and this has been a huge success.

Discussing office documents, he reminded that the office document formats existed prior to the World Wide Web. These formats made advanced documents with the main goal of being printed to paper. They were backwards compatible for generations, and far more complex compared to HTML. OOXML is considered complex because they included all the options that had been available in the old DOC format. ODF has a better vantage point where it is simpler then OOXML, while still far more complex then HTML.

Another remark about the focused standards, was that ODF and OOXML are close in time, use and functionality. This commercial battle between the two actors was regarded as healthy, but it may confuse users and create problems. The competing specifications should differ in more then syntax. Since Microsoft just had issued a statement saying that they are going to

support ODF, Håkon noted that the way it looks at the moment, OOXML will disappear and that over time, ODF will come out on top. This is because it is impractical to support two formats satisfying the same user need. Albeit, there is a lot of prestige in OOXML for Microsoft, and you can never predict the future.

HTML was believed to live the longest life, at least in backwards compatibility considering the vast amounts of information is stored in this format on the Internet today. He noted that the expansion of the web has been a success for ideas, but the specification hardens since so many people use HTML that it is hard to change the specification. The reason for using HTML earlier was due to its simplicity, and now its also due to the amount of documents written in this format.

Reflecting over when people notice standards, Lie commented that most people only notice standards when something fails. Perhaps in an URL on the web, and in file extensions.

Not surprisingly, the HTML format is Håkons preference in everyday life. The reason for that being that he claims this format has the ability to cover everything he needs a document format to do. With hints of resent, he explained that he is forced to use office document formats when dealing with schools, and it is often the doc format.

5.1.2 Interview with Ole Hanseth

Ole Hanset is a professor in the Department of Informatics at the University of Oslo. His research revolves around infrastructures, standardization, and actor networks.

Ole Hanseth's definition of a standard, in a wide sense, was:

A standard is a specification or structure that defines something that is common for a large group of people

Further, he explained that a specification can be thought of as a recipe for doing an activity. Hence, when an activity is being executed, but the entity that performs this activity is interrupted and a new entity has to take over, that new entity has a recipe to follow, leading to a successfully executed activity.

In order for a standard to be regarded as such, it is not enough for Hanseth that it is defined a standard by the appropriate organization, it also has to be widely used and implemented.

Another quality of a standard was identified to be that it sometimes enables coordination of work. To elaborate, if a group of people perform a task

where there are dependencies on subtasks, you don't have to know all of these subtasks in order to do the one you are assigned to do.

The best example of a standard he could think of was the natural language. In Norwegian language, the language has a specification decided by språkrådet, while a great number of dialects utilize different varieties of the words making up the language.

Hanseth categorized standards into three different types of standards:

- Security
- Quality
- Compatibility
 - Communication

The type of standard significant for the thesis is the communication type standard. Hanseth stated those type of standards were often needed when a large group of people need to do an activity the same way. Like communicating over the web, or talking on a phone.

He also pointed out that it's a challenge to combine flexibility and standards. Acknowledging that the setback of having a large group of people using a standard, makes it resistant to change, consequently less flexible.

When asked about the reason for having overlapping standards, or standards that covers the same need, he explained that this is the way it inevitably is. The reason being historical, or logical. To elaborate, he explained that in some cases, for instance the health services, there is a standard for storing patient data in a certain way in a country, while in a different country they store the same type of data by a different standard. Deducting that when you make standards independent of each other, but for the same purpose, they are bound to overlap.

Related to overlapping standards, we discussed the situation where overlapping standards merge together. Using the same example as above, he described that sometimes, the system used in one country is to be acquired in another. The standard for storing the data will then either be merged, used instead of, or substitute the prior standard used for storing patient data in the country purchasing the system. Although, there is no way of knowing if a standard will affect other standards in the future. The importance of backwards compatibility was voiced in association with this part of the conversation.

When asked about the organizations that define standards, he said that those will protect the purpose of a standard as good as anything else. He noted that organizations like ISO are very bureaucratic and slow, but have a successful strategy for publishing standards.

DOCX was Hanseth's choice of format in everyday life. The reason for this was that it came setup with his computer, and that everyone else uses it. However, his experience with the format he used was described as rather lousy. The result of these experiences made him stop using other formatting than simple text and a simple layout of his document. He would rarely use figures, since he found using them often would result in a mess. With these reasons in mind, he regards the format as a poor standard.

He noted that people rarely pay attention to standards, unless something is wrong or crashing. And that people usually don't care as long as things appear to work.

Ole Hanseth pointed out the increasing converge to XML at the end of the session. He recognized that several special data structures needed to be supported, especially in the health services, and keeping the data in XML makes the task of sharing data easier.

5.2 Case findings

There were two main findings from the case. Firstly, the regarding the overall experience drawn from the case. Secondly, the potential of the case will be detailed.

5.2.1 Experience

The problem of converting between the focused formats is a very complex problem, and this complexity offered many challenges. Using XML as a dispatcher, and having a decent library for parsing helped a lot with dealing with this complexity, although it was still underestimated.

The initial idea of converting through mappings resembling a dictionary prevailed, and was an efficient solution to the complex issue, and gave a simple frame of mind to view the problem with.

5.2.2 Potential

A framework that performs conversions like the one presented in this thesis, is found plausible. The overlapping formats do share mappings between eachother with equivalent sets of elements.

The ability to expand the framework to include other formats that cover the same needs is also plausible. An implemented solution of the encountered problem with conversions from DOCX to HTML and ODT, is within reach and will be discussed in the following chapter.

6 Discussion

In the problem definition, a set of questions were raised. In this chapter, we discuss these questions in relations to the findings and theory presented in the former part of the thesis.

We start by looking at the main inquiry, then move on to the theoretical questions, and lastly the technical questions.

6.1 Is it possible to convert between OOXML, ODF, and HTML?

First of all, the complexity of this task should be recognized. The variations in the specifications are vast, and mapping all aspects of the formats between each other is considered to be an elongated task.

Observing the results from the framework, it shows that just accomplishing text conversions without any formatting proved to be an enveloping and complex task. However, it was a feasible task, and there were successful conversions of text content between most the focused formats. Assuming the remaining parts of the specifications could be mapped in a similar way, complete conversions between the focused formats are more then likely. Remembering that complete conversions to HTML is to be supplemented with CSS, microformats, and possibly MathML. The extensions of the framework in order to do so are elaborated below under the technical subquestion.

Considering the amount of converters and plug-ins dedicated to converting between ODF and OOXML, and the existence if not abundance of converters between those two formats and HTML, you should at least have the ability to convert to one format at a time. This could be achieved utilizing both conversion algorithms from proprietary applications, and from open source software. The two Office document applications this thesis concerns itself with; MSOffice and OpenOffice.org, have recognized HTMLs cast coverage, and adapted to this market. Microsoft Word has a “save as Web Page” option available in Word 2007, and ODF even has a preview in web option in addition to being able to store your document in HTML. The main difficulty seems to be to have people use these alternatives, either due to not knowing of their existence, or lack of interest.

As for Microsoft including support for ODF, some views differ from others. Gary Edwards of the Open Document Foundation posted[51]:

There are some bumps of misunderstanding that must be addressed.

For instance, people continue to insist that if only Microsoft would implement ODF natively in MSOffice, we could all hop on down

the yellow brick road, hand in hand, singing kumbaya to beat the band.

Sadly, life doesn't work that way. Wish it did.

He argues that that neither ODF or OOXML are inclined to compromise their specifications. They are both constructed to fit in one specific application, and is tailored as such. This does not speak for an easy transition to embed them in the same environment. Regarding the system, this is not viewed a problem, seeing that the system is not supposed to change anything in the files, or be fitted in an applications environment, but rather convert them with static mappings from a database.

As Håkon noted, to support two file formats can prove impractical. Nonetheless, it is trusted that Microsoft choice to support ODF is done with some grounding. Gary points out that Microsoft has 550 million desktops that expects backwards compatibility and some sort of continuity. And according to Edwards, Sun Microsystems already stated that they will not compromise the implementation of OpenOffice.org in order to be compatible with those 550 million MSOffice desktops. The logic in this as a business decision is clear, you don't make an effort for your competitor. It appears however that the first tries to merge the two formats, if not by specification but by application has been done, and with HTML already in the mix, a framework that supports all three formats, even if it is MSOffice, could be in the distance.

Regarding the main inquiry, the findings collected in this thesis supports a positive answer to this question, but with dependencies. These would be that you know where to find it, and that it works sufficiently.

The theoretical part of the problem definitions follows, the questions highlighted as below:

6.2 What is a standard?

As anticipated, the definition of a standard seemed quite different depending on perspective.

Using the different groups of actors Krechmer argued in his article [58]; implementers, users and creators of standards, a grouping for these perspectives is provided. For an implementer a standard could be a set of rules to apply to a situation or environment, while the user only uses the standards, perhaps oblivious to its existence unless in need of support. For the creators, the set of rules is possibly first and foremost a process that leads to a path to satisfy a need.

In addition to the perspectives, the definition also depends on what the standard is used for. We can see this reflected in Hanseth's grouping of stan-

dards into types. And also in how Wium Lie narrowed down the definition of a standard suiting this thesis' problem area. Håkon also talked about how a standard is used when something crosses the line. This is closely related to communication, where two people are unable to communicate before they speak the same language, or in relation to this thesis, the inability to digitally share a document until the same standard is utilized.

The definitions supplied to this thesis have some main components in common. In both the interviews, the word specification was used for the technical part of a standard. As we remember from section 3.3.1, Krechmer notes it as common agreements, and MIT labs refers to a standard as set of rules, conditions or requirements. In all cases, a standard is recognized to have a set amount of descriptive rules on how to accomplish an activity. Much like a recipe as discussed with Hanseth. Further, most of the standard definitions, and both interviews, emphasize that the set of descriptive rules has to be common for a large group of people in order to be called a standard.

As both the interviewees noted, it is difficult to change a standard as their userbase grows. It becomes complicated to reach all users with a change to the standard, be it an enhancement or a necessary fix. But in order to use a standard, as Håkon pointed out with HTML as an example, you don't have to use it appropriately. In relation to this, we have to consider how rapid changes happen in the IT-industry. Based on this, a standard in mint condition in regards to its purpose is only destined to last a set amount of time.

To cope with the changes in the industry, the standards has to be expanded or adapted, or new standards will take over the new territories. There is a limit to how much a standard can be expanded or adopted before it gets too cumbersome, using the MSOffice format as an example since the backwards compatibility over a long period of time has cluttered the specification of the format. The requirement for backwards compatibility is the major reason for this detriment, but it is an unavoidable requirement to ensure we can access our historical endeavors.

In the interviews, the interviewees agreed that regular people usually notice standards when something fails. It was noted that people for the most part just use whatever is at their disposal. The few times people are consciously aware that they are using a specification or a format, is at the file extensions, perhaps remembering what you can open in what program. Choosing a format for everyday people is consequently much decided by the market. Hence, no intentional choice of standard is taken for most people.

Hanseth touched in on a very interesting subject where he announced that he just adapted to what he had at his disposal, limiting his formatting due to his applications performance. Non-technical people may have a higher likelihood of leaning towards this solution. There is a belief that technical

savvy people at least attempt to locate a better solution to satisfy his or her need before surrendering to this behavior.

The thought of standards operating in an expanding sense with the third HCI wave, was disputed in the interview with Håkon. Considering the converge of the industry to XML as a layer of exchange, it may support the irrelevance of standards needing to expand or adapt in a heavyset manner. Albeit, the new uses of technology should complement the development of new standards. Unless the needs for standards for our technology are saturated, we would need new standards coping with the expanded use.

While political and economic aspects to this question is recognized, it will not be addressed in this thesis.

Summing up all of this, a standard can be seen as a market-driven set of descriptive rules to perform an interactive activity that is used by a large group of people.

6.3 Do open standards help to enable communication and universal design?

As mentioned above, a regular person is likely to be unaware of standards as they use them. Alas, for a print disabled this may not be the case. As Miriam Nes describes in her thesis [41], the print disabled may improve their environment with a system like DAISY[33]. The digital playback in these sort of systems use the specifications of a standard to organize the information on a printed medium to be read out loud to the user. Hence, when a standard is not used properly the system will fail in rendering the information on the printed medium to the person using it. If the system succeeds, in on overy simplified manner, the document is considered universally designed. Below, we will discuss whether open standards improve this situation or not.

Let us first consider how it would be for a print disabled person without any standards at all. If there was an abundance of ways to communicate over digital mediums, the task to convey information through a playback system is next to impossible. This would apply not only to print disabled, but to all digital communications. But since we seem to enjoy to communicate digitally, some standards have been set, either by the market or by standard setters.

Further, we narrow this down to how it would be if there were no open standards, just proprietary standards. The fate of a print disabled persons ability to read information would then be in corporate hands. Open standards therefore aid universal design both in cost and freedom to choose.

Open standards have the quality of being free to use and implement, thus causing no financial supplements to any software constructed for enabling universal design. Krechmer states in his list of requirements for open standards that a transparent process is of great importance. This gives the stakeholders the opportunity to get involved in the creation of the standard and having a chance to voice opinions or needs, making open standards hold the ability to become a participatory design process.

In development methods, the participatory design approach to development have reaped benefits like higher probable use of the system, a sense of ownership, and higher customer satisfaction as well as secured use of the end product. Bratteteig claims that in Scandinavia we also choose this way of developing systems due to an increase in workplace democracy by giving the members of an organization the right to participate in decisions that are likely to affect their work[43]. This could perhaps be applied to the development of open standards too, but with caution. As pointed out by Jakobs et al.[55], some knowledge of the area is required, to avoid this situation to become counterproductive. Therefore, unlike in participatory design where actual end users are a part of the development process, including a group Krechmer points out; namely implementers of standards. They would have in-depth knowledge of the situation the open standard will be applied to, and provide meaningful requirements to a standards setting committee, in addition to making the open standards development process as democratic as possible without adding a counterproductive element. Perhaps the implementers even would ensure the universal design quality of open standards further then at present day.

Some efforts are done towards universal design on the web. The Web Accessibility Initiative (WAI)[34], is an example of that. Their guidelines are considered to be the de facto international standard for Web accessibility. But in order for these guidelines to be useful, they have to be implemented by the many developers that create websites.

A great deal of today's websites do not follow the HTML specification. Within universal design, this is a common problem. It does not matter if an open standard is a great one, if it is not followed. The reason for this situation could be that the HTML specification is very simple, causing a great deal of people that are not technologically savvy to make a websites. This lack of knowledge about how to properly use a specification is identified as a main culprit for hindering universal design on the web by Hauge and Fardal[52]. Despite these difficulties, HTML is the most successful open standard, with its vast reach, and ease of use, it is the best candidate to enable universal design on the web.

Concerning the part where it is questioned if open standards help enable communication, we consider the governmental influence. As mentioned

in section 3.3.5, an increasing amount of governments use open standards for their public information. This is contributing to free communication between governments and the public. Perhaps along with this contribution, the government influence the public to utilize these same applications for auxiliary documents in their everyday lives due to the convenience of relating to one application only.

Regarding the question of open standards aiding to enable communication and universal design, it is fair to say that they do just that. With regards to both the addition of freedom of choice and free of cost. Although they cannot fulfill this enabling without the help of the developers.

The technical questions revolved around the system are discussed below:

6.4 Is it possible to create a framework that eases conversions between overlapping formats ?

To address this question, we make use of the case framework of the thesis, and OOXML, ODF and HTML. HTML is considered overlapping with OOXML and ODF as long as it is supplemented with additional markup languages and CSS.

In a lecture in a course at the University of Oslo [35], it was stated that

The HTML[sic] has turned the Internet into a world-wide library.
The XML[sic] has turned the Internet into a world-wide business integration platform.

This statement is considered to be valid, whereas XML is extremely suitable for interoperability, due to its structure and ease of transfer. Hence, using XML as a basic building block for the framework is a beneficial and logical choice.

Regarding the framework, there is a long way to go before it supports full conversions between the overlapping formats. However, it did complete conversions between most formats. This shows there is some promise in the framework to support full conversions in the future. It also unraveled some issues that needed to be addressed.

First, I'd like to address the shortcoming of the framework. The reason is expected to be a bug in the code concerning taglevels that reach beyond 1, or in other words mappings that are 1:3 or higher. For the purpose of this thesis, the development of the system had to be stopped in order to write the final report. However, since it did work to convert text content to DOCX, the solution is considered to be in the near future. By conducting test

and performing debugging, the cause of the problem is believed to quickly reveal itself.

Further, we consider the improvements to the system as a whole, starting with the framework. All improvements considered below do not distort the flow of the framework, and maintains the communication between the modules.

A possible improvement is to modify the Dependency table to be future oriented. This could be done in a way illustrated in figure 22.

Dependency
+id: INT, AUTO_INC, PK
+ruleid: INT, FK
+depvalue: VARCHAR
+deptype: VARCHAR

Figure 22: The dependency table.

The value of deptype could define the dependency. If the deptype is defined as "tags" you would know that the depvalue is a list of the tags that is needed in order to map properly with the destined formats element. In the future, this kind of table could also aid when adding attributes to the framework. If the deptype is an "attribute", the depvalue could be the referenced elements required attribute. The Rule table could be modified for this purpose, setting the dependency attribute to be an enumerated list of id's from the Dependency table.

Another way to structure the Rule table is shown in figure 23. This solution disregards attributes, but instead aims to solve the tag problem in the simplest way possible. As we can see there is a taglevel attribute to give away the amount of tags, and also only one dependency to the already existing Dependency table, that contains a set of tags.

Rule
+id: INT, AUTO_INC, PK
+fromSpec: VARCHAR
+toSpec: VARCHAR
+fromSpecData: VARCHAR
+groupTag: VARCHAR
+taglevel: INT
+dependency: INT, FK

Figure 23: Another possible database setup.

Whether the list is in the Rule table as a list of dependencies, or in the Dependency table as a list of tags or values, does little difference at present time. It would be fewer queries to the database with the list in the Dependency table, but it may be easier to code if the list is in the Rule table. The benefits and disadvantages are close to balanced out in relation to the taglevels. The list of dependencies in the Rule table account for it to be used to mark other dependencies than tags in the future. But as mentioned, this accountability may not be needed or purposeful at this point in time, and a more direct solution may be beneficial.

The website has to be augmented to validate the input from the user. It is done without accounting for security or data validity at present time, due to prioritizing other issues. When it is published online, these restrictions should be in place. It will secure the data better, and aid the user when he or she tries to insert faulty data, be it willingly or not.

In addition to these augmentations to the website, it should be rewritten to Python as was the original preference, to be consistant in the use of programming languages in the system. The handling of the websites inputform and the handling of the command line input should also be put in the same module.

The use of CSS for presentation, MathML for representation of mathematical expressions on the web, and microformats was not implemented. Choosing the simplest conversions first kept these formats at a distance. CSS is the undisputed sidekick to HTML giving you the ability to format your information. MathML has been praised by mathematicians in regards to publishing mathematical papers on the web[60], and the use of microformats are expanding[40]. The faith in these addition to HTML to create an equivalent environment as the office documents remain.

To improve further, a test suite should be written. Both functional tests, unit tests, and regressions should be made. Functional tests to make sure single methods returns the expected values, and unit tests to be sure modules work as intended. Regression tests are most critical for the Mapping module, making sure it creates the same converted file on every execution of a source file.

The rationale behind the systems structure is grounded in the desire to automate as much as possible. This is considered to minimize overhead, and reuse of code to match the conversions between overlapping formats. This theory applies as anticipated to the framework made in the case.

Regarding the aim of constructing the framework in a generic manner, where you easily can add new formats to the framework, this was not addressed in a satisfactory manner. There was an efforts to make sure hardcoding was done as little as possible, and as few places as possible. These hard-coded

format specific values should be put in a separate configuration file. At the given moment, in order to add a format to the framework, you have to add it in four different places. You should only have to edit a configuration file, naming the extension, the location of the content XML file, and its context. In addition to adding the rules of mapping to the database of course.

Importing files containing mapping rules would be a vast improvement in speeding up the insertion of rules to the database. Whether they are already existing, or needs to be constructed, the possibility to import these kind of files should be accounted for. Seeing that there are many converters already that have the ability to convert between the formats, assuming the existence of such files is not a long stretch.

The user groups that want to convert between OOXML, ODF and HTML, are as explained in the problem area, everyone that knows how to read or write a document on the computer. This doesn't necessarily mean that they all will use this framework as a stand-alone application. That scenario is rather unrealistic. However, if accessible from the application they usually write or read their files in, the chance for it to be used is much higher. And perhaps the conscious choice of what format to store your file in would be present if you are presented with one.

In a framework like the one described, we have to consider the lack of consistency in websites. As noted above, HTML is a specification that is often used differently then intended. This might lead to difficulties in having conversion work properly as well. Browsers are forgiving with faulty HTML markup and allow for some mistakes, and occasionally assume fixes for defective markup. This approach could be taken regarding the framework also, but with great care. If there are structural problems you do not want to replicate them in the new document, and sometimes the faulty HTML is consistent and needs context to make sense. For example, as Hauge et al. points out in their thesis, some use the header elements for other uses then intended[52].

Another thought is the possibility to remove redundant information from the conversions. If we recall the Markoff chain Weaver explained[62], perhaps this could be applied to these overlapping standards. Some markup are statistically decided by the probability of its occurrence after the current markup. Perhaps this could be applied to a possible solution for the multiple mappings between the tags of the formats in addition to speeding up conversions. Although speed is not a recognized issue in the framework.

Creating the framework gave experience with the incredible complexities of OOXML and those of ODF. This complexity is perhaps inevitable to fulfill a demand for backwards compatibility. What use would a digital document be if you could not read it in two years?

Proprietary versus open standards is a a considerable debate, but for the purposes of this thesis it is acknowledged rather than discussed.

Routing back to the question of the possibility of a framework that eases conversions between overlapping formats, it is safe to say that it is feasible. The framework from the case shows good direction, and has great potential after having successful conversions between most of the formats.

6.5 Can a framework that converts between overlapping formats be developed successfully using open source technology?

All the technology utilized in the case were open source or free software. With the exception of Python, the remaining technologies are perhaps among the most used in web development today.

An advantage with using these technologies, is that they are well-tested, and reliable. Should a problem occur, you have the comfort of having a large community dedicated to the software and the web is filled with resources. Help or information regarding how to fix your problem is easy to find.

With these arguments, and the frameworks potential being as good as any other system developed with proprietary software, it is fair to conclude that it is very possible to develop a framework to convert between overlapping formats using open source technology only.

7 Conclusion

This thesis has been centered around the creation of a system that performs conversions between OOXML, ODF and HTML. It has essentially been divided in two parts that have influenced each other greatly throughout the process; namely theory and practice.

A great deal of insight was gathered in the context of these open standards, to understand their purpose and environment. This context studies were supplementary to knowledge about the formats themselves regarding intent, specification and origin.

The intersection between theory and practice is one that is found closely related to the focused document standards. We find that sharing digital documents without discrimination is proposed through theory with suitable specifications. But only with a purposeful application of these specifications in practice will enable sharing of documents without discrimination of person, hardware or software.

As we recall, the main goal of the thesis was to find out if it is possible to create a system that perform conversions between OOXML, ODF and HTML. I would argue that it is possible, based upon the system developed in this thesis, backed up by the applied theories. Even though the framework had shortcomings, it did show good promise and that converting between the formats is feasible.

Regarding standards, an attempt was made to find out what a standard is perceived as by organizations and individuals. As mentioned in the discussion, a finite definition was not accomplished. Instead, we reached an understanding of the main components common in the definitions of a standard. These main components are that they are a market driven set of descriptive rules for accomplishing an activity that is used by a large group of people.

We also saw that a detailed definition of a standard is easier as its area of application is narrowed down. Open standards is correspondingly difficult to define, but distinguishes itself where this term requires the standard to be free to use and implement in all occurrences.

7.1 Weak points of the thesis

Regarding the investigation of what a standard is perceived as, some quantitative data could be preferable as a supplement. A survey was planned, to send to a group of people that are not technically savvy, asking only what a standard is. This was discarded because it was considered to be a too

long path away from the core of the thesis. However, it is also something that would have grounded the theories of how regular people experience standards in practice.

I'd like to make a remark about the distribution of time regarding the thesis. A lot of time was spent studying the context and theories, perhaps too much. If more time was shifted to the programming, the system might have been closer to fulfill the requirements set during the planning.

A weak point of the system is the low number of test files used. By using a diverse and generous set of test files, more issues on a generic and detailed level could have been unveiled.

7.2 Future work

The most pressing task to be done is to complete the conversions from DOCX to HTML and from DOCX to ODT. To achieve this the bug with taglevels beyond 1 should be addressed.

Some of the future work on the system was proposed in the discussion, and I would argue that many of these proposal should be building blocks of the future work. Most importantly, before expanding the framework to include more tags than text, a test suite should be developed and additional test files should be utilized.

As mentioned previously, developing an open source system in a community have several advantages. Such a community would be preferable regarding the further development of the system. In order to achieve that, the system should be published on sourceforge.net or a similar website. But before that, I would advise to remove all hard-coded parts into a configuration file. In addition to this, the website should validate the input before it touches the public, and also be written in Python. There is some documentation of the system; the code is commented, and the case chapter could be modified into being documentation. This should also be released with the system.

There are two main paths to take after completing the text conversions, those are either to expand the system to include attributes for text, or to include more tags. I would argue that attributes native to the text element should be focused on next. That way the framework is ready to receive additional tags with its attributes and dependencies without any excessive coding.

Regarding the research on what a standard is perceived as, the quantitative survey is recommended to be performed. In addition to this, some scenarios regarding the interaction between one or more humans and one or more

computers could be played out. These activities would help to grasp the human part of the interactive activity of sharing a document in a greater extent. Hence, going deeper into the practical part of document sharing, where this thesis primarily travelled the theoretical part.

There is a vast range of possible extensions to this thesis. An appealing study is to investigate the occurrence of Markoff chains in document formats, or in the contents of the documents. Another, and very interesting research option to look further into, is the effect or possibility of introducing Participatory Design to the open standards development process.

References

- [1] <http://www.forskning.no/artikler/2009/mai/220764>. Online, July 2009.
- [2] <http://opensource.org>. Online, July 2009.
- [3] <http://www.thefreedictionary.com/format>. Online, July 2009.
- [4] http://www.linfo.org/free_file_format.html. Online, July 2009.
- [5] <http://goopen2009.friprog.no/english>. Online, July 2009.
- [6] http://en.wikipedia.org/wiki/File:Iterative_development_model_V2.jpg. Online, July 2009.
- [7] http://www.design.ncsu.edu/cud/about_ud/udprincipleshtmlformat.html. Online, July 2009.
- [8] <http://etsi.org>. Online, July 2009.
- [9] <http://libguides.mit.edu/content.php?pid=14830&sid=110312>. Online, July 2009.
- [10] <http://www.ul.com>. Online, July 2009.
- [11] <http://www.oasis.open.org>. Online, July 2009.
- [12] <http://www.oasis-open.org/specs/>. Online, July 2009.
- [13] <http://www.oasis-open.org/who/>. Online, July 2009.
- [14] <http://www.iso.org>. Online, July 2009.
- [15] <http://ecma-international.org>. Online, July 2009.
- [16] <http://www.iso.org/iso/pressrelease.htm?refid=Ref1181>. Online, July 2009.
- [17] <http://www.w3c.org>. Online, July 2009.
- [18] <http://www.w3.org/People/Berners-Lee/>. Online, July 2009.
- [19] <http://boycottnovell.com/2009/06/04/odf-and-vietnam/>. Online, July 2009.
- [20] <http://www.w3.org/History/1989/proposal.html>. Online, July 2009.
- [21] <http://thenextweb.com/2008/07/29/the-world-wide-web-grows-a-billion-pages-per-day/>. Online, July 2009.
- [22] <http://microformats.org/about/>. Online, July 2009.
- [23] <http://microformats.org/wiki/value-class-pattern>. Online, July 2009.

- [24] <http://www.dessci.com/en/reference/mathml/default.htm>. Online, July 2009.
- [25] http://www.oasis-open.org/committees/membership.php?wg_abbrev=office. Online, July 2009.
- [26] <http://opendocument.xml.org/milestones>. Online, July 2009.
- [27] <http://tools.services.openoffice.org/odfvalidator/>. Online, July 2009.
- [28] http://en.wikipedia.org/wiki/File:Open_XML_main_components.svg. Online, July 2009.
- [29] <http://www.uio.no/studier/emner/matnat/ifi/INF3330/>. Online, July 2009.
- [30] <http://httpd.apache.org/>. Online, July 2009.
- [31] http://www.phpmyadmin.net/home_page/index.php. Online, July 2009.
- [32] <http://www.dropbox.org/>. Online, July 2009.
- [33] <http://www.daisy.org/>. DAISY, Online, July 2009.
- [34] <http://www.w3.org/WAI/>. Online, July 2009.
- [35] <http://www.uio.no/studier/emner/matnat/ifi/INF3210/v06/>. Online, July 2009.
- [36] Friprogsenterets Visjon. <http://www.friprog.no/Hva-er-Friprogsenteret>. Online, June 6th 2009.
- [37] GoOpen 2008. <http://www.friprog.no/Friprogmagasinet/2008-01/GoOpen-2008>, Online, July 2009.
- [38] GoOpen 2009. <http://goopen2009.friprog.no/>, Online, July 2009.
- [39] ODF: nå er det åpne formater som gjelder... *IT-avisen*, 1, 2009.
- [40] John Allsopp. *Microformats - Empowering Your Markup for Web 2.0*. Springer-Verlag New York, www.springeronline.com, 2007.
- [41] Miriam Eilen Nes Begnum. Appraising and Evaluating the Use of DAISY - For Print Disabled Students in Primary and Secondary Education. Master's thesis, University of Oslo, 2007.
- [42] Tim Berners-Lee. HTML Design constraints, 1992. <http://www.w3.org/MarkUp/HTMLConstraints.html>, Online, July 2009.
- [43] Gro Bjercknes and Tone Bratteteig. User Participation and Democracy. A Discussion of Scandinavian Research on System Development. *Scandinavian Journal of Information Systems*, 7(1), 1995. <http://folk.uio.no/tone/Publications/Bjerk-bratt-sjis-i95.html>, Online, July 2009.

- [44] Michael Brauer, Robert Weir, and Mary McRae. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office#announcements, Online, July 2009, 2009.
- [45] Vannevar Bush. As we may think. *Atlantic monthly*, 1945. <http://www.theatlantic.com/doc/194507/bush>, Online, July 2009.
- [46] Susanne Bødker. When second wave HCI meets Third Wave Challenges. ACM, 2006. University of Aarhus, Denmark - Department of Computer science.
- [47] Dan Connolly, Rohit Khare, and Adam Rifkin. The evolution of Web Documents: The Ascent of XML. *World Wide Web Journal*, 2:119–128, 1997.
- [48] Patrick Durusau, Michael Brauer from Sun Microsystems Inc, and Inc. Oppermann, Lars from Sun Microsystems. *Open Document Format for Office Applications (OpenDocument) v1.1*, 2007. <http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf>, Online, July 2009.
- [49] Ecma. *Office Open XML File Formats*, 2006. <http://www.ecma-international.org/publications/standards/Ecma-376.htm>, Online, July 2009.
- [50] Ecma. *ECMA-376 2nd edition Part 4*, 2008. The file: ECMA-376, Second Edition, Part 4 - Transitional Migration Features.pdf Annex D.
- [51] Gary Edwards. CFD and Grand Convergence. <http://openstack.blogspot.com/2007/10/cdf-and-grand-convergence.html>, 2007. Online, July 2009.
- [52] Frank Sætrehaug Fardal and Aud Marie Hauge. Universell utforming. Master's thesis, University of Oslo, 2006. <http://www.duo.uio.no/sok/work.html?WORKID=28982>, Online, July 2009.
- [53] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - The complete book - International Edition*. Prentice-Hall Inc., 2002.
- [54] Hewett, Baecker, Card, Carey, Gasen, Manteia dn Perlman, Strong, and Verplank. ACM SIGCHI Curricula for Human-Computer Interaction. 1996.
- [55] Kai Jakobs, Rob Procter, and Robin Wiliams. User participation in standard setting - the panacea? *ACM*, 6, 1998. <http://portal.acm.org/citation.cfm?id=301693>, Online, July 2009.
- [56] Peter Johnson. *Human computer interaction - Psychology, Task Analysis and Software Engieneering*. McGRAW-HILL Book Company Europe, 1992. <http://sigchi.org/cdg/cdg2.html#2.1>, Online, July 2009.

- [57] Christopher A. Jones and Fred L. Drake Jr. *Python & XML*. O'REILLY, 2002.
- [58] Ken Krechmer. Open Standards Requirements. *The International Journal of IT Standards and Standardization Research*, 4(1), 2006. <http://www.csrstds.com/openstds.pdf>, Online, July 2009.
- [59] Håkon Wium Lie. *Cascading Style Sheets*. PhD thesis, University of Oslo, 2005. <http://people.opera.com/howcome/2006/phd/>, Online, July 2009.
- [60] Robert Miner. The importance of MathML to Mathematics Communication. *Notices of the AMS*, 52(5), 2005.
- [61] OASIS. On the surge of the public embracing open source. <http://www.odfalliance.org/mission.php>, 2007. Online, July 2009.
- [62] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois Press, 1975.
- [63] Helen Sharp, Yvonne Rogers, and Jennifer Preece. Interaction Design: Beyond Human-Computer Interaction. http://www.id-book.com/downloads/Chapter_7_ID2e_slides.ppt, 2007. Online, July 2009.
- [64] Ian Sommerville. <http://www.comp.lancs.ac.uk/computing/resources/lanS/SE7/Presentations/PPT/ch17.ppt>, 2004. Presentation, Online, July 2009.
- [65] Richard Stallman. Copyright vs Community in the Age of Computer Networks - Free software and beyond. Lecture in Chateau Neuf the 23rd of February in Oslo, 2009.
- [66] Andrew Updegrave. "Openness" and the Pursuit of Knowledge. *Standards Today*, VII(398), 2008. Featured article, <http://consortiuminfo.org/bulletins/apr08.php#feature>, Online, July 2009.
- [67] W3C. *HTML 4.01 Specification - W3C recommendation 24 December 1999*, 1999. <http://www.w3.org/TR/html401/>, Online, July 2009.
- [68] Håkon Wium Lie and Janne Saarela. Multipurpose web publishing using html, xml, and css. *Communications of the ACM*, 42:95–191, 1999.

Appendix A: Consent forms

Consent form

I, Ingunn Rønningen is a student at the University of Oslo, currently writing a master thesis with standards as a topic, more specifically office document standards, and the conversion of those. In coherence of this thesis I conduct interviews with competent people within the field of interest.

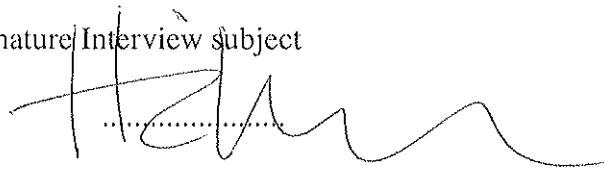
For research ethical reasons, I will need your signature, and by that your consent in giving me an interview on the terms described below.

The interview is voluntary, and you are free to interrupt at any time without any negative consequences.

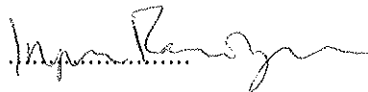
The information retrieved will be used in the thesis and thesis only.

The information will be kept secure and used responsible.

Signature Interview subject



Signature Student



Date



Consent form

I, Ingunn Rønningen is a student at the University of Oslo, currently writing a master thesis with standards as a topic, more specifically office document standards, and the conversion of those. In coherence of this thesis I conduct interviews with competent people within the field of interest.

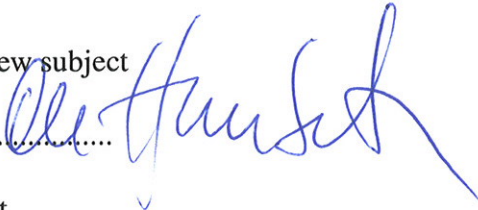
For research ethical reasons, I will need your signature, and by that your consent in giving me an interview on the terms described below.

The interview is voluntary, and you are free to interrupt at any time without any negative consequences.

The information retrieved will be used in the thesis and thesis only.

The information will be kept secure and used responsible.

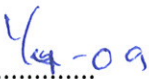
Signature Interview subject

.....

Signature Student

.....

Date

.....

Appendix B: Semi-structured interview questions

1. What is a standard – explanation
2. How would you define a standard?
3. Why do standards exist? Note: HCI wave overlap connect?
4. What is the purpose/intention/formål of a standard?
5. What are the actual implications of standards? Note: universell utforming
6. Why do we have overlapping standards like OOXML and ODF?
7. Are those who create standards protecting the purpose of standards?
+what and why
8. Do you think people notice standards at all, and in what sense?
9. In the future, will we still have overlapping standards for communicating documents, or will there be a phasing off or a merger into a broad standard?
10. What format do you use on your documents, and why?
11. Have you experienced difficulties using standards in handling of office documents?

Appendix C: Code

Readme.txt file:

Jul 25, 09 15:28	readme.txt	Page 1/1
<pre>1 This system is created as part of a master thesis in the University of Oslo. 2 It is a framework for constructed to perform conversions between overlapping 3 formats. 4 5 It currently converts between ODT, HTML and DOCX (with a current issue so you ar 6 e not able to convert from DOCX, but to DOCX). 7 8 The only input to the framework is from command line, its 9 Usage: python formHandler.py filetoconvert whattoconvertto 10 11 Some test files are supplied in a testfiles folder. 12 13 There are some rm commands and system calls throughout, 14 so keeping the file structure intact is extremely important. 15</pre>		
Saturday July 25, 2009		1/1

View - HTML markup:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Converter 1.0
</title>
</head>
<body>
  <h1>Convert and add rule</h1>

  <form action="handler.php" method="post">
    <h2>Convert file:</h2>

    <input type="file" name="file" value=""/>
    To: <select name="targetFormat">

      <option value="ODT">ODT</option><option value="HTML">HTML</option>
      <option value="DOCX">DOCX</option>
    </select>
    <input type="submit" value="Press me" name="btn"/>

    <h2> Add rule:</h2>

    From format:
    <select name="fromSpec">
      <option value=""></option>
      <option value="DOCX">DOCX</option>
      <option value="HTML">HTML</option>
      <option value="ODT">ODT</option>
    </select>
    <input type="text" name="fromSpecData"/>
    <br/>
    To format:
    <select name="toSpec">
      <option value=""></option>
      <option value="DOCX">DOCX</option>
      <option value="HTML">HTML</option>
      <option value="ODT">ODT</option>
    </select>
    <input type="text" name="toSpecData"/>
    <br/>
    Group Tag:
    <select name="groupTag">
      <option value=""></option>
      <option value="start">start</option>
      <option value="end">end</option>
    </select>
    <br/>
    Dependency:
    (Starting with the lowest, ending in the highest)
    <input type="text" name="dep"/>
  </form>
</body>
</html>
```

Model:

```
Jul 25, 09 16:52 handler.php Page 1/1
1 <?php
2
3 $usr = "root";
4 $pwd = "lol";
5 $db = "converter";
6 $host = "localhost";
7
8 $cid = mysql_connect($host, $usr, $pwd);
9
10 if (!$cid) { echo("ERROR: " . mysql_error() . "\n"); }
11 if ($_SERVER['REQUEST_METHOD'] == "POST") {
12
13     $thefile = $_POST['file'];
14     # if its a file, it wont work, theres no further handling
15     if ($thefile != ""){
16         print $thefile;
17     }
18     else{
19         $tex = "New rule added";
20         $fromSpecData = $_POST['fromSpecData'];
21         $fromSpec = $_POST['fromSpec'];
22         $toSpec = $_POST['toSpec'];
23         $toSpecData = $_POST['toSpecData'];
24         $groupTag = $_POST['groupTag'];
25         $dep = $_POST['dep'];
26
27         $SQL = " INSERT INTO Rule ";
28         $SQL = $SQL . " (fromSpec, toSpec, fromSpecData, toSpecData, groupTag) VALUES ";
29         $SQL = $SQL . " ('$fromSpec', '$toSpec', '$fromSpecData', '$toSpecData', '$groupTag') ";
30         $result = mysql_db_query($db, $SQL, $cid);
31
32         # If there is a dependency it should be inserted correctly
33         if ($dep != ""){
34             $ruleidquery = "SELECT id from Rule where fromSpec = '$fromSpec' and toSpec = '$toSpec' and fromSpec
35 Data = '$fromSpecData' and toSpecData = '$toSpecData' and groupTag = '$groupTag' ";
36
37             $res=mysql_query($ruleidquery);
38             $row = mysql_fetch_row($res);
39             $ruleid= $row[0];
40
41             $query = "INSERT INTO Dependency(ruleid, thetags) VALUES ($ruleid, '$dep')";
42
43             $depreresult = mysql_db_query($db, $query, $cid);
44             if (!$depreresult){
45                 echo("ERROR: " . mysql_error() . "\n$SQL\n");
46             }
47
48             $getdepid = "SELECT id from Dependency where thetags = '$dep' and ruleid = $ruleid ";
49             $res=mysql_query($getdepid);
50             $row = mysql_fetch_row($res);
51             $depid = $row[0];
52             $putidinrule = "UPDATE Rule SET depid = $depid where id = $ruleid ";
53             $depreresult = mysql_db_query($db, $putidinrule, $cid);
54             if (!$depreresult){
55                 echo("ERROR: " . mysql_error() . "\n$SQL\n");
56             }
57
58             $tex = $tex . " with dependency.";
59         }
60
61         if (!$result) {
62             echo("ERROR: " . mysql_error() . "\n$SQL\n"); }
63     }
64 }
65
66 mysql_close($cid);
67
68 ?>
```

```
1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  # This module treat the input and sends it along to the treatfile
6  # command line only at the moment
7
8  import sys, os
9
10 def supportedFileTypes():
11     return ["odt", "html", "docx"]
12
13 from treatFile import *
14
15 # check usage in args from commandline
16 try:
17     theFile = sys.argv[1]
18     convertTo = sys.argv[2]
19 except:
20     print "Usage: python formHandler.py filetoconvert whattoconvertto"
21     exit(1)
22
23 # make sure the file exists
24 if theFile == "":
25     print "Empty file input"
26 else:
27     if os.path.exists(theFile):
28         print "The file is:", theFile
29         print ""
30         fileformat = theFile.split(".")[1].strip()
31
32         # some input tests
33         if fileformat == convertTo:
34             print "Why would you want to convert to the same format?"
35             exit(1)
36         if fileformat not in supportedFileTypes() :
37             print "Cannot convert to "+fileformat+", supported formats are"+' '.join(supportedFileTy
38 pes())
39             exit(1)
40         if convertTo not in supportedFileTypes():
41             print "Cannot convert to "+convertTo+", supported formats are"+' '.join(supportedFileTy
42 pes())
43             exit(1)
44
45         print "Processing the file "
46         theConvertedFile = processFile(theFile,convertTo)
47         # the location
48         print theConvertedFile
49
50     else:
51         print "The file: %s does not exist."%theFile
```


Control:

Jul 25, 09 17:05	treatFile.py	Page 1/1
<pre>1 #!/usr/bin/env python 2 # -*- coding: iso-8859-1 -*- 3 #Author: Ingunn Rønningen 4 5 from NewFile import * 6 from DOCXFile import * 7 from HTMLFile import * 8 from ODTFile import * 9 from Mapping import * 10 11 import os 12 13 # Treats the file, and tells the other parts of the controller what to do. 14 # return the location of the converted file. 15 16 def processFile(theFileName, ct): 17 """ 18 Get the file object, filename, and what to convert to 19 returns the converted file 20 """ 21 ft = getFileType(theFileName) 22 23 # This is oldfile info gathered in one place 24 if ft == "docx": 25 of = DOCXFile(ft, ct, theFileName) 26 if ft == "odt": 27 of = ODTFile(ft, ct, theFileName) 28 if ft == "html": 29 of = HTMLFile(ft, ct, theFileName) 30 31 # Create a newfile instance to send to mapping. 32 newFileName = theFileName.split("/")[-1].split(".")[2]+"."+ct 33 34 if ct == "docx": 35 nf = DOCXFile("docx", "", newFileName) 36 if ct == "odt": 37 nf = ODTFile("odt", "", newFileName) 38 if ct == "html": 39 nf = HTMLFile("html", "", newFileName) 40 41 # create the mapping object with the old file and new file instances 42 # to make the conversion 43 m = Mapping(of, nf) 44 45 # Fetch the location of the converted file 46 convertedFile = m.getConvertedFile() 47 48 # clean up the environment 49 runcleanup(ct, ft) 50 51 return convertedFile 52 53 def runcleanup(ct, ft): 54 55 #check what has been done and clean up accordingly 56 if ct == "docx": 57 #remove the document.xml from docxcontext 58 os.system("rm docxcontext/word/document.xml") 59 if ct == "odt": 60 #remove the content.xml from odtcontext 61 os.system("rm odtcontext/content.xml") 62 63 def getFileType(fil): 64 """ 65 return the end of the file 66 """ 67 end = fil.split(".")[1] 68 return end</pre>		
Saturday July 25, 2009		1/1

```

1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  #portable import:
6  #gotten from from http://codespeak.net/lxml/1.3/tutorial.html
7  try:
8      from lxml import etree
9      #print("running with lxml.etree")
10     except ImportError:
11         try:
12             # Python 2.5
13             import xml.etree.cElementTree as etree
14             print("running with cElementTree on Python 2.5+")
15         except ImportError:
16             try:
17                 # Python 2.5
18                 import xml.etree.ElementTree as etree
19                 print("running with ElementTree on Python 2.5+")
20             except ImportError:
21                 try:
22                     # normal cElementTree install
23                     import cElementTree as etree
24                     print("running with cElementTree")
25                 except ImportError:
26                     try:
27                         # normal ElementTree install
28                         import elementtree.ElementTree as etree
29                         print("running with ElementTree")
30                     except ImportError:
31                         print("Failed to import ElementTree from any known place")
32
33     class NewFile:
34         """
35         Has all the file information, and parses the incoming file
36         """
37         fileType = ""
38         convertTo = ""
39         fileName = ""
40         unzipdirectory = "unzipped"
41         parsedXML = ""
42
43         def __init__(self, ft, ct, fn):
44
45             self.fileType = ft
46             self.convertTo = ct
47             self.fileName = fn
48
49             #debug
50             #print "Filetype: ", self.fileType
51             #print "Convert to: ", self.convertTo
52             #print "Filename: ", self.fieName
53
54         def parseXML(self):
55             """
56             parse the content files
57             """
58             # what to parse, it needs to be a string, already set in the subclasses
59             theXMLString = self.fileXML
60
61             # parse
62             root = etree.fromstring(theXMLString)
63
64             # set the
65             self.parsedXML = root
66
67             #debug
68             #print "The tree:"
69             #print(etree.tostring(root, pretty_print=True))

```

```
1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  from NewFile import *
6  import os
7
8  class DOCXFile(NewFile):
9      """
10     The OOXML specifics for docx are found in this class
11     """
12
13
14     # Set the data other then the content
15     # extract the content
16     fileXML = ""
17
18     def __init__(self, fileType, convertTo, fileName):
19         NewFile.__init__(self, fileType, convertTo, fileName)
20         # if its gonna be converted
21         if convertTo != "":
22             self.fileXML = self.unpackFile()
23         try:
24             NewFile.parseXML(self)
25         except ValueError:
26             print "Parsing failed"
27
28
29
30
31     def unpackFile(self):
32         """
33         Specifics for unpacking DOCX
34         and returning the XML String with content
35         """
36         print "unpacking"
37         unpackthefile = "unzip %s -d %s/"%(self.fileName, self.unzipdirectory)
38         os.system(unpackthefile)
39         # where the specific content file will be found
40         theContentFile = open("%s/word/document.xml"%self.unzipdirectory,"r")
41         theXML = ''.join(theContentFile.readlines())
42         # remove the files from unzipped, dont need them anymore
43         # be careful
44         removefiles = "rm -rf %s/*"%self.unzipdirectory
45         os.system(removefiles)
46         #print removefiles, " executed"
47         return theXML
48
49
50
51
52
```

```
1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  from NewFile import *
6
7  import os
8
9  class ODTFile(NewFile):
10     """
11     The ODF specifics are found in this class.
12     """
13     fileXML = ""
14     parsedXML = ""
15
16     def __init__(self, fileType, convertTo, fileName):
17         NewFile.__init__(self, fileType, convertTo, fileName)
18         # check if its a to or from file
19         if convertTo != "":
20             self.fileXML = self.unpackFile()
21             try:
22                 NewFile.parseXML(self)
23             except:
24                 print "Parsing failed"
25         else:
26             print "ODT is the target format"
27
28     def unpackFile(self):
29         """
30         How to unpack and find the content of ODT files
31         """
32
33         print "unpacking"
34         unpackthefile = "unzip %s -d %s/"%(self.fileName, self.unzipdirectory)
35         os.system(unpackthefile)
36         # where the specific content file will be found
37         theContentFile = open("%s/content.xml"%self.unzipdirectory,"r")
38         XMLString = ''.join(theContentFile.readlines())
39         # remove the files from unzipped, dont need them anymore
40         # careful here..
41         removefiles = "rm -rf %s/"%self.unzipdirectory
42         os.system(removefiles)
43         #print removefiles, "executed"
44
45         return XMLString
46
47
48
49
50
51
52
53
```

```
1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  from NewFile import *
6
7  class HTMLFile(NewFile):
8      """
9      The specifics for HTML are found in this class
10     """
11     # This format requires a special start and ending to the content file
12
13     fileXML = ""
14
15     def __init__(self, fileType, convertTo, fileName):
16         NewFile.__init__(self, fileType, convertTo, fileName)
17         # check if its to or from file
18
19         if convertTo != "":
20             self.fileXML = self.unpackFile(fileName)
21             try:
22                 NewFile.parseXML(self)
23             except:
24                 print "Parsing failed"
25
26         else:
27             print "HTML is the target format"
28
29
30
31     def unpackFile(self, filename):
32         #content is the html file
33         file0 = open(filename, "r")
34         return ''.join(file0.readlines())
35
36
37
38
```

```

1  #!/usr/bin/env python
2  -*- coding: iso-8859-1 -*-
3  #Author: Ingunn Rønningen
4
5  from NewFile import *
6
7  import MySQLdb
8  from StringIO import StringIO
9  import os
10
11 #portable import:
12 try:
13     from lxml import etree
14     #print("running with lxml.etree")
15 except ImportError:
16     try:
17         # Python 2.5
18         import xml.etree.cElementTree as etree
19         print("running with cElementTree on Python 2.5+")
20     except ImportError:
21         try:
22             # Python 2.5
23             import xml.etree.ElementTree as etree
24             print("running with ElementTree on Python 2.5+")
25         except ImportError:
26             try:
27                 # normal cElementTree install
28                 import cElementTree as etree
29                 print("running with cElementTree")
30             except ImportError:
31                 try:
32                     # normal ElementTree install
33                     import elementtree.ElementTree as etree
34                     print("running with ElementTree")
35                 except ImportError:
36                     print("Failed to import ElementTree from any known place")
37
38 class Mapping:
39     """
40     Converts the file by reading conversion rules from the db
41     """
42     """
43
44     fromFile = ""
45     newFile = ""
46
47     theConvertedFile= ""
48     conn = ""
49     cursor = ""
50
51     def __init__(self, theFile, newFileO):
52
53         self.newFile = newFileO
54         self.fromFile = theFile
55
56         print "In Mapping "
57         print "fromFile is :", self.fromFile
58         print "newFile is :", self.newFile
59
60         # first connect to the db
61         self.connectToDb()
62
63         # find a start tag first
64         newFileString = self.getStart(self.fromFile, self.newFile)
65
66         # traverse the xml tree and go through the conversion by the getRule
67         root = self.fromFile.parsedXML
68         allTheItems = root.getiterator()
69
70         print "Building file"

```

```

71     #if empty no crash
72     themiddle = ""
73     themiddle = self.makexml(root)
74
75     newFileString = newFileString + themiddle + self.getEnd(self.fromFile, self.new
File)
76
77     # Disconnect from the db
78     self.killConnect()
79     # Set the converted file and write and zip if needed
80     self.theConvertedFile = self.fixFile(newFileString)
81
82
83     def makexml(self, item):
84         # some vars
85         thetag = item.tag
86         fromfront = ""
87         fromback = ""
88         newFileXML = ""
89
90         morexml = ""
91
92         # odt style tags come a bit different.
93         if "urn:oasis" in thetag:
94             theend = thetag.split(" ")[1]
95             # extract the namespace
96             thenamespace = thetag.split(":")[6]
97             thetag = thenamespace+"."+theend
98             # as to docx style tags..
99         if "openxmlformats" in item.tag:
100             # do a w but have it in a config later
101             theend = thetag.split(" ")[1]
102             thenamespace = "w"
103             thetag = thenamespace+"."+theend
104
105         # get the equivalent tag
106         cTag = self.getRule(thetag, self.fromFile.fileType, self.newFile.fileType)
107
108         contents = ""
109         cElement = ""
110         # if the tag isnt empty or unvalid
111         if cTag != None and cTag != "" and cElement != "<>" and cElement != "</>":
112             taglevel = 0
113             wheretoggetcontent = ""
114
115             if isinstance(cTag, dict):
116                 # how far down to the content
117                 taglevel = int(cTag["taglevel"])
118                 # what item holds the content
119                 wheretoggetcontent = cTag["dependency"]
120                 # the tag to convert to + the new dep
121                 cTag = cTag["fromspec"]+"."+cTag["dependency"]
122
123             # if theres no 1:many mapping
124             if item.text != None and item.text != "" and taglevel == 0:
125                 contents = item.text
126
127             # 1:many mapping
128             if taglevel != 0:
129                 try:
130                     # find the contents using the item it is mapped to in rules
131                     contents = self.recursiveContentFinder(item, item)
132                 except:
133                     contents = ""
134
135             # make for the tags we know
136             if cTag != "Unknown":
137
138                 # built up by more then one tag
139                 multitag = ""

```

```

140     multistart = ""
141     multiend = ""
142     if "." in cTag:
143         multitag = "ismulti"
144         thetags = cTag.split(",")
145         for atag in thetags:
146             # changed where they start and end
147             multistart = "<"+atag.strip()+">"+multistart
148             multiend = multiend + "</"+atag.strip()+">"
149     # 1:1
150     if multitag == "":
151         fromfront = fromfront + "<"+cTag+">" + contents
152     else:
153         fromfront = fromfront + multistart + contents
154
155     # dorecur
156     for child in item.getchildren():
157         morexml = self.makexml(child)
158
159     # fromback
160     if cTag != "Unknown":
161         if multitag == "":
162             frombacktag = "</"+cTag+">"
163             fromback = frombacktag + fromback
164         else:
165             fromback = multiend + fromback
166
167     newFileXML = fromfront + morexml + fromback
168
169     return newFileXML
170
171 # Finds the content of a 1:many mapping
172 # uses the mapping in the Rule table
173 def recursiveContentFinder(self, item, stopper):
174     thestopper = stopper
175     if item == thestopper:
176         return item.text
177     for kid in item.getchildren():
178         if kid.getchildren() != None:
179             for achild in kid.getchildren():
180                 self.recursiveContentFinder(achild, thestopper)
181
182
183 def getConvertedFile(self):
184     return self.theConvertedFile
185
186 # Writes the file to its context
187 # If its docx or odt it zips
188 def fixFile(self, newFileString):
189     ofolder = "outputFile"
190     destination = self.newFile.fileType
191     if destination == "odt":
192         whereToPutTheFile = "odtcontext/content.xml"
193     elif destination == "docx":
194
195         whereToPutTheFile = "docxcontext/word/document.xml"
196     elif destination == "html":
197         whereToPutTheFile = "%s/%s" % (ofolder, self.newFile.fileName)
198
199     # write the file to the destination
200     fileo = open(whereToPutTheFile, "w")
201     fileo.write(newFileString)
202     fileo.close()
203
204     # zip it to the destination
205     if destination == "docx":
206         zipthefile = "cd docxcontext && zip -r %s *"%(self.newFile.fileName)
207         os.system(zipthefile)
208         os.system("mv docxcontext/%s outputFile/. "%self.newFile.fileName)
209     if destination == "odt":

```



```

210     zipthefile = "cd odtcontext && zip -r %s *"%( self.newFile.fileName)
211     # > /dev/null for ingen output
212     os.system(zipthefile)
213     os.system("mv odtcontext/%s outputFile/."%self.newFile.fileName)
214
215     # location of the converted file
216     return "File is at %s/%s"%( ofolder, self.newFile.fileName)
217
218     # get the dependency
219     def getDependencies(self, ruleid, direction):
220         thetags = ""
221         if direction == "to":
222             sql = "SELECT thetags, id from Dependency where ruleid = %d"%ruleid
223         else:
224             print "Wrong input to getDependencies"
225             exit(1)
226         self.cursor.execute(sql)
227         # don't need the id yet
228         for tt, depid in self.cursor:
229             thetags = tt
230         return thetags
231
232     # Gets rule from DB
233     def getRule(self, elementName, froms, tos):
234         theConvertedTag = "Unknown"
235         # check both ways froms = html, tos = odt elemtn = p
236         rulesql = "SELECT id, fromSpec, depid, toSpecData, fromSpecData from Rule where (fromSpec = '%s' and toSpecData = '%s' and fromSpecData = '%s') or (toSpec = '%s' and fromSpec = '%s' and toSpecData = '%s')"%( froms, tos, elementName, froms, tos, elementName)
237
238         self.cursor.execute(rulesql)
239         if self.cursor.fetchone() != None:
240             for id, fspecname, depid, fromspec, tospec in self.cursor:
241                 dependency = ""
242                 # if the fromspec data is the same as the element
243                 if fromspec == elementName:
244                     if depid != 0:
245
246                         # is the dependency from or to
247                         if fspecname.upper() == froms.upper():
248                             dependency = self.getDependencies(id, "to")
249                             theConvertedTag = tospec+ ","+dependency
250                         else:
251                             theConvertedTag = tospec
252                     else:
253                         theConvertedTag = tospec
254
255                 else:
256                     if depid != 0:
257                         # if its the other way around it gets a bit hairy.
258                         # Have to get that items content, so we change it with the innermost tag.
259                         # so send off that info aswell
260
261                         #also check if the dependency is the correct way
262                         if fspecname.upper() == froms.upper():
263                             dependency = self.getDependencies(id, "to")
264                             howmanydeps = len(dependency.split(","))
265                             theConvertedTag = {"fromspec":fromspec, "dependency":dependency, "taglevel":
str(howmanydeps)}
266                         else:
267                             theConvertedTag = fromspec
268                     else:
269                             theConvertedTag = fromspec
270
271                 return theConvertedTag
272
273
274     def getStart(self, fromFile, newFile):
275
276         theStart = ""

```

```

277     oldFileType = self.fromFile.fileType
278     newFileType = self.newFile.fileType
279     # look in both directions - put this back in one query
280
281     fromsql = "SELECT toSpecData, l from Rule where toSpec = '%s' and fromSpec = '%s' and groupTag = 'start'
282     "%(newFileType, oldFileType)
283     self.cursor.execute(fromsql)
284
285     for fdata, bs in self.cursor:
286         theStart = fdata
287
288     tosql="SELECT fromSpecData, l from Rule where toSpec = '%s' and fromSpec = '%s' and groupTag = 'start' "%
289     (oldFileType, newFileType)
290     self.cursor.execute(tosql)
291
292     for tdata, bs in self.cursor:
293         theStart = tdata
294
295     #print "Start %s - %s : "%(newFileType, theStart)
296     return theStart
297
298 def getEnd(self, fromFile, newFile):
299     theEnd = ""
300     oldFileType = self.fromFile.fileType
301     newFileType = self.newFile.fileType
302     #check both directions - put this back in 1 query
303     sql = "SELECT toSpec, fromSpec, fromSpecData, toSpecData from Rule where ((toSpec = '%s' and fromSpec =
304     '%s') or (fromSpec = '%s' and toSpec = '%s')) and groupTag = 'end' "%(newFileType, oldFileType, newFileType,
305     oldFileType)
306
307     fromsql = "SELECT toSpecData, l from Rule where toSpec = '%s' and fromSpec = '%s' and groupTag = 'end' "
308     "%(newFileType, oldFileType)
309     self.cursor.execute(fromsql)
310
311     for fdata, bs in self.cursor:
312         theEnd = fdata
313
314     tosql="SELECT fromSpecData, l from Rule where toSpec = '%s' and fromSpec = '%s' and groupTag = 'end' "%
315     (oldFileType, newFileType)
316     self.cursor.execute(tosql)
317
318     for tdata, bs in self.cursor:
319         theEnd = tdata
320
321     #print "End %s - %s"%(newFileType, theEnd)
322     return theEnd
323
324 def connectToDb(self):
325     #how to connect
326     print "Connecting to DB "
327     self.conn = MySQLdb.connect (host = "localhost",
328     user = "convert",
329     passwd = "doexfl",
330     db = "converter")
331
332     self.cursor = self.conn.cursor()
333
334 def killConnect(self):
335     print "Killing connection to DB "
336     self.cursor.close()
337     self.conn.close()
338

```