# MASTEROPPGAVE

A prototype for Digital Norwegian People's Archive

Utarbeidet av:
Minh Quang Than

Fag:
Masterstudium i anvendt informatikk

Avdeling:
Avdeling for informasjonsteknologi, 2012

Østfold University College

Faculty of Computer Sciences

Master thesis (45 ECTS)

# A prototype for Digital Norwegian People's Archive

*Author:*
Minh Q. Than

*Supervisor:*
Prof. Børre Ludvigsen

Halden 2012

## Abstract

The Norwegian People's Archive of which this is a prototype meant to be a repository for Norwegians to voluntarily deposit their personal documents which are not cover by the Legal Deposit Act of 1998. The material can be deposited interactively in open formats, time to be released into Public Domain can be chosen but not later than 70 years after owner's death. When the material becomes a part of public domain, it can be annotated if the owner allows. A minimum metadata is obligatory at time of submission, and more metadata can be added later. The metadata, file formats, and archives used in the system are open and conforms to the standards developed and used at The Library of Congress. The repository is restricted to Norwegians identified as anyone with a Norwegian National Identity number. The system architecture is derived from the reference model for an Open Archive Information System, focusing on the system scalability and flexibity so that it can scale for serving an anticipated population. The design of the system is platform-independent. The archives in the system are autonomous, which means that the archives are self-described so that in case the archive system goes down, from the archives in the backup media, the whole system can be rebuilt.

**Acknowledgements**

First and foremost, I wish to express my deepest gratitude to Prof. Børre Johan Ludvigsen, who encourages and guides me from the beginning to the last moment of the work. Without his unmeasurable help, I may not be able to finish this dissertation. He is a friend, an inspirer, and an indispensable supervisor who helps me realize that doing research is one of my best pleasure. He helps me figure out the definition of a best teacher, who not only provides knowledge, guidance, experience and opportunity, but also has great patience, helpfulness, and inspiration.

Special thanks to Jan Høiberg, Christian Heide, and Nils Bekken who helps me solve my financial, procedural, and technical problems I encountered during my working period. I appreciate all the help of the Faculty of Computer Sciences for providing me all necessary equipment to build and test the prototype.

I would like to thank Ms Laila Fagerdal, Ms Ellen Høy-Petersen, Ms Inger Steiner Hansen, and the International Office of Østfold University College who have helped me since the first day I came to Norway. Thank Østfold University College and Norway for giving me an invaluable opportunity to live and study in Norway, the country that took a permanent part of my heart. Thank all my friends who stay, study, party, and travel with me during the period we are together.

Thank my family and my friends who always stay beside me, understand me and share my feelings, my problems and my pleasure.

# Contents

# List of Figures

# 1 Introduction

## 1.1 Purpose

In Norway, the law of "Legal Deposit of Generally Available Documents" requires that all the publication outside the private sphere has to be sent to the Norwegian National library for archiving. Many other research projects of digitalization and electronic archives have been conducted so far.. The institutions vary from university, state, national, and even multinational level. The archive content is also very different: historical pictures, cultural heritage computing, old films, old music, or even personal documents. Most of them cover in a small content area, and involve more with accessing rather than widely depositing, rather than a public-available system for all citizens to deposit and preserve for long-term period. However, those previous efforts play as a strong background for building a digital national people's archive.

Currently, the private-own documents of every day life, e.g. video, image, and sound, are out of public preservation. In the other words, they are preserved or not by their owner. Theses documents, however, have significant cultural, historical, and other social values for future generation. With the development of internet and computer technology, digital documents in private sphere are turning into enormous amount, which raises the needs of protecting these heritages. Without a systematic approach, these heritages can be lost unrecoverably. Therefore, a system that helps the citizens preserve their own private documents as archives on a long-term period for future use is a solid solution in this situation. To build such national-wide and privacy-related system successfully, many aspects should be taken into consideration such as privacy and security protection, copyright, preservation and publishing methods, etc.

This thesis is developed as a response for the need of an archive system that will handle and preserve the documents in the private sphere in Norway. Such system can be called *The Digital Norwegian People's Archive*(NPA), or in short *the system* in this thesis for simplicity. The aim of the thesis are the overall design of the system, guidelines for building a complete system, and the attempt to build a prototype focusing on documents ingestion process and medadata input as the first elements toward a working system.

## 1.2 Research strategy and design

The research strategy of the thesis, or the approaching method to solve the research problem [38], is obviously *design and creation.* The design and creation strategy focusing on developing new Information Technology (IT) artefacts such as constructs, models, methods and instantiations [31]. These artefacts are also the outcomes and contributions of the research process. In case of this thesis, the outcome are the IT artefacts described in the statement of purpose above.

The outcome system should be assess as an archive system, together with the preservation of private digital documents. Consequently, various related knowledge background should be analyzed: design and functions of archive system, data format and content archive, archive value assessment and extraction, the requirements associated with private documents, the interface to collect, describe and deliver archive, the appropriate technology to implement, etc. These are the data need analyzing to solve the research question.

## 1.3 Content outline

The rest of this thesis is organized as followed: the related literature, technologies, and some important related terms and concepts are covered in the chapter 2, The design of the system and its implementation guidelines are introduced in the chapter 3. The prototype is covered in the chapter 4. Result summary, findings and contribution, and future research directions are discussed in the chapter 5.

# 2 Research background

## 2.1 Literature overview

Various literature and information sources are used the basis for the development of the system. This section will introduce them with the overview of relevant content, and details of important concepts and terms are discussed in other sections in this chapter and throughout the whole thesis.

### 2.1.1 Ownership and Copyright

Personal documents are intellectual property, which are covered by the Media Ownership Act [33]. In addition, the creation and sharing of digital archive involves activities covered by Copyright Act [9]. Firstly, the Norwegian Legal Deposit Act [11] is to ensure that documents containing generally available information are deposited in national collections, so that these records of Norwegian cultural and social life may be preserved and made available as source material for purposes of research and documentation. A document is made available to the public when copies of the document are offered for sale, hire or loan, or when the document is distributed in other ways outside the private domain, or when information contained in the document is made available outside the private domain by means of presentation, showing, broadcasting, on-line transmission, and the like. The Act does not cover the private documents for archiving purpose.

Secondly, the Norwegian acts and regulations related to Media and Culture which are published on the website of the Norwegian Ministry of Culture [12] are also associated with the purpose of the NPA system. The important documents included in this web page are the Copyright Act, The Media Ownership Act, guidelines for interventions pursuant to the Media Ownership Act, Regulations relating to the return of stolen and unlawfully removed cultural objects, Regulations relating to a prohibition against the export of cultural objects, and a link to a collection of translated Norwegian Legislation in the Library of University of Oslo. The Copyright Act contains 61 items in 9 chapters relating to copyright in literary, scientific and artistic works and to protect the ownership rights of authors and artists. The purpose of the Media Ownership Act is to promote freedom of expression, genuine opportunities to express one's opinions and a comprehensive range of media.

The article "Copyright issues relevant to the creation of a digital archive: a preliminary assessment" (J.M Besek, 2003) [3] is another relevant document which describes the copyright's rights as a set of right for making copy, creating derivative, distributing, performing and displaying to the public of an object or document. The author also mentions the copyright requirement, mandatory deposit, copyright ownership, the debates on the ownership of electronic rights, discussion on unpublished work, the Digital Millennium Copyright Act and international issues related to copyright, the article is ended with the summary of different approaches to acquire a work for digital archive and proposes contract constraints for each of them. Although the law mentioned in Besek's article (2003) [3] is the copyright law in the United States, it also has similar aspects in Norway.

The set of public licenses promoted by Creative Commons [43] are the other important documents which "give the public permission to share and use" intellectual works. Because the licenses are non-revocable, the system can create the license agreement between it and its users based on Creative Common licenses, so that it can use the documents for long-term preservation and sharing.

These documents are good references for creating a copyright terms and license agreement for people who submit their own documents to the archiving system, as well as the proper methods to form the archive access and backup policy that complies with the copyright act in Norway.

### 2.1.2 Archive system and long-term preservation

Lazinger and Tibbo [26], Borghoff [4], Giaretta [18] cover various important terms and concepts in digital preservation. The concept metadata are all mentioned in these literatures as the central point for creating digital archive. Giaretta [18] analyzes the different classifications for metadata based on (1) understandability, (2) origins, context and restrictions and (3) the way in which the data and "metadata" are grouped together. Different ways of classication for digital objects are also pointed out. [18] also has a long list of useful references which not only support the content but also provide the readers more materials to have a deeper understanding. Both Borghoff [4] and Giaretta [18] include the reference model for Open Archival Information System (OAIS)[6] that serves as general framework for digital archives, together with the process model for a deposit into the Deposit systems for electronic publications (DSEP). This model will be analyzed in detail in next section.

Other issues discussed in the three books [26, 4, 18] include threats, maintenance, migration, emulation of digital documents. In addition, Borghoff [4] introduces popular software system for archiving, Lazinger and Tibbo [26] reviews various archive projects in many countries, and Giaretta [18] discusses various tools and practices that used in digital archive system, including the financial strategy and cost analysis.

The depositing phase of documents requires trust from both users and the system. In the report of Groven [19], the trust strategies used in capturing, validating, reformatting digital material, and adding new description and metadata are summarized. Building trust in the initial capture is the most critical phases. The report covers the methods to establish, enhance and assess the trustworthiness of digital records, which are very useful to set the list of criteria to establish trust on the digital document submitted by user so that it can be accepted as an archive in the system. It also gives the methods of preserving trust during the archive management processes.

### 2.1.3 Personal archive and collection

Identifying various personal documents and archiving needs of them are covered by [28, 32, 10]. These authors the growing trends of personal documents in digital era and the urgent need for personal archiving system. Various types of personal documents, issues and challenges in managing personal digital collection focusing on born-digital material are discussed [28]. The author also points out the opportunities of using them for reproducing the lives of individuals. The book shares the thinking of ten authors in the various aspects of the management of digital information. It suggests the way cultural institutions can capture the new forms of documentation and how individuals could better manage digital information.

Marshall [32] analyzes the way of thinking of non computer expert people on preserving their personal documents. After pointing out the their assumptions, the author introduces the four issues of personal digital archiving and suggests the coping strategies to these issues. The introduction used in the article is easy to understand for many readers, whether expert or non-expert. The system being built in this project should also utilize such way of guiding reader to help the users become aware the problems of protecting their personal documents and do the right way for preserving them.

Marshall [32]'s book is divided into two parts. In the first part, the author introduces the naivety of people who are not computer expert about their personal documents. They think that backing up the system is the same thing as a long term archive, using free email with attachments to create archive, using social media sites to store photos and videos, writing important files to external storage is the ways for long-term preservation, and copying all data from old computer to new stronger computer. They also do not worry about the use of their documents later. They are self-confident on their ability to assess their own things so they don't need to add any metadata to their documents. The author claims that the reason leading people think in that way is that they have four assumptions in their heads. These assumptions reveal four challengers for personal digital archiving: digital stewardship, tracking distributed storage, detecting asset value, and long term access. In the part 2, the author examines the four challenges and discusses promising technological directions and requirement for each. By answering for questions based on the challenges, the author proposes the solutions for what to keep, where to put, and the methods of maintaining getting back the documents.

In addition, Cox [10] introduces the new trend of becoming archivist for each individual citizen. In the Internet age, with the emerge of many form of personal documents in enormous amount such as blog, email, self-recording video, web pages, together with the traditional one like letters, diaries, old family albums and even desks, pens, inks, and books to touch the past. It suggests the new kind of assistance for individual to maintain personal and family archives, together with more protection on those of special or extraordinary significance because there any many issues that can endanger the creation and maintenance of personal and family archives. After the consideration of the wide range of documents and objects type in personal archives, the author also gives the assessment for the future of personal and family archives. The discussion of different personal documents type in the book [10] is useful to identify the digital objects to be preserved in the building archive system. For example, the author points out that people are searching for the traditional letter, diary, photos, maps and other personal document support personal identity and everyday use like ATM card, passport, birth certificates, driving license, etc., to digitize them into digital form, and then build their own archive to keep these artifacts of nostalgia. These point out the need for sustainable solution to preserve such personal documents in long-term basis.

## 2.2 Digital preservation

### 2.2.1 The OAIS model

the Open Archival Information System (OAIS) [6] is a referential model for any long-term digital information preservation system. It does not specify any design and implementation. Instead, it defines basic archival terms and concepts, elements and processes needs for long term digital preservation and access, and full range of functionns of an archive system.

An OAIS Archive preserves information for a Designated Community to use [6]. A Designated Community is defined as “an identified group of potential consumers who should be able to understand a praticular set of information”. This definition also means that the preservation function of the archive is successful if its information is understandable by its Designated Community.

#### The OAIS model environment

The environment surrounding an OAIS archive is illustrated by the following figure 2.1:

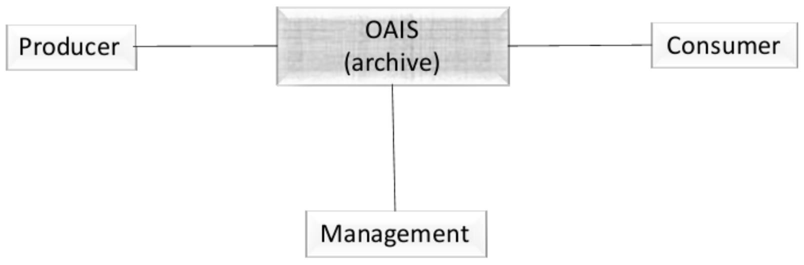

Figure 2.1: Environment model of an OAIS
(Source: CCSDS 650.0-M-2 recommended practice [6])

There are three entities surrounding an OAIS archive: *Producer*, *Management*, and *Consumer*. *Producer* are users or client systems that provide information into the *Archive*, *Management* are the people who set the overall pocily to the archive, not the people involved in day-to-day Archive operations (which includes within an administrative functional entity in the archive).

The interaction amongs these entities are depicted in the figure 2.2

*Producer* provides a Submission Information Package (SIP) to the *Archive*. The *Archive* checks the SIP toward admission policy, if satisfied the *Archive* will create the Archive Information Package (AIP) from the information in SIP together with newly created preservation information. The AIP will be transfered into an Dissemination Information Package (DIP).

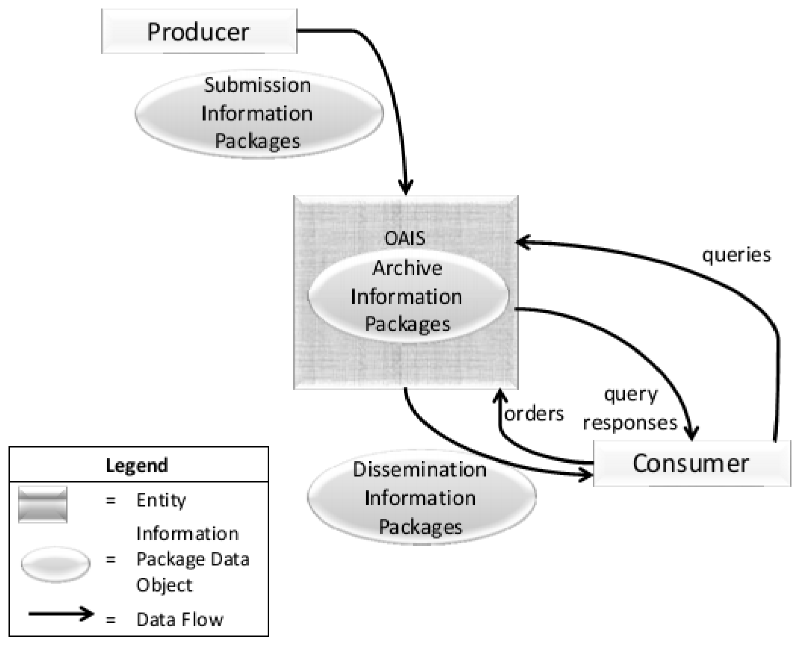
Producer
Submission Information Packages
OAIS Archive Information Packages
queries
query responses
orders
Consumer
Dissemination Information Packages
Legend
= Entity
= Information Package Data Object
= Data Flow

Figure 2.2: Archive External Data
(Source: CCSDS 650.0-M-2 recommended practice [6])

### Functions of an OAIS archive

The following figure 2.3 show the basic units and functions within the *Archive*:

The *Ingest* unit receives the SIP, produces the AIP into the *Archival Storage*, the *Access* unit get the AIP from the *Archival Storage* to produces the DIP to response consumer's queries. The descriptive information from SIP also get through *Data management* unit to improve the *Access* unit. Within the *Archive*. The *Management* entity administrates the NPA system corresponding to the preservation plan agreed as with *Producer* and *Consumer*.

### The Archive Information Package

The AIP is the essential data entity of an OAIS archive. The structure of an AIP is shown in the figure 2.4

There are many concepts and units shown in the structure of an AIP. The Package Description describes the AIP by providing the information that used to search for a specific package. The packaging information identifies how the information is stored within an AIP. The AIP contains the *Content Information* and its *Preservation Description Information* (PDI). As shown in the structure, many other information objects are packed within the package together with the data object itself. These different types of additional information objects provide more information about the data object, e.g. "data about data" or "metadata". There are three types of metadata are shown in the illustrating figure 2.4: the representative information is the metadata that provides the

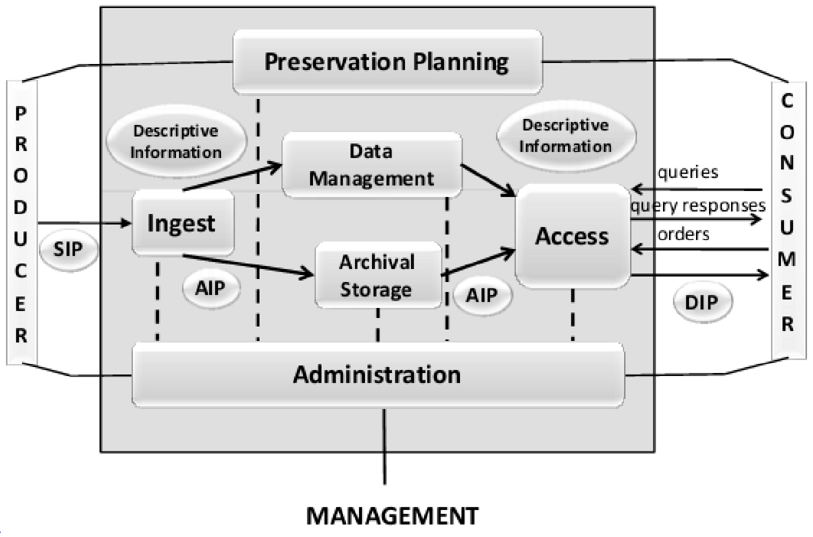

Figure 2.3: Archive Functional model
(Source: CCSDS 650.0-M-2 recommended practice [6])

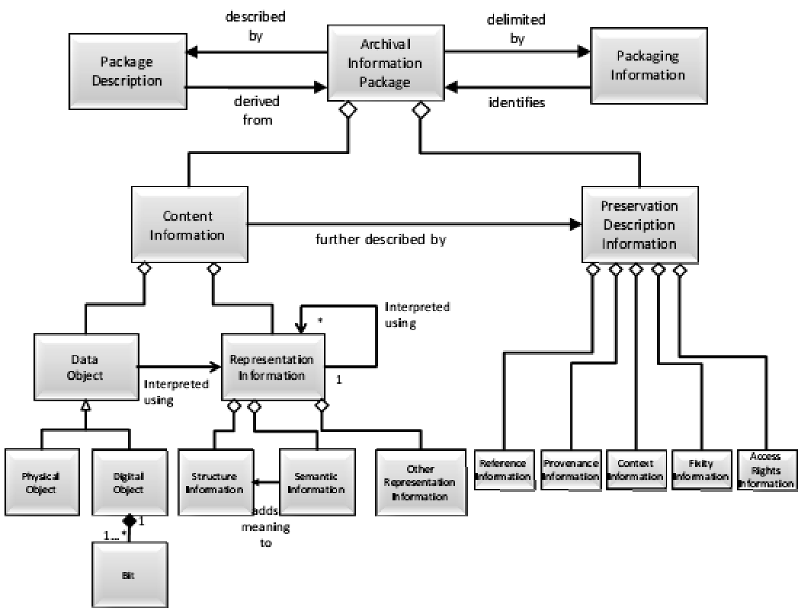

Figure 2.4: Archive Information Package
(Source: CCSDS 650.0-M-2 recommended practice [6])

"information that maps a Data Object into more meaningful concepts", the PDI provides the origins, context and restrictions that used to know "what and where the digital object came from", the packaging information provides information about the way "to bind and identify the components of an Information

Package". This classification of metadata is proposed in [18]. Details of the definition and examples of these metadata can be found in chapter 3 of [18].

#### Other basic concepts

- Data: a reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing.
- Digital Object: an object composed of a set of bit sequences.
- Long Term: a period of tim long enough for there to be concern about the impacts of changing technologies, including support for new media and data formats, and of a changing Designated Community, on the information being held in an OIAS. This period extends into the indifinite future.
- Long Term Preservation: the act of maintaining information, Independently Understandable by a Designated Community, and with evidence supporting its Authenticity, over the Long Term.
- Authenticity: the degree to which a person (or system) may regard an object as what it is purported to be. The degree of Authenticity is judged on the basis of evidence.
- Independently Understandable: a characteristic of information that is sufficiently complete to allow it to be interpreted, understood and used by the Designated Community without having to resort to special resources not widely available, including named individuals.
- Designated Community: an identified group of potential Consumers who should be able to understand a particular set of information. The Designated Community may be composed of multiple user communities. A Designated Community is defined by the archive and this definition may change over time.

These concepts are defined in the book [18]. These fundamental concepts are used and affect throughout the NPA system, thus quoting the definitions are needed.

### 2.2.2 Digital object, file type and format

#### Digital Object classification

Different classifying methods of digital objects are identified by Giaretta [18]. A digital object can be a simple object which is treated as a whole, or a composite object which is a collection of simpler components. For example, an Hypertext Markup Language (HTML) document can be seen as a composite object which embeds text, images, and video within it. On the other hand, an image file can be treated as a simple object. However, this classification is not completely clear cut.

A digital object can be rendered or non-rendered. The rendered objects are those that can be processed by some software or devices to generate a rendering for users to interprete what they see/hear/feel/taste. In reverse, a digital object that requires the users to know what the content means to do further process are non-rendered. In fact, any digital object should be understandable after they are properly processed, but the non-rendered object requires more processing steps and more knowledge from the users to understand, rather than the direct way through users' senses. For example, a file which stores GPS positions in longitude and latitude coordinates might be seen as a non-render object, because when users look at these numbers, they cannot understand unless they have more knowledge about the coordinates and the position are shown in a three dimensional (3D) map.

Another classification is static or dynamic object. The static ones are unchanged as bit sequences such as images, documents, video when software or hardware process or render it. The dynamic ones are naturally changed over time such as database files or a collection which has more files to be added in the future.

Digital objects can also be active or passive. A passive one are used by other application to do something, and the active one does something such as an archive system preserves digital documents. This is also not a clear cut classification.

These above categorizations are not mutually exclusive.

### Document type and file format

Digital object are encoded as bit sequences and stored in form of files. There are many ways of classifying a type for digital object, but the most popular way is based on how the object is used. Similarly, the method to identify and classify document type should be mainly based on how the documents are used in practice, such as documents of image type are seen as still images, video type are seen as motion pictures, audio type are heard, etc. There are several way to encode each document type, and a standard way to encode a document is *file format*. In the other words, a document can only be rendered or processed for its intended usage if its file format is known.

### Open File Format (Open Format)

As discussed previously, file format (or format) is the way to encode a digital document. The series of bits of a digital document will not contain other meaning but the mathematical meaning of the bits themselvies, unless its file format is known. A file format can be either open or proprietary. A proprietary format is the format protected by copyrights, patents, royal-fee or other restrictions, which means that the specification and usage of proprietary format are restricted. In contrast, an open standard format is the format that has specifications published widely for any person can use and implement without any monetary cost and copyright and patent restrictions. Open standard formats can be defined as file formats that encode content in such a way that it will be freely available in the unforseenable future [45]. Because the specifications of open formats are available and they can be used without cost, open formats are the preferred formats for long-term preservation.

### File format detection

The process to identify the format of a file, or file format detection, is archiving effort. The purpose of detecting format of a document in archive system is straightforward, e.g. to be able to transform it back to its intended use. Currently there are different methods to identify the format of a file:

- Using file extension: there are standard extensions for each file type such as jpg, png, bmp, svg for image files; ogv, mp4, wmv for video files; mp3, wma, oga for audio files; gpx for GPS tracking files, etc.
- Using internal metadata or "magic" bytes: inside some files, there are some information or bytes that indicate the format of the file. Detecting file format can be done by reading these information or special bytes. For instance, Richard Ogley includes a list of magic bytes and how they are stored in a file [39], that can be used for format detection.
- Reading the whole file content: many different algorithms are used to identify file type are described in the articles of Li and Wang [29], Amirani [1], Karresand and Shahmehri [25].

### Re-use of Digital Objects

The purpose of preservation is to keep digital objects usable and understandable by a Designated Community [18]. The usability and understandability of digital objects obviously depend mainly on the representative information described in 2.2.1. The concepts are different when applied on different objects. For example, an image object might be usable if it can be rendered as an visual object seen by human users. A source code of a software is usable if it can be compiled, run, and do the same job as the one which is also created at the time of depositing, and so on.

### Migration and Emulation

In the previous section 2.2.2, the function of an archive system is to preserve digital object usable and understandable. However, technologies are changing, and the old software or hardware system that support the preserving digital objects might disappear. In that case, there are two different approach to keep the digital objects remain usable and understandable. The first approach is to transfer the old objects into new objects that can be used in the new system with the same intended purpose, this approach is called *migration.* The second solution is to create a "virtual" environment of the old system for the digital object to be used, rendered, or displayed, etc. The second approach is called *emulation.* Which strategy is used for keeping digital object usable and understandable when technology changes is depend on the change and other practical aspects. The problem of archive accessing and different solving approaches as emulation and migration are discussed in Borghoff's book [4].

## 2.3 Metadata

Metadata is the "data about data", so basically metadata of an archive will tell what it is, how it was created, how it is structured, how it is managed, etc.

Metadata provide users and computers more information about the data object in AIP. Therefore, metadata is the essential part of the data object in the NPA system, without metadata the data object might become unusable and incomprehensible. The more metadata about the data object, the more usable and understandable it becomes. How much metadata is enough? Or how much metadata should be provided to make the data objects usable and understandable by the Designated Community of the archive? The amount of metadata needed depends on the Designated Community's background knowledge. the NPA system should ensure that any data objects preserved are provided with enough metadata.

### 2.3.1 Metadata Categorization

In the previous part 2.2.1, a way of metadata categorization based on different entities contained in an AIP are shown. There is also a popular classification based on the function of the metadata from National Information Standards Organization (NISO), which identifies three main types of metadata: descriptive metadata, administrative metadata, and structuring metadata [37]. Descriptive metadata provides the information that is used to identify amongs other objects or documents of the same type, such as object title, owner, subject, etc. Structural metadata describes how the components of composite objects are combined. Administrative metadata are the information used to manage digital objects, such as access permission, file format, date of creation. In addition, two subsets of administrative metadata are sometimes listed as separate ones: rights management metadata that contain information of intellectual property rights, and preservation metadata that needed to perform preservation activities [37].

### 2.3.2 XML and JSON

Extensible Markup Language (XML) [5] is a class of documents that conforms Standard Generalized Markup Language (SGML) [ISO 8879]. An XML document must follow all syntax rules defined in XML specification [5], in this case the document is well-formed, otherwise it will not be considered as an XML document. XML contains storage units called entities, or elements. Each elements has a number of attributes, and content. Element's content contains other elements or data in form of text or binary. An XML schema describes the logical structure of an XML document, e.g. describes the elements that might exist in the XML documents, which attributes they have, and what elements they might contain. An XML is said to be valid if its content satisfies its schema, which is declared using Document Type Definition (DTD), XML Schema Definition (XSD), or other schema description language. The XSD itself is also an XML document, thus it is preferrable to be used to define schema for XML document over DTD. The specification of an XSD file is given at [14]. A metadata can define the set of information contained in its corresponding XML document using a schema definition language. Such definitions of many standard metadata used for archiving purposes are given in XSD form. Eclipse, a popular Integrated Development Environment (IDE) for software development, supports visualizing the structure described in XSD file.

JavaScript Object Notation (JSON) [23] is a lightweight data-interchange format based on a subset of ECMA script (or Javascript) Language specification [20]. In JSON there are two types of structure: a collection of name/value pairs and an ordered list of values. They are considered as object and array, the universal data structures in almost modern programming languages. Standardized together with Standard ECMA-262 3rd Edition - December 1999, JSON is one of serialized forms of data structure objects in many C-family programming language such as C, C++, C#, Java, Javascript, PHP, Python, etc. JSON is encoded in text format, human-readable and writable, language independent. JSON is optimized to encode data, and it is easy for machine to read and write JSON. However, JSON does not support binary data and not extensible, thus JSON can only be used as a data exchange format between applications, and cannot be used to exchange content of digital documents (because they can be in binary form).

On the other hands, XML can also be used as a serialized form of data structure objects in many programming languages. The advantages of it over JSON is that it supports binary data such as the content images, video, software binary codes, etc., thus XML is the better document exchange format. XML is a well-known and widely-adopted standard language for describing data structure. Therefore, XML is the preferrable standard in archiving system to store digital documents and their metadata. The introduction of widely used metadata for arching purposes are given in the rest of this section.

### 2.3.3 Dublin Core

Dublin Core is a standard metadata for describing, discovering and managing resource. Dublin Core contains a set of general terms and vocabularies which are encouraged to be used with the same meaning across system, organization and communities. Dublin Core can also be employed to cooperate with Resource Description Framework (RDF) for machine-understandable semantic description. Dublin Core is maintained by Dublin Core Metadata Initiative (DCMI). The structure, usage and implementation instruction of Dublin Core, as well as the advantage of using a standard metadata are available at DCMI's homepage.

### 2.3.4 METS

The definition of Metadata Encoding and Transmission Standard (METS) is given in its homepage [34]:

> The METS schema is a standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library, expressed using the XML schema language of the World Wide Web Consortium. The standard is maintained in the Network Development and MARC Standards Office of the Library of Congress, and is being developed as an initiative of the Digital Library Federation.

METS is a structuring metadata, which not only describes the structure of digital documents, but also describes the whole archive package by embeding or linking (refering) other metadata. The structure of a METS schema is given in the XSD file. The structure of the root element of a METS document (the <mets> element) are given in the figure 2.5.

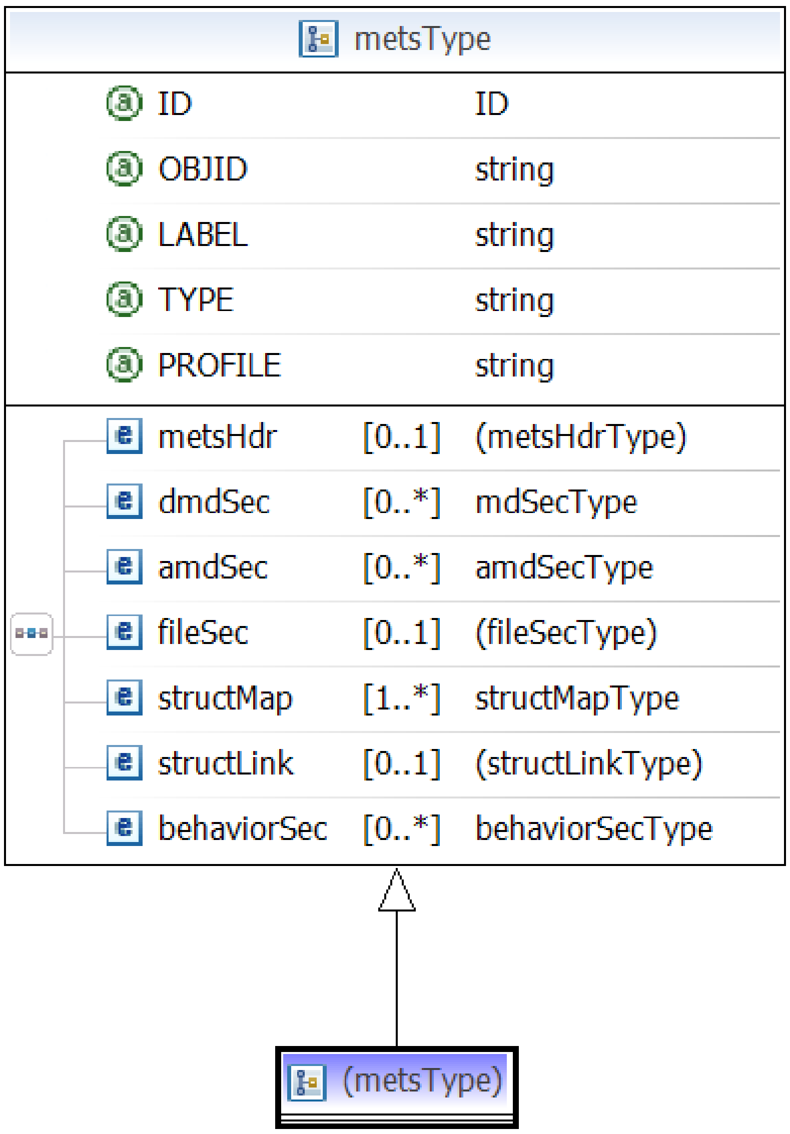

Figure 2.5: Structure of a METS file

The essential element in a METS document is structural map (*structMap*), which contains a root *div* (division) element. The root div can have unlimited number of child elements, and this feature is also applied to all of its child elements. This structure map is obviously capable to describe any kind of structure of the document's content. The METS document can also describe (and even embed) the files contained in the archive (within *filegrp* element in *fileSec* element). METS can also embed or refer to descriptive and administrative metadata via the elements *dmdSec* and *amdSec*. The *structLink* is mostly used to describe the structural map of a web archive. The *metsHdr* and *behaviorSec* stand for METS Header and Behavior Section. The detail usage instruction of METS can be found in its homepage [34].

### 2.3.5 Other metadata used for archiving

In the METS schema, the element *dmdSec* and *amdSec* are used to embed descriptive and administrative metadata. Metadata Object Description Standard (MODS) [35] and Dublin Core are often embedded in the *dmdSec* elements. There are four types of administrative metadata: technical metadata, right metadata, digital provenance metadata, and source metadata. NISO Metadata for Images in XML (MIX), Technical Metadata for Text (TextMD), AudioMD and VideoMD are the popular schema used for describing technical information of images, text, audio, and video documents. ALTO is also a technical Metadata for Optical Character Recognition (OCR). PREMIS (Preservation Metadata) is ametadata needed to support the long-term preservation of digital materials. More useful information about these metadata and other popular metadata used in archive are available at the Library of Congress standard homepage [30].

These above metadata are often associated externally with the digital documents. There are some metadata are embedded inside the document files. For example, an audio file might include the ID3 metadata, or an image file might have the embedded Extensible Metadata Platform (XMP), Exchangeable image file format (EXIF) and IPTC photo metadata [21] inside, inside a video file should also have its title, duration, bitrate, etc. As these metadata are embedded to the document files, they are preserved together with the documents (if the documents are preserved as authenticated as it should be).

# 3 System design

## 3.1 Design goals, requirement and challenges

Normally a document is archived by an archivist who also has the responsible to check the quality of the archive such as document content, metadata, copyright, authenticity, etc. An archive system normally accepts a small set of documents. The archive are often published to a certain set of consumer corresponding to the agreement between archive owner and archive system, and the content of the archived document might not be changed. The archive systems that operate in this way or similar are "normal" or "popular" one, and the OAIS referential model and many software preservation systems introduced in [4] can mitigate the problems in building such archive systems.

However, the NPA system has many different characteristics that must be addressed and it needs a different design. It is a public-available, shared archive system, where each depositor is guided to do the tasks as archivists of their own personal documents. The system has to manage many different types of document, and the intervention and management effort of the system administrator should be minimized because the amount of documents are huge. The value-creation process is motivated by all users. The system must provide the appropriate tool to encourage each type of user to do his job well. In the other words, the system must provide the tools for its depositors to create the as good archive of their personal document as the archivists do, for its consumers to improve the archive quality, and for the administrator to define and control the ways of other users doing their own tasks while the administrators are not allowed to access the documents and the administrator-intervention needs of other users must be minimized.

The design for NPA system is also based on the OAIS referential model with similar architecture and functions, but some part of it are changed and some new appropriate components and concepts are brought into the design to make it works in the different way described above. In the other words, the goal of this thesis is not to design a totally new archive system with different architecture and functions from the "normal" one above, but rather focusing on the differences between the NPA and the "normal" ones, as well as the solutions for those diferrences.

## 3.2 System Overview

The prototype system is an approach toward the referential one according to a guideline. The prototype and roadmap are presented in the chapter 4.

The rest of this chapter is organized as follows: firstly, the overall design of the whole system is presented, followed by more details that explain the overall desgin. A part of the details are presented in the form of social cultural guideline and technical guideline to build the system because they are at higher abstract level and can be considered as recommendation.

The figure 3.1 shows the overall design of the system, focusing on the archive-related functions.

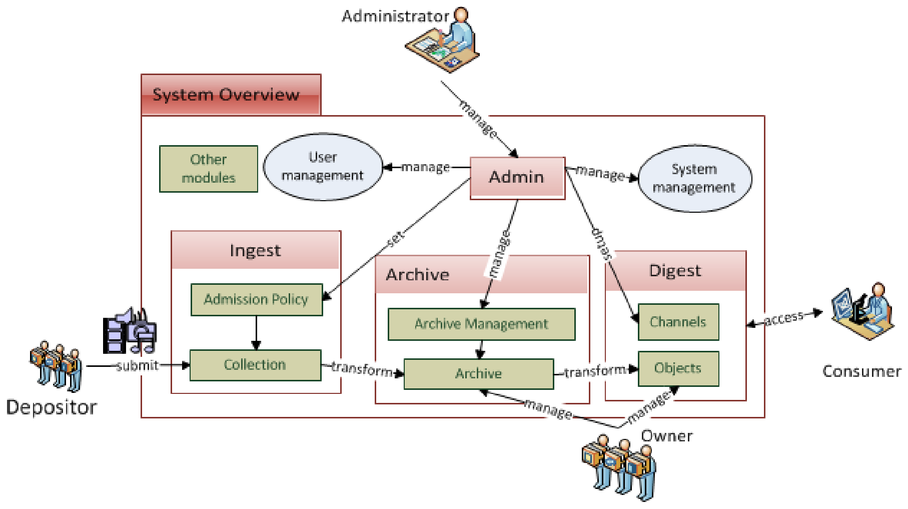

Figure 3.1: System overall design

The design of NPA system is based on the OAIS referential model. It has a set of global modules and three sub-systems: ingest, archive, digest. The system has 4 types of users: administrator, depositor, archive owner, and archive consumer. The administrators manage the system through admin module. The whole system has an front-end interface that serves the interaction between the whole system and its users, including user registration and authentication. The interface should also include the introduction to NPA system, system policy, frequent asked questions, etc.

The figure 3.1 rather shows the basic flow of the document deposited for initial understanding instead of the actual architecture of subsystems. The administrator set the admission policy, archive management policy, and setup digesting channels. The depositors submit their documents as collections through the ingest subsystem according to admission policy. The accepted collections are transformed into an archive and stored in the archive subsystem. Archive owner can manage their own archive, and they might decide to publish their archive to consumer via a set of channels in the digest subsystem which are also established by administrator.

## 3.3 User

The guideline in this subsection is the foundation to create system policy, archive publishing policy and user agreement policy. The system should ensure that all users comply with the system policy before accessing the system.

### 3.3.1 Depositor

The only eligible users that allowed to deposit their documents into the archive system should have close associations with Norway. It is suggested that any people who has a valid Norwegian identification (ID) number (Fødselsnummer) can deposit their documents because they live or have resided in Noray. The information of these people who have the number is stored in National Population Register (Folkeregistrering) [16]. The structure of this eleven-digit number is described at the Regulations for national registration [42]. The Folkeregistrering also stored information of the number's bearer's death, which might be used for processing inheritance feature of the system (see dødfall [13]).

### 3.3.2 Administrator

While administrator can perform various administrative functions, the system should prevent them from violating the privacy policy set by archive owner. The NPA system are expected to have different types (or roles) of administrator. Each type of administrator will manage a specific function in the system. The administrators for the system should be authorized people from public authorities of Norway. It is suggested that the National Library and Ministry of Culture should be among such authorities because of their role of preserving cultural and historical heritage of the country. Details of each type of administrator will be discussed later.

### 3.3.3 Archive owner

The depositors are the original owners of the deposited archive. However, the archive owner may give their ownership to others. For instance, the ownership might be transferred to the legal inheritors in case the owner dies, or the owner might grant the ownership to the public domain.

### 3.3.4 Consumer

Consumer are the user who have access to the digest system. They might be researchers, students, or individuals who want to access the documents published by archive owners. The consumers can also enrich the archive with more suggestion to the system and to archive owner. Details of the value enrichment are discussed at 3.7.5.

It is suggested that the use of all users account should be free of charge. Administrator's accounts should be created by an authorized administrator. Other users should have suitable accounts to access to the desired functions.

### 3.3.5 User registration, authentication and authorization

User registration can be done via front-end interface of the NPA system. Administrator should set the user policy, which can vary among each type of user.

To access the documents published in the digest subsystem, a person need to have a consumer account of the NPA system. He might be required to share a number of information such as name, email address, occupation, researching purpose, etc., together with evidence for these required information. Another way of granting access as consumer to the system is integrating other accounts from other authorities such as universities and colleges in Norway using a common login scheme like Feide [15].

For depositor's account registration, it is recommended that the account eligibility need additional verification. Firstly, a depositor's account should be in a a valid Norwegian ID number. The verification method can be either address verification by sending an confirmation code to the address registered in National Population Register, or by using a form of electronic signature like BankID [2]. In addition, because the NPA system is suggested to be a public service available to Norwegian resident, it can be integrated into the shared login solution like ID-porten/MinID. In that case, the depositor do not need to register for an account directly in the system. It is suggested that a depositor account should be used as an archive owner account, and have the same rights as a consumer.

For archive owner account, obviously the original owner of an archive should have a depositor account. However, the inheritor might not have a valid Norwegian ID number. In this case, it is suggested that the inheritor should register for a consumer account first, and then the ownership's access to the archive can be granted to his consumer account.

### 3.3.6 User access summary

The guideline for user access is illustrated in the figure 3.2. The administrator

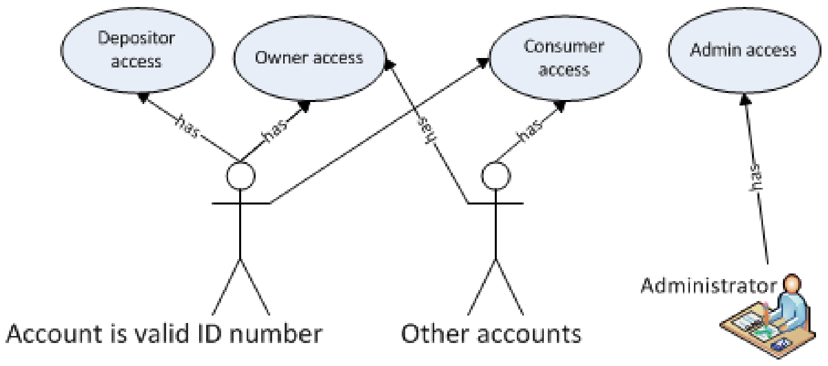

Figure 3.2: User access illustration

should have admin access to the system using the administrator account. The users that have accounts in form of a valid Norwegian ID number (depositor) can deposit documents to the ingest subsystem (depositor access), manage their own documents (archive owner access), and access published documents. The other users have the same rights as depositor except the depositor access. In the other words, other users may have the owner access if they are given or inherit an ownership.

### 3.3.7 Ownership, inheritance, copyright

Personal documents are intellectual properties, thus the document archives are inheritable. When the owner of an archive dies (The event are determined from the National Population Register), the ownership should be transfered to his or her legal inheritances.

The person who deposits his documents should hold the ownership and copyright associated with his documents, e.g. the depositor must be the author of the documents. According to the Copyright Act [9], the archive can be kept privately in at most 70 years after the last author's dead. Consequently, documents deposited into the system should be turned into public domain at most 70 years after the deposior's death. The owner can also decide the time these documents to become published earlier.

## 3.4 Document Profile

### 3.4.1 Concept Definition

Document profile (DP) is an important concept used in the NPA system. A DP describes a class of digital objects that have some specific characteristics in common. DP defines the specific criteria to assess value of the digital objects described, and the methods to preserve and share them.

Various characteristics of digital object are identified at 2.2.2. DP is a tool to manage various types of digital objects for both administrator and archive owner. The following information might include in a DP:

- The characteristics of its objects.
- The structure of its object.
- The metadata used to describe its digital object, and how metadata are associated with the digital object.
- The admission policy of its object.
- The interfaces for updating (include creating) the object, it also ensures that the object satisfy admission policy, and the interfaces for re-using (or sharing, viewing, displaying, etc.) the object.
- The preservation planning (including update policy) for its objects.

Because the most important role of this system is preserving the documents for long-term preservation, the suggested archive update policy is to prevent documents and their metadata (i.e. the archive objects) to be deleted and changed, so all changes made to the archive objects should be recorded. In the other words, once a document is accepted as archive, it will be preserved permanently, and the owner might not delete or change its content or its metadata, unless the update policy permits to do so (in that case, all the old versions are still kept).

### 3.4.2 Document Profile Examples

The concept of DP defines the different ways to manage different type of digital document in the NPA system. Some of the below DP example *might not have a practical use*, such as the *Single File Profile* and the *Image Profile* because they are too simple. However, they are also included to clarify the DP concept.

### Single File Profile

This is the profile for a single file as a digital object in the system.

- Characteristics: simple object, static, rendered or non-rendered, and passive or active.
- Structure: an object is a file.
- Metadata: each should have a filetype attribute, and each filetype has a different set of metadata associated with it.
- Admission policy: each filetype has a different set of minimum information which must be filled.
- Interfaces: enable user to upload file, editing metadata, download the file and view its metadata .
- Preservation planning: Each filetype (and format) has a different migration plan set by administrator. Update can be made on metadata only and all versions are kept.

### File Combination Profile

An object in this profile is a set of directories and objects in the *Single File Profile* (files with metadata).

- Structure: made of *Single File Profile* objects, stored in normal directory tree structure for files.
- Characteristics: composite, static and passive, it can be rendered or non-rendered.
- Metadata: the sum of all metadata used to describe each type of files included, the metadata might not only include standard ones but also user-defined ones.
- Admission policy: each individual child object should satisfy its admission policy. The object must not be empty.
- Interfaces: the users should be able to edit each child object and the directory structure through the updating interface, the re-use should show the file together with its metadata.
- Preservation planning: the file content might not be changed, only metadata are can be changed and all changes are kept. The migration should be made based on preservation plan of each *Single File Profile* object.

### Image Profile

This is the profile for a single image as a digital object in the system.

- Characteristics: simple object, static, rendered, and passive.
- Structure: an object is an image file.

- Metadata: XMP (external or internal), EXIF (embed inside the file), Dublin Core and IPTC metadata used to describe its digital object.

- Admission policy: the following Dublin Core terms must be given: title, subject, author.

- Interfaces: enable user to upload file, editing metadata, visualize the image.

- Preservation planning: The content of the image file might not be changed by owner. The images format might be Digital Negative (DNG) which embeds the original proprietary raw file. The original format specfication might be released for use for free, thus it is then supported by the system, and the images might be migrated into that format. Update can be made on metadata only and all versions are kept.

### Image Album Profile

The DP *Digital Image Album* defines that an album has a name and a description. It is made of a number of *image* objects in *Image Profile*. In addition, all images must includes photographer's name, capturing date, capturing location, an assigned caption, and a description for it. All images must contain valid EXIF information. The minimum width and height of images must be 400 pixels.

The above description show various items in the profile:

- Characteristics: the digital object is rendered (by displaying its images and metadata), composite (and each image is a simpler component), static and passive.

- Structure: made of images.

- Metadata: EXIF, all information in minimum descriptive information, and the additional information provided by the owner.

- Admission policy: all minimum information should include, the image files must contains valid EXIF information, the size must not smaller that minimum size.

- Interfaces: provide the methods for user to create or update an album that satisfy the admission policy, i.e. verify file format, verify EXIF existence, verify that minimum metadata are given. The sharing interface should visualize the images in the album together with their metadata in a proper manner for human viewing.

- Preservation planning: because the images contained in an album is the objects described by *Image Profile*, each of them is preserved according to *Image Profile*. Update can be made on metadata only and all versions are kept, more child objects can also be added.

### Static Web Profile

This is the profile for a static website (which contain static HTML pages only).

- Characteristics: the digital object is rendered (by displaying its content), composite (containing mixed content of text, image, sound, etc.), dynamic and passive.

- Structure: a series of snapshots of a website. Each snapshot is made of a set of HTML files linked with its resource files.

- Metadata: Dublin Core and RDF. The metadata are embedded inside the HTML files.

- Admission policy: all files should be supported by the system, the content of each HTML file must have at least a meta tag which contains *name* and *content* attribute. All HTML files must be well-formed and valid according the *DOCTYPE* declared in each file. The title and description of the whole object must also be given.

- Interfaces: the editing interface enable user to make changes on the website, or retrieving file content from an Uniform Resource Locator (URL), it also has to verify that the admission policy as satisfied. The sharing interface should visualize the website as a list of snapshots and can also show the formatted content of each snapshot.

- Preservation planning: The content of the HTML files and the number of files might be changed, but the changes are not persisted into the old files, the new version is saved as a snapshot. Each snapshot is packed in a TAR file.

### Web Application Profile

This is the profile for a web application (a website with server-side script).

- Characteristics: the digital object is rendered (by displaying its content) or non-rendered (in case the application does something), composite (containing mixed content of text, image, sound, etc.), dynamic, and active or passive.

- Structure: A set of files, each file can either executable (source code of a server-side script) or not (HTML or image, sound, etc. which do not need a compiler or an intepreter to run), together with metadata of the whole object, and metadata for each source code file.

- Metadata: Dublin Core and RDF. The metadata are embedded inside HTML and source code files, the metadata associated with each source code file (stored externally).

- Admission policy: title, description, target environment, hardware requirement, install instruction, and description for each source code file must be provided, policy for other files is the same as the policy of *Static Web Profile*

- Interfaces: the editing interface enable user to update each file and its metadata (if any). The sharing interface should visualize the website and host the server-side script to run, or it can allow user to download all files together with all metadata (including the installing instruction).
- Preservation planning: The content of files and metadata might be changed, but all the old versions are kept, each file has an update history of its content and its metadata (so it is different from the snapshots in the *Static Web Profile*).

### Assembly Software Profile

An assembly software is a program written in a specific assembly language on a specific hardware. For instance, an assembly program written in the 8086 assembly language can run on a computer with 8086 processor. The program might get input from user, process the input to produce the output for user. Similarly, an assembly program written in the language for a 8052 micro-controller might only run if it is installed on the environment of that controller. By installing and running this program, the user can order a 8052 micro-controller system to do something according to program he wrote. The object in this type of DP is active.

- Characteristics: the digital object is non-rendered, composite or/and simple (might contain one or more assembly files, but there is only one program after compile), static and active.
- Structure: a set of source code files of specific assembly language, which is compiled into one software.
- Metadata: Dublin Core and user-defined attributes.
- Admission policy: the user must provide the title, author, description, target environment description, hardware requirement, hardware layout and connection, installing instruction, input, output or effects.
- Interfaces: the editing interface enable user upload files and update metadata, sharing interface should show or illustrate the result or how the software runs, enable user to supply input and show the output or effects.
- Preservation planning: Update can be made on metadata only and all versions are kept. The administrator should check for availability of hardware or target environment for the software to do appropriate preservation tasks such as migration or emulation.

### Generic Software Profile

- Characteristics: non-rendered, composite, static and active.
- Structure: a set of source code files.
- Metadata: Dublin Core and user-defined metadata.

- Admission policy: description for each source code file, title, description, target environment, hardware requirement, install instruction, structure, input, output or effects of the program must be provided.
- Interfaces: the editing interface enable user to change files and provide metadata, sharing interface should show or illustrate the result or how the software runs, enable user to supply input and show the output or effects.
- Preservation planning: Update can be made on metadata only and all versions are kept. The administrator should check for availability of hardware or target environment for the software to do appropriate preservation tasks such as migration or emulation.

### 3.4.3 The need for Document Profile

By showing the content of a DP, the purpose of defining the concept is obvious. The above shows there are different types of documents are contained in each DP. The complexity, abstract level, characteristics, metadata, admission policy, and other factors also vary from DP to DP. A DP object can contain the objects of other DP. When that case happens, the container might also override the specifications of its child DP (if needed).

The DP concept is the solution for the challenges discussed in 3.1. The metadata define the set of information needed from the users. The admission requirement ensures the content quality. The interfaces and preservation planning allow users to interact with the system for to create, update, share archive value. The DP is also a unified tool for administrator to define the ways for users to interact with archives in the system without administrator's involvement. How the DP concept is applied into the system will be mentioned throughout the rest of this chapter.

### 3.4.4 Specifying Document Profile

Four things must be included to specify a DP: Content and Metadata(s), Interfaces, Tools, and User Description as illustrated in the figure 3.3.

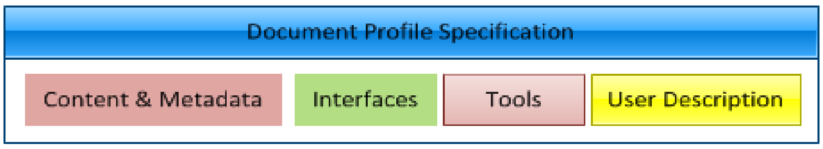

Figure 3.3: Document Profile Specification

#### Content and Metadata

The *Content and Metadata* specified which components and their characteristics contained in the DP, how the files and components in the digital object are being organized in the system, what information needed to describe the files and components and how these information associate with them. As discussed in the previous examples, the structure of the components object might follow the tree structure in File Combination Profile, or might be only in flat structure in Image Album Profile, or in cross-linking structure in the two Web Profile. The set of metadata used to describe files and components can be defined by Administrators, and they may different for each type of document. The recommended metadata for each type are discussed in the section 3.6.3. The different way to associate the files (or components) with their metadata are discussed in the section 3.6.7.

### Object structure

A digital object of a DP is stored as an archive package in the archive subsystem (see illustration in the figure 3.4). The archive package contains object's components, user metadata and system metadata. The *Components* of the object are files or other smaller components (which are also the objects of other DPs of the owner). The *User Metadata* are all the information collected from the user to describe the document. The *system metadata* contains all the information that the system puts into the archive such as the description of DP, the information of archive owner, date of submission, changes or publishing, etc. The system metadata put into the archive must be enough to enalbe the archive package to describe itself using the content inside it. This property is called "archive autonomousity", it is discussed in the section 3.7.4. During

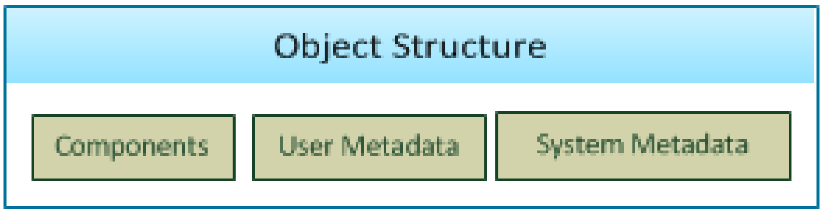

Figure 3.4: Document Profile Object's Structure

the ingesting process, these components of the object might not be grouped together. When the digital object is building in the ingest subsystem, it is called *Collection*. In the other words, the *Collection* might only have its files and their user metadata. When the collection has enough user metadata and is ready to be archived, the complete DP object will be built in the ingest subsystem and transfered into the archive subsystem. Similarly, during the digesting process, more components can be added into the object for better consuming. For instance, when an Image Album object is shown, the channel might generates some thumbnails of the images contained in the object for better performance.

### Document Profile Interfaces

The DP's *Interfaces* provide the means for users to interact with the digital object. The structure of *Interfaces* is visualized in the figure 3.5. Each DP should have *Component Interfaces* for modifying its child components. The child components can be either a set of files from users (for instance, file upload and delete *File Combination*, or the existing DP objects (for instance, add and remove *Image* objects from an *Album*). Through *Component Interfaces*, users should at least be enable to add and remove the components. The system might also provides the content editing interfaces for users to edit the content of each component directly from the system. DP interfaces must also include *Metadata Interfaces*, which show object's metadata, allow user to update user metadata, and save the changes into the object's metadata. The *Validators* component of DP's interfaces verify the object structure and all metadata given in a well-formed according to admission policy and update policy. The *Review Interfaces* allow users to review the *Object*, together with showing its *Changes*'s history. The object might be published, re-used or made changes by Consumers through *Consuming Interfaces*. In addition, some additional glsdp's *Tools* are needed during the depositing and preservation process of the digital object. One important tool of each DP is the archive package tool that is used to build the archive package from collections in the ingest subsystem by collecting all needed metadata and combine with the files and user metadata. The archive package tool is also used to persist the archive package into the archive subsystem's storage, unpersist it and update information, and transfer the archive package into a suitable consuming form in the digest subsystem. The archive package tool is essential for the DP to be used, and it is attached into the DP's Interfaces. Other tools such as file format conversion or migration tools can be added later.

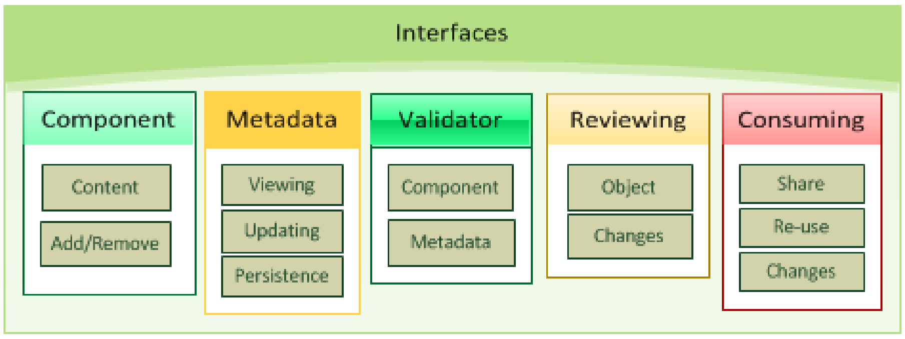

Figure 3.5: Document Profile Interfaces

### User Description

Administrators also need to provide *User Description*, which includes all the information needed by other users for depositing, updating and consuming the digital objects such as how documents are being preserved, the instruction to deposit, admission policy, update policy, preservation plan, etc. DP's interfaces are used in all three subsystems. They are used in the ingest subsystem by Depositor, in the archive subsystem for update archive metadata (and the archive files if allowed), and in the digest subsystem by Consumer to provide more information to the document. The User Description might also include the information for administrator the develop the DP, as the development involves many administrator's roles in the system. These different roles are discussed in the section 3.7.7.

## 3.5 System Architecture

The architecture of the system will be discussed and how the challenges mentioned in 3.1 are solved are also shown. In the architecture does not show users for simplicity, but their involvement will also be explained. The guideline to build the system are discussed in the rest of this chapter. The architecture is

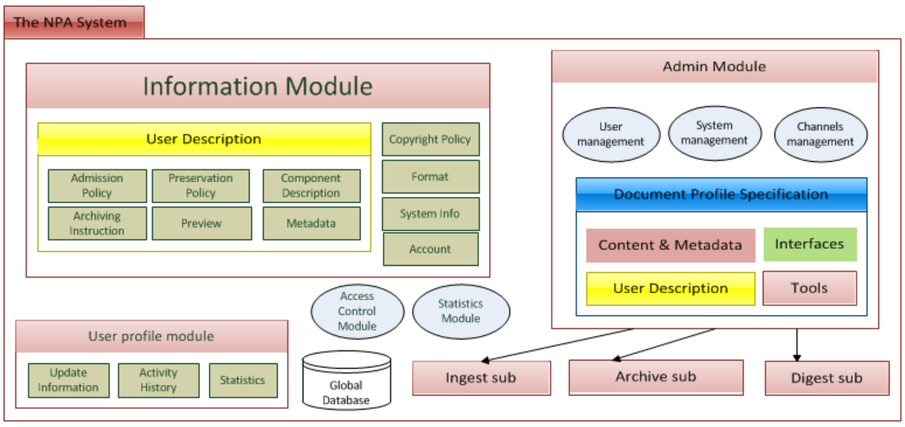

Figure 3.6: Archive subsystem

given in the figure 3.6. The system has an *Information Module* to provide all the information to users: copyright and ownership policy, file format support, information about the system, how to create an account, etc. A set of information is brought from the DP's *User Description* as described in 3.4.4. The *Access Control* module is used for ensuring access policy in the system, such as user registration, user login into the system and subsystem, and other access integration, authorizing access to documents and archive, etc. The *User Profile* module is where users can update their information and view the activity log, together with the statistic logs related to their documents (including depositing, owned, and consuming documents). The *Statistic* module collects the statistical information thorought the system and inside its subsystems and shows each type of user the appropriate information.

The *Admin* module is the tools for Administrators to manage the system. The management tasks include the development of DP, managing users, setting up and configuring digesting channels, and other system-related tasks such as closing system for maintenance, building report, managing three subsystems, etc. These tasks also affects the three subsystems. For example, when a new DP is built, it will be added into the corresponding subsystem for running. Each subsystem has an underlined *Database* to support all of its internal modules, including DP's components inside it.

The three subsystems are explained in the following part. Trchitecture of each subsystem is visualized with some examples of DP objects.

### 3.5.1 Ingest subsystem

The Ingest subsystem provides tools for Depositor to prepare their documents for archiving. The admission policies are set by administrators to ensure the quality of the archive will be made. By defining the interfaces and admission policies in DP, the admistrators can control the ways digital objects deposited to the system and their quality without reviewing the object content. The

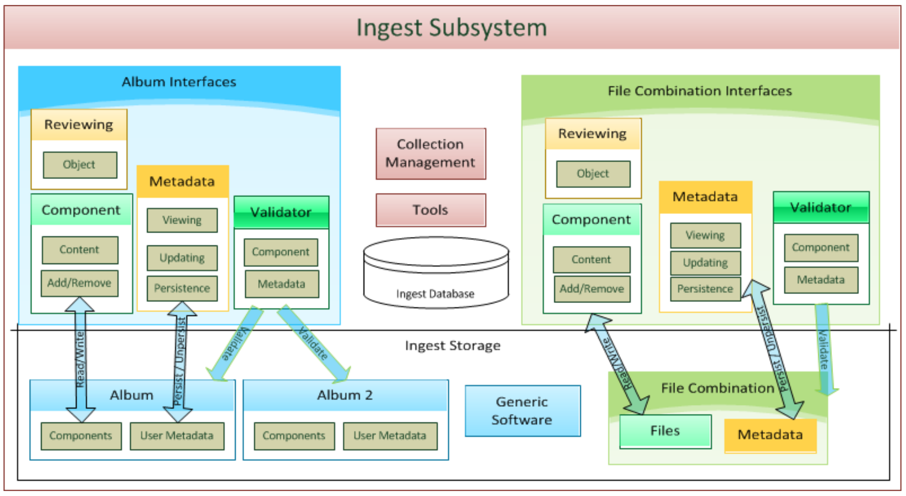

Figure 3.7: Ingest subsystem

updating interfaces in the DP are used to allow Depositor to submit their documents and metadata. The interfaces also check the quality of the submitted objects according to admission policies. When the documents are stored in ingest subsystem, they are called *Collection*. A *collection* and an object of a DP are different. Specifically, a collection is a submitting version of the object, e.g. the collection contains the files and user metadata needed for the creation of the object. When a collection satisfies the admission policy of its DP, it will be allowed to submit. Definitely the decision of submission is made by Depositor, and the Depositor has to accept the archive policies and preservation planning specified in the submitting DP. The ingest subsystem also has to ensure that the files submitted are supported by the system. The discussion about file format support is given in 3.7.1. When the Depositor submit a qualified collection, the ingest subsystem will add preservation and system metadata into it to create a proper AIP and put it into the archive subsystem.

The architecture of the ingest subsystem is illustrated in the figure 3.7. In the figure 3.7, two collections of *Album* DP, one of *File Combination* DP, and one of *Generic Software* DP. The interfaces of *Generic Software* are not shown. The submitting documents and their metadata are stored in *Ingest Storage* in form of *DP's Collection*. The component interfaces in DP's interfaces are used to allow Depositor to make changes on components, and the metadata interfaces receive user metadata from Depositor and persist them into object's user metadata. The Validator Interfaces validate the digital object according to admission policy and shows the result to Depositor during updating process and in the collection management module. The *Collection Management* module has three functions: to summarize the information about the collections being submitted of the Depositor, to allow Depositor to make decision of starting document preservation, and to create the AIP and transfer to the archive subsystem. The set of tools from DP (*Tools*) such as file format conversion tool, file retrieving from a URL, help Depositor during submitting process. The interfaces of each DP are retrieved from admin module.

Each Depositor might have a “limited” storage for storing their documents temporarily before they are archived. The limitations put on the storage is set by administrator based on the capability of the system. These limitations are not fixed. The more powerful the whole system is, the looser the limitations become.Each Depositor might have a “limited” storage for storing their documents temporarily before they are archived. The limitations put on the storage is set by administrator based on the capability of the system. These limitations are not fixed. The more powerful the whole system is, the looser the limitations become. More details are discussed at 3.7.2.

### 3.5.2 Archive subsystem

In the NPA system, unlike other systems, the archive owner might update their archive. The archive subsystem is where the archives are preserved and updated. An owner might also export the archived documents back into the ingest subsystem to submit as new collection if needed, for instance, when the owner need to change the content of documents but it is not allowed by DP preservation planning, or when he want to combined smaller objects into a bigger one such as combining some images into an album. By allowing updates on archive, the system allows owners and consumers take part in the value enrichment process after the documents are accepted. This will improve the quality of the archive. The updates are done through the editing interfaces of the DP.

The figure 3.8 illustrates the architecture of the archive subsystem. Digital objects, either the original or updated one, are stored in form of AIP in *Archive Storage*. All component of DP interfaces are used in the subsystem. Updating a digital object of a DP can be done using the corresponding interfaces together with validators. The owner might also review the object and its changes history. The subsystem has the *Time ensurement* module (visualized as a clock) that regularly checks if the publishing time of an archive has reached, then it will send it to the digest subsystem for publishing. The publishing time is set by archive owner, but as agreed by archive owner on publishing policy, the maximum time can be set is 70 years after the original owner’s death. The Owner can also preview (or test) how their documents are being published through the consuming interfaces of DP. The archive subsystem also provides all tools and devices needed for the preservation tasks according to preservation plan of each DP. Obviously not all tools and devices are available in the beginning, they need to be added later by administrators. The *DP Tools* are built according to the preservation plan of DP. The *Archive Tools* are the internal module used for packing or unpacking archives, or doing other tasks in the subsystem (such as changing archive’s owner, collecting statistical information, etc).

### 3.5.3 Digest subsystem

Digest subsystem is the place where the archived documents are re-used and enriched. It is suggested that once an archive is published, the publishing is permanent, e.g. the owner cannot withdraw his publishing decision. However,

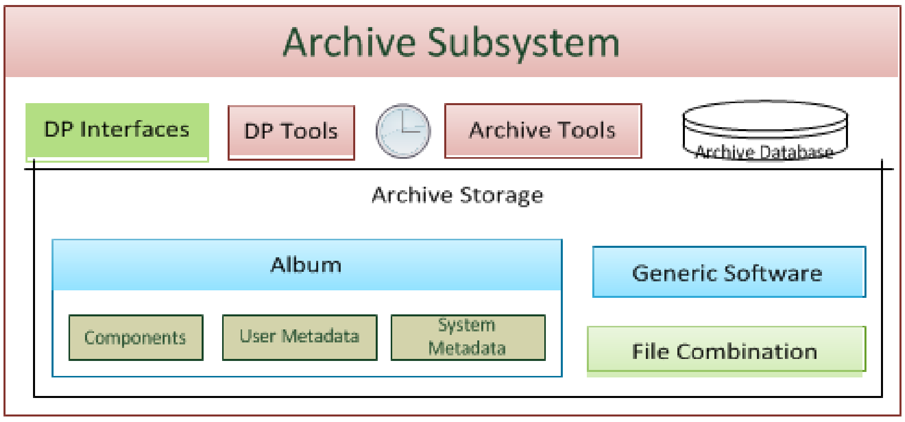

Figure 3.8: Archive subsystem

the owner might update a published archive and might decide to keep the updates for publish later.

### Digesting Channels

Different types of documents need different channels for re-using. For instance, the re-use of an *Image Album* object is showing the images inside its images with metadata. Or for the *Static Web* objects, a channel might be similar to the Web Archive Pages of The Library of Congress (LC) [27] where the objects can be searched or browsed by name, subject, title and all snapshots of each object are shown. The channel to re-use a *Web Application* object might just allow consumers to download all files and metadata in the archive, or it can be a virtual host where the archive owner or consumer can install the web application and run it.

The DP concept and DP object provides the appropriate methods for administrators to manage different digesting channels through DP's *sharing interfaces*. The different channels that can be used to share an *Web Application* object shows that the sharing interfaces are extendable. When the capability of the digest subsystem is not so powerful, the administrator might decide to allow a *Web Application* object to be downloaded only, but later he can add another channel where the re-use can be as authentic as the original application.

### Digesting Objects

Digesting Objects are the archived documents published by their owners via digesting channels. Through channels, the archive owner might also see the feedback, suggested content and metadata changes to their documents by consumers. The archive owner might decide to add these metadata into their original archive and publish their document again or keep for publishing later.

### Module architecture

The architecture of the module is illstrated in the figure 3.9. Four different channels are shown in the figure: Web Host, Broadcasting, Content Delivery

Network (CDN), and File Host. Other channels can be used to delivery such as email, printed documents in public library, sending to other systems, etc. In the figure, the *File Host* channel allows user to get the DP objects in the form of files. The *Broadcasting* channel shares objects through broadcasting like television or radio wave. *Web Host* channel enable owner to put their object on a web server for Consumer to re-use or even make changes (i.g. giving more information about images in album). The *CDN* channel is a distributed solution for delivering media through internet. In the figure, it is used to deliver images, audio and video. The figure also illustrates the different interfaces in consuming interfaces (re-use, share, change). The *Reviewing* shows Consumer the published objects and their changes history. The Consumers access and update the published objects via *Query* module. The *Tools* allows Consumer to configure their ways of consuming. In fact, the three channels Web Host, File Host and CDN above can be combined into one channel that deliver the documents via webserver, and Consumer can select their preferred way in this case.

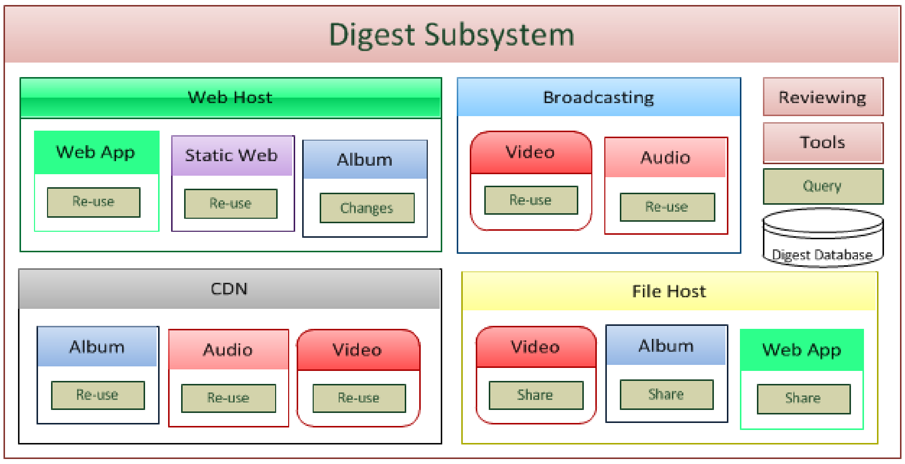

Figure 3.9: Digest subsystem

## 3.6 Document Profile Implementation Guideline

A DP should model one type or class of personal document such as emails, blog (or diary), text document, website, video, audio, album, film, book, classroom report, classroom exercise, a travelling plan, etc. The administrator should define some popular type of DP, and allow users to suggest new DP as well. The DP specification is an important part of the NPA system. Therefore, a guideline for developing specification for DP.

### 3.6.1 Editing object components

Most of the digital objects are composite, which is made of smaller components. The list of components that might be included in a digital object should be built based on the essence of the document. If the child components are files, the recommended interfaces should allow users to upload multiple files at once, or to upload a file package and extract, or to select files from their existing files in the system via a searching interface for file by file's metadata. In a similar manner, when the child components are other objects, the interfaces should enable users to search and add multiple child objects into the digital object.

### 3.6.2 Adding metadata

For specifying metadata that can be used to describe a DP object, the administrator should be able to select the desired metadta from a list of available metadata in the system. If the metadata is not available, the administrators should be able to add it into. By supporting this feature, the system becomes scalable. Because XML is a standard for storing metadata, the recommended way to add a metadata into the system is by uploading the XSD file of that metadata and filling the description information for the metadata. After the schema is added into the system, it should be made available to administrators.

### 3.6.3 The recommended set of metadata

The set of metadata used to describe each type of documents may vary. However, there are some popular standard metadata which are often used to describe each type of document, and the more generic one such as Dublin Core and MODS can be used for many types. The following list of metadata are recommended to be used in the system because they are open and widely accepted:

- Dublin Core metadata: the terms in Dublin Core are very general that are useful to describe the document to give it the universal meaning, thus the metadata enable the document to be linked in bigger semantic networks of resource.
- MODS: the MODS metadata should be used to describe digital object. Although MODS and Dublin Core contains similiar descriptive information, the set of information in MODS metadata is larger than Dublin Core.
- IPTC: the IPTC metadata standard are often used to describe the context information of image. However, some information in the standard are generic and popular enough to be utilized to describe other documents.
- Technical metadata: each type of document has a set of metadata used to describe them. Many standards are developed by The Library of Congress. The recommended standards in the system are NISO MIX, AudioMD, VideoMD, TextMD for describing technical information of Image, Audio, Video, Text, which are the popular documents.
- METS: METS is the recommended metadata to describe the structure of the document and the archived digital object.

For the documents that have not had open and popular metadata standards to describe, the recommended solution is that the Administrators should define a set of required information which are often used to describe that kind of document, together with the above generic metadata standard such as Dublin Core, MODS and IPTC.

### 3.6.4 Admission Policy

Admission Policy is an important and complicated part of the system. The ideal case is that when a Depositor submit a valid document with enough and valid metadata, it should be admitted to the system. The goals of the system are to get more documents and metadata submitted, and the quality of document and metadata should be checked. The more document and metadata are added, the more value the archive object is. Therefore, the system should encourage Depositor (and other users) to contribute more to the system by submitting document and metadata. In addition, the system should enable the users to do so with ease. However, to ensure the document are in high quality and metadata submitted are enough and valid, a minimum metadata requirement and document validity should be specified as admission policy. The admission policy should be different for different types of document. However, by imposing admission policy and data validation, it may reduce user's desire of contribution to the system. If the admission is strict and many metadata are required, it will reduce the user's desire to deposit. In the other words, it will make users tired because the effort spent for the depositing or contributing exceed their patience.

### 3.6.5 Metadata, XML and user input

Because XML is the recommended format for metadata, the system should provide appropriate ways to collect metadata from users corresponding to some XML schema. If the users are familiar with XML and the metadata, the system might allow them to supply the corresponding XML file, and the system only need to validate the file and document toward admission policy of the system for the document. However, normal users might not be not familiar with XML, it is difficult for them to enter correct information to the metadata. Several methods can be used to help them enter the metadata of their document with ease. For example, some metadata can be detected automatically from the content of the document files submitted by users, so the users are only asked to confirm the information instead of entering. Another way is to provide the list of descriptive information as a list of *attributes* and each attribute is presented as an input field for user to enter the corresponding information. There are some attributes which exist in many metadata, such as title and description, thus asking for users to supply the same information in many metadata are not convenient for users. The system might define a list of attributes (and therefore define new metadata) and allow user to input only once for each attribute, then mapping attributes into XML file according to the schema of other metadata.

The developer's implementation efforts of the system for above methods are different. Some methods are better for users to supply documents and metadata, but the effort of realize it or solving the technical problem of it is very high when the target environment of the system is on a web server and the interactive interface with user are a web page. Reversely, some methods needs many effort of users, but it is very easy for developer to implement.

For example, the first method of supplying XML file requires least effort to implement, but it is very difficult for users who have no knowledge of XML to do so. The effort of implementation for the second method of detecting metadata automatically from documents files is different for each type of document file and metadata, although this method is convenient for users.

The method of showing needed information as a list of attributes, which can be called *attribute method*, is suitable for normal users with no XML knowledge. The *attribute method* can be implemented as an interface that shows all of the required fields to users (together with the help or instruction for each field), and then ask user to fill the information. There are some automatic solutions which are already done for this case. For example, a master thesis that build a mapping techniques to load an XSD file and generate the corresponding web form using java [41], some IDE also supports this features such as Visual Studio and Eclipse, though the system user interfaces are recommended to built on web pages. Nevertheless, mapping every attribute and element content as an input, the form will be complicated for user if the metadata have too many attributes to be filled, and thus it can make the user tired and reduces user's submission effort.

Another solution for the *attribute method* is that the interfaces should display only required attributes with their validation rules, and other attributes in the metadata are shown as optional attributes or popular attributes, which can be added by users if they need. The interface might help users by copying the filled information to another child components or objects. By doing that way, it might convenient for user, but it may also reduce the data quality if the user just want to copy the fields to all other components to satisfied the minimum requirement, and in this case, without the human effort, it is very difficult to check the filled information.

The method of defining new metadata for common attributes need a mapping process after to map attributes into XML files of other metadata. The recommended implementation for this mapping process is that the system should load the structure of an XSD file and visualize its elements and attributes for administrators to select on screen. When an element or an element's attribute is selected, its XPATH [44] is returned to associate with the a pre-select common attribute.

### 3.6.6 Validator

As many users are not familiar with some technical metadata and XML, the system should provide more information about the required metadata to guide users to supply correct information and minimize the difficulties of entering information in XML form, while enable users to supply metadata in various ways 3.6.5. In addition, as discussed in 3.6.4, the system should specify a minimum metadata requirement and build validators to verify the document and its metadata for admission. By default, all metadata are described using XML schema, so when user enter all information needed to build the XML file, the file must at least be valid toward its schema first. Validators should at least validate the metadata according to the schema, and the administrator should be able to put additional validations on document and objects like the above DP examples.

Different methods can be used to verify metadata. If the method of uploading XML file is used, validations should be performed on the uploaded file toward its schema. Many programming languages have a module to perform such validation. If there are some other additional rules specified by administrator, they should be implementd manually. Similarly, if the above *attribute method* is used, the rules defined by schema and the additional rules must be implemented manually as well.

The system should provide a convenient interface for administrators to define and associate rules with metadata and object components. For example, the system might visualize the XML schema file on screen for administrators to select which attributes or components need validation, and then select desired rules from a list of validation rules available in the system. In case the administrator cannot find his desired validation, he can still define the name of the validator, and provide the description for it, then ask system's developer for an implementation.

### 3.6.7 Metadata Persistence

The obvious format used to encode metadata is XML because it is an open standard which is widely used in many data-interchange environments. Different persistent forms for metadata can be used. For example, the XML content can stored in the system in forms of file. Each metadata can be stored in a file or all the content of metadata are embeded into a structuring metadata format like METS and stored as one file. Another the persistent form is to store the metadata in database system and the database can be persisted into files. If this persist form is used, a mapping method from XML format into database's entities and relations need to be developed as well.

Similarly, there are also different methods to associate metadata with the components of the documents. For instance, the METS [34] metadata allow internal and external references, which can be used to associate other metadata to files or objects in the archive. On the other hands, the above database method which uses relations among entities for the association. Another method is to store metadata as files and associate with the document (which is also made of files) using a specific directory structure "convention" based on the structure of the document. For example, an *File Combination* object can be divided three directories: *user metadata*, *system metadata*, and *files*. The files in the object is stored in directory tree structure, then the metadata for each files can be stored in the same structure and the same file name, but in a different directory inside the two other metadata directories above. These methods are not mutual-exclusive, e.g. the Administrators may use the METS files to store the association rules between the metadata and components in digital object, while files in the object are associated with their metadata in a directory convention.

The recommended data encoded format used for archiving purpose is XML format. Therefore, the XML files need to be written into archive storage (persistent process) or to be parsed from the archive storage (unpersistent process). However, such processes are not suitable when the metadata are frequently changed or updated by users during the intial submission or consuming periods because parsing, processing and writing data in XML form take more resources (memory, processor) than handling data as traditional structural object of the programming languages such as array or user-defined objects. Consequently, the system might read and write metadata content in the traditional structural objects during such temporary periods to validate, update, display, etc. for better performance. The persistent and unpersistent processes for such objects into other persistent forms are well supported in various programming languages.

### 3.6.8 Archive Package Tool

As mentioned in the section 3.4.4, the archive package tool is the essential tool of a DP that performs the transformation of the digital object from a collection into archive package, from a persistent package into a temporary form that can be processed by the system and vice versa, and then from an archive package into consuming forms. One important aspect needed to be considered when building the tool is the method used to maintain updates and changes of the archive (called *Archive Update Mechanism.* As all changes need to be recorded and the old version are need to be kept, several techniques can be used to implement the mechanism. For instance, when changes are made on the archive package, the whole copy of the old package is retained. The other technique is to make changes on the object's components and metadata, and then add the change event into the package structure together with the old components or metadata. These techniques are similar to but not as complicated as the mechanism of revision control (or source control, version control) in software developement.

The implementation of the *Archive Update Mechanism* should be DP-specific based. Each DP is suitable for a specific archive updates mechanism. For instance, the method of keeping all snapshots of in the Static Web Profile is sufficient to be used as the archive update mechanism of the DP object. However, in other DP object, the child components might not be changed, so the suitable mechanism depends on the particular persistent method discussed in the section 3.6.7, i.e. the changes can be recorded into database, or into files and the old files are kept, together with the change events in the structuring metadata. One of the metadata that might be used to record such events is PREMIS, which is also a standard developed and used by the Library of Congress.

### 3.6.9 Consuming Tools and Techniques

The NPA is built for preserving and sharing values of documents, thus providing the sharing interfaces for consumers are the value sharing process. When the amount of documents and their associated metadata become huge, various advanced data processing and analyzing techniques such as data warehousing, data mining, semantic searching, optimizing data query, etc, should be applied on the data to provide best consumer's experience.

## 3.7 Guideline for implementation

The provided design of the system is platform-independent. It should run on a web server to maximize accessibility to its users. It is recommended that the system is implemented using open-source platform and technology such as using PHP, Python, Java as programming language, GPL Mysql or MariaDB as database server, linux as the operating system, etc. Several guidelines for some important aspect of the system are given in this section.

### 3.7.1 File Format Support

The evaluation for a file format that might be suitable for digital preservarion should take various Evaluation Factors [17] into account: seven sustainability factors, quality and functionality factors, rendering factors (or re-using factors). The criteria to select the format of files in the archive system should obviously be open and royal-free, thus enable the system to perform migrating and emulating activities free of charge. The list of accepted file format can be adopted from the list (published by LC based on the same criteria) of preferred file format for each type of documents including *Still Image*, *Sound*, *Textual*, *Moving Image*, *Web Archive*, *Datasets*, and *Geospatial.* The recommended format for each type of document can be found at the Content Categories Homepage [8] of LC by navigating to each content category, under the "Preferences in summary" section.

For the other non-supported file, the depositor have to convert their documents into supported ones. The system might provide the conversion guideline and suggest alternative format for the unsupported format to prevent depositor from losing information in conversion process. It can also provide the conversion tool in the ingest system if possible. The list of supported files should be extendable. When a new format is added, the archived documents might also be migrated to the new format if needed.

### 3.7.2 Storage

Because the storage technology are fortunately being improved, the cost per a storage unit is reasonable if the storage policy, admission policcy and storage management are created based on the criteria that can prevent depositor submitting unrelated content.

#### Storage policy

The depositor might have a large amount of private document to be ingested, and these unique data contain priceless cultural and historical value. Therefore, in the ideal case, there should be no size limit of the collection, provided that the archive is well described so it is deserved to preserve and enabled to be assessed in the long future. However, the implementation for this ideal case should adhere the practical storage capability, some limitations might be imposed in the ingest system. These limitations should guide depositors to the right way of deposition, rather than prevent depositors from submitting their bigs files. Two typical limitations are minimum description criteria for each document type and maximum stay time of the files in the ingest system.

To ensure that the collection's value can be extracted later, a set minimum description criteria should be defined. When submitting their documents, the depositors are required to provide description information for every object exceed these limits. These description should be validated automatically according to some rules. For example, the information might be checked for grammar and lexical errors, the size of description information should exceed a minimum, and it might not be redundant, etc.

For the system's endurance, there might be a limitation of time on the amount of data should be kept in the ingest system in order to promote depositors to provide description data and prevent the depositors from using their storage at the ingest system for backup purpose. When a collection is accepted, it will be moved from the ingest system and preserved in the archive system, hence recover the storage capability of the ingest system. The archive system storage is supposed to be able to preserve the well-described documents without limitation.

#### Storage Management Guideline

Because in the ingesting system, depositor might upload many upload file. Therefore, the system might have to utilizes a distributed file system to support the ingestion function. It is recommended that the storage location should be placed near the depositor's geographical home location (e.g. each server for each region, county, or even city, based-on ingesting statistics). This requires implementation solution should support the distributed architecture.

For security and safety reason, a centralized storage system should be used to support this function. This system is suggested to be the National Library. In the digest system, because the amount of content could become very large, the system might utilizes the power of cloud-based solution such as CDN, which improve both performance and security

### 3.7.3 Time ensurement module

The time ensurement module regularly checks archives for publishing into the digest subsystem. The module should be able to read the publishing information of the archived objects in the archive subsystem. However, to keep the privacy of the archive objects, the module might not be allowed to open the archive package. Therefore, the recommended solution for this problem is that the information needed to check if the publishing time of an archived object has passed should be given in the package's filename as suggested in 3.7.4. This schedule can be implemented in a form of a time-based schedule program like cron.

### 3.7.4 Archive autonomousity

The AIP is recommended to be autonomous, i.e. when an AIP is acquired, the content of the AIP can describe itself. The autonomous AIP can be backed up easily by copying without any further information collecting. As the main functions of the system is to preserve the private documents in form of archives, so by making its archive packages become autonomous, the archive packages can be "decoupled" from the system for a long term preservation plan in secure storage media. When the system goes down and its running storage media are damaged, the whole system can be recovered easily from archive packages stored in backup media because all the information needed are packed inside the packages. In the other words, the archive packages can also be brought into a new system running on a different platform as long as it implements the specification of the system and the corresponding DPs, which are platform independent and based on open standard.

An autonomous archive package should be able to fully describe itself by the information stored inside without further information from the archive system. However, as the archive policy might not allow other people to open before a certain time, the archive package should show the necessary information via its filename without openning the archive content.

The following information should be included in archive package filename:

- Owner ID: to identify the owner (in case it is located somewhere than the usual place).
- Archive ID: to know whether the system have it or not.
- Admission time (and date): to identify the system information (policy, version, etc.) when the archive was created.
- Document profile ID: to know how to open it, how the archive package is structured.
- Last update time: to know the version of archive, because the archive might be migrated during the period from admission time.
- Last original owner's update time: to know the latest time when the original owner is alive.
- Publishing time: to know whether the system can publish it (with its owner's permission or with default system policy). By the system policy, the latest publishing time that the owner can set is 70 years after the original owner's dead.

### 3.7.5 User participation in value enrichment and extraction

The value of a document depend mostly on its content. However, the metadata essential for value identification, assessment and extraction, because without it, the document might not be identified and use. Therefore, while the content of the deposited document should be kept as original as possible, its metadata should be enriched as much as possible. The owner should take the main role of the enriching process, e.g. providing metadata. In addition, other users can also take part in this process by adding metadata when the documents are published based on the document's content and its previous metadata. Because the amount of documents in the system might be very high, various data mining and indexing strategies should be applied on the metadata and document content to help consumers get the document and statistical report they need.

### 3.7.6 Copyright and licensing

The authorities managing the archive should create appropriate license agreements between the system and its depositors. The authorities should not only encourage the depositors to submit their documents, but also explain the terms of the specific licenses which the depositors are applying on their private documents when the documents are archived and shared.

Several solutions can be used to implement these agreements. The first solution is that the system provides an agreement containing a list of licensing options that might be applied on all types of document deposited. If Depositors decides to submit their documents to the system, they have to accept the agreement, which means that when the documents are published before their copyright protections expire, the archive owners have to choose an option from the provided list in the agreement. Another solution is that the agreement is not system-wide, but DP-specific, which means that depending on the type of documents, the agreement might be different.

Because the system promotes the public sharing of the private documents, the license agreement of the system should maximize the consumer's participation in value enrichment processes. Therefore, the list of Creative Common licenses are recommended to be included as the required licensing options in the license agreement between the system and its depositors.

### 3.7.7 System Administrator's role

Various tasks mentioned in various sections so far can be summarized as followed:

- Developing DP's interfaces and validators.
- Specifying metadata requirement.
- Assessing file format support and prepare preservation planning for files.
- Developing different channels for re-use.
- Developing data processing and analyzing tools in digest subsystem.
- User administration: user, owner, consumer.
- Providing more hardware device if needed (e.g. more storage).
- Managing system's performance and taking appropriate actions to improving performance, such as providing more servers for load balancing.
- Tracking system status and doing maintenance tasks.

These tasks should be assigned to the suitable staff of the authorities managing the system. Because administrator's tasks strongly affect the behavior of the system, it is recommended that the above administrator's roles should taken by many different administrators. In addition, each administrator should have limited rights to control the system. The system can employ the Role-based Access Control (RBAC) technique which can assign rights to each role which can be taken by one or more administrator accounts. Such roles are beyond the scope of the thesis. Within this thesis, when referring to administrators that change or control something in the system, it indicates that the administrators have the appropriate roles and rights to do so.

### 3.7.8 Security

Because the system stores the "sensitive" data which are the private documents that might not be published yet. Therefore, security is an important aspect which has to be considered before the system can accept the private documents submitted by Depositor. By dividing the system into three subsystems, the security protection is easier inside the system, such as using advanced authentication methods discussed in the section 3.3.5, and the access to these subsystems might be provided through Hypertext Transfer Protocol Secure (HTTPS) instead of Hypertext Transfer Protocol (HTTP). In addition, hardware, storage, secure backup and other components must be protected from unwanted access. In addition, different authorities might have administrator accounts to operate and control the system, thus the usage policy of the system should have some regulations for administrators such as an administrator must not give his account information to others, or an administrator account is created only with a written permission from an authorized person, etc. However, the further discussion about security is beyond the scope of this thesis.

### 3.7.9 Scalability and flexibility

A system is scalable if it can be "easily" expanded (or scaled) to a bigger size. Similarly, a system is flexible if it can be "easily" changed or added new features, components, functions, etc. How "easily" the actions are done depend on the context of the actions. Basically, the costs, efforts and (or) resources spent to perform such actions should be low. Scalability is an important factor of the system, as it is supposed to have a large amount of users and to receive, process, preserve and delivery a large amount of data. Similarly, the system should be flexible enough for the managing authorities to add more functions and components to reply the continuous changes of the practical situation such as the changes in file format and metadata version, the occurence of new types of document and components (like conversion tools and channels), etc.

The design of the system is scalable and flexible. By dividing the system into three subsystems which only share digital objects among them enable each subsystem to be managed, operated and upgraded separately without affecting others. The design of each subsystem is also scalable. For example, each subsystem can be divided into different components that serve the submission of different DPs or different sets of users in the different regional areas (which can be obtained from user account information). Similarly, each channel in the digest subsystem can be served using a separated server for better performance. The archive autonomousity feature in the secton 3.7.4 also maximizes the implementations for archive storage in the system. When the new global tools and DPs are created in the system, they can be delivered into subsystems and used. The DP make the system scalable and flexible for adding more types of document, tools, and channels into the system. In addition, the change or creation of a DP will not affect other existing DPs, thus the system can adapt the new changes in the its environment.

# 4 The prototype specification

## 4.1 Prototype's overview

In this chapter, the design of the system presented in the chapter 3 will be called the “referential model”, and the prototype (or demo) system is an implementing step on the roadmap to the working system described by the referential model. A working system might be only a minimal system that can start accepting, preserving and sharing certain types of documents, and it can be extended later to a final one, which might be called that can run in a long term period without major changes. Because of time limit, the purpose of building it is to prove that the design and concepts work as intended. Some features are simplified, some are built as stubs which are not available but can be extended later, and some are not available since they are not essential. The prototype is the iterative approach to implement the essential part of the full system, but it is not tweaked for performance.

### 4.1.1 Technology Selection

As mentioned in the previous section, the design is platform-independent. Therefore, the choice of platform used to implement the system depends on the authorities that support the system.

**Environment**

The selection of technology and platform for this thesis is based on convenience for the developer. All tools and software used in the prototype system are free and open sources. The technology used in the prototype is LAMP, which is stand for Linux - Apache - Mysql - PHP. PHP is a server-side script language which is platform-independent, widely supported, and mature enough for satisfy all current requirements of the system. If any needs arising in the future which are outside of PHP capability, the demo system can easily be extended with other server-side script languages such as Python, Perl and Ruby (supported by httpd Apache web server). It can also be extended to work with Java technology using mod_proxy of httpd Apache to send “java-request” to other Java Servet Container. Many people claim that PHP has problem with scalability, i.e. it is difficult to expand the built system into a bigger one. However, thanks to recent improvement of PHP contributor's efforts, the problem is becoming less and less. For example, Facebook by now is one of the most-viewed pages built from PHP.

### Yii

Yii [46] stands for “Yes, it is.”. It is a high-performance, well-designed PHP framework, with many attractive features, and are capable enough for building a large system. It provides many facilities for developer, but it does not impose developers to use it. As stated in its homepage, “Yii comes with rich features: MVC, DAO/ActiveRecord, I18N/L10N, caching, authentication and role-based access control, scaffolding, testing, etc”.

Yii, like many modern web frameworks, employs the Model-View-Controller (MVC) pattern (or usually called MVC “model”) which is a programming design pattern adopted into web development. In the MVC model, the Controller is used to process information, Model is to manipulate data, and View produces the interface between users and the website. The MVC process is illustrated in the figure 4.1.

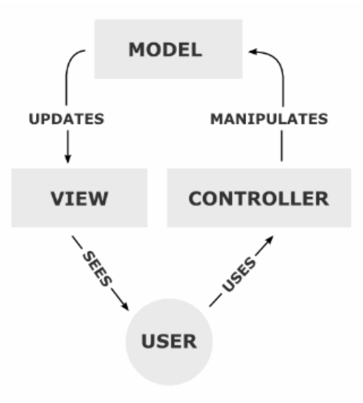

Figure 4.1: The MVC process (Source: the Wikimedia Commons [36])

A Yii web application can be divided into modules and shared components. Adding external extensions and libraries are also easy. Yii is also scalable and suitable for high performance systems. It support database caching, international translation, RBAC, Test-Driven Developement, etc. More information about Yii can be found at its homepage [46].

The Yii’s CActiveRecord class (and its parent class CModel) can be used to implement the “model” component in MVC model. They provide necessary methods validate data and mapping data into database. Mapping from a table in a Relational Database Management System like MySQL to a CActiveRecord class can be done using Gii, a tool provided by Yii framework. Developer can also generate a controller class and many views that allows user to input and modify the data modelled by the corresponding generated CActiveRecord class.

### jQuery, CK Editor and JSTree

jQuery, which is one of the most popular (and best) javascript libraries, is supported in Yii for developer to write client script. jQuery can be extended using plugins which can be provided by other parties. Two plugins added into jQuery in the prototype are CK Editor and JSTree. CK Editor is used to provide a What you see is what you get (WYSIWYG) text and HTML editor on the web. JS Tree is the library for providing an interactive tree structure to user in a web page. More information about jQuery [22], CK Editor [7], and JSTree [24] are available in their homepages. The detail use of these components are discussed in the section 4.2.

### 4.1.2 Prototype architecture

The whole system is implemented as a Yii's application with 4 modules: ingest, archive, digest, and admin. The three ingest, archive, digest module implements the three subsystems, and the admin module implements the *Admin Module* inThey are the implementation for the 3 subsystems and the *Admin* module. How the features are implemented is explained in the section 4.2.

Only the *File Combination* DP is chosen to be partly implemented. It allows Depositor to create collections and upload files into them, together with editting their metadata via an editting interface for both files and metadata. The interfaces are integrated into the ingest subsystem, and only Depositor can access it. The archive subsystem and digest subsystem can be accessed by any registered user, but they are not implemented yet. The *Admin Module* can only be accessed by the account named “admin”. As specified in the referential model, the *Admin Module* controls the whole system, including the *Information Module* which provides the basic information about the system and DPs to users.

### 4.1.3 File arrangement

The files of the prototype system is arranged as follows:

- All files are stored in a directory named “ndfa”.
- The database file is stored in *ndfa/protected/data*.
- The places of controller, views, model, component, module, and others follow Yii's directory convention.
- The collections of the ingest subsystem is stored as *ndfa/protected/modules/ingest/archive*, so this directory and its childs must be writable for web server process.
- The *ndfa/assets* is used to publish HTML element files, thus it must be writable.
- The *ndfa/protected/modules/archive/storage* is used to store archive.
- The *ndfa/protected/modules/digest/channels* must be writable by web server to publish document.

## 4.2 Feature implementation

The following features are implemented in the demo system.

- User registration, authentication and Depositor elegibility check. Depositor's accounts are the those whose names are valid Norwegian ID number.

- The basic DP tools are built including an XML schema browser, a tool to change DP user description, the basic metadata validator, the attribute and defining interface.

- The File Combination DP is partly implemented. The basic component and metadata requirements for admission are built.

- The *Help* MVC implements the *Information Module* whose content are defined by administrator from the *Admin Module*. The content includes the basic information of the system such as system policy, copyright agreement, user guide, and the list of supported file types and formats, the information about DP.

Because the Administrators control the system, including DP's tools and interfaces, the above features can be also considered as a part of Admin Module.

### 4.2.1 Data Model

The database supports the prototype system is modeled in a Entity-Relationship diagram shown in the figure 4.2. The schema is produced using Mysql Workbench. The modeling file (datamodel.mwb) can be found in the same directory of the SQL script file (see section 4.1.3). The explanation for each entity shown in the diagram (which has the name of entities in lowercase with underscore) is explained in the following part of this section.

#### info_page

The *Info Page* entity models the basic information pages of the system, which are a part of Information Module.

#### xmlschema

The *Xmlschema* entity models a metadata which is defined by admnistrator. The administrator can add new metadata by uploading its XSD file.

#### document_profile

The *Document Profile* entity models a DP. The *name* property is unique and supposed to indicate to component that implement the DP, though this feature is not available yet. The entity allows administrator to change the user_guide property which is displayed in the Information Module.

#### user

The *User* entity models the accounts in the prototype system. As mentioned before, the username of the Depositor must be a valid Norwegian ID number. One email should be associated with only one user. The *status* property is used for account verification (not implemented in this prototype).

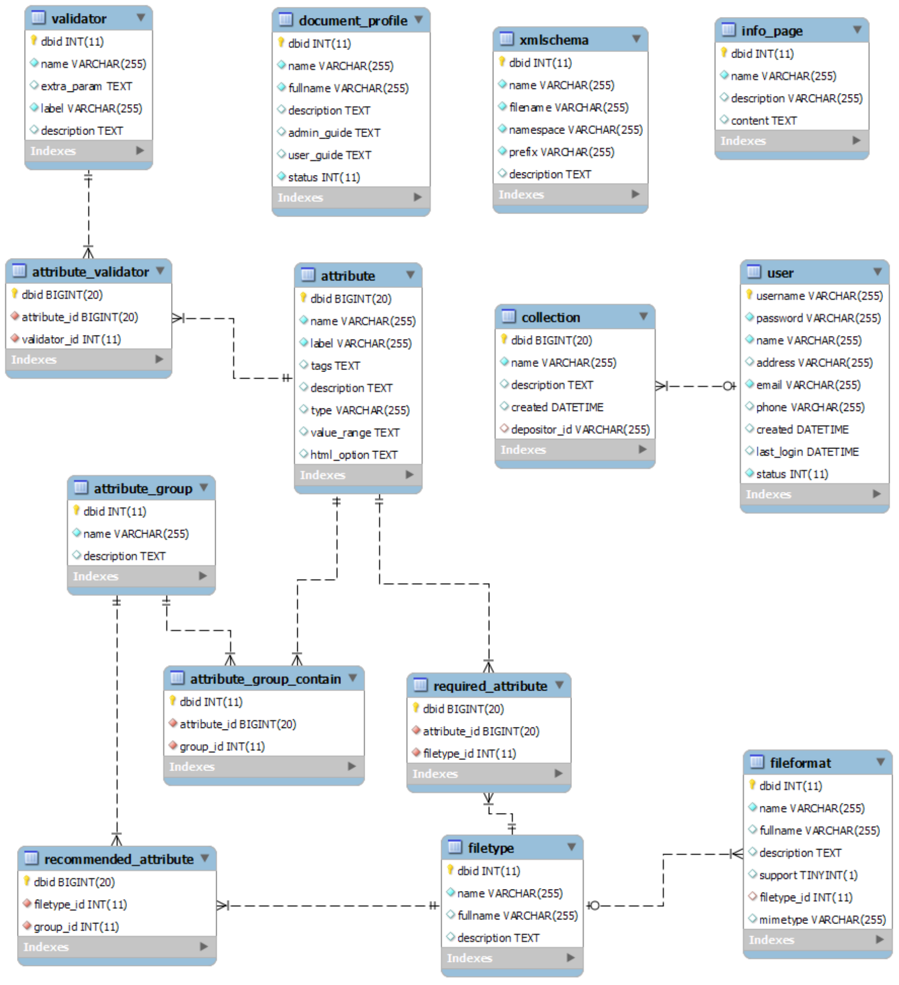

Figure 4.2: Data model of the prototype

### collection

The *Collection* entity models the collection stored in the ingest subsystem. Each collection belongs to a DP specified in its *profile_ id* property. However, currently in the demo system, all collections belong to the File Combination Profile.

### attribute

The *Attribute* entity model an input field for user. The *name* property is the unique name of the attribute in the system. The property appears to the users with a name defined in *label* property. The *tags* attribute defines a number of tags separted by comma to enable administrator to search for attribute easier. The *type* property defines the type of input fields used to collect user information. If the input field is a dropdown list, the set of its options is defined in *value_ range* property. The *html_ option* is used to format the appearance of the HTML input. Many similar attributes can be grouped into an attribute group (modeled in *Attribute Group* entity).

#### validator

The *Validator* entity defines the validation for an input field of a DP. The *label* property indicates the unique name for the validator. The *name* indicate the name of the class or method that implements that validator. The *extra_ param* property allow administrator to define extra parameters to the validator. See validator's implementation 4.2.2 for more information.

#### filetype

The *File Type* entity is used in the File Combination DP. A file submitted in File Combination DP is required to specify its file type. There is a set of file format for each file type, and some file formats are supported.

#### fileformat

The *File Format* entity is used to defined file formats for the system. It has a *support* property to indicate the file format is supported by the system or not. Although practically it may belong to more than one, a format in the demo system belongs to only one file type. However, administrator can still define a "Ogg video" and a "Ogg audio" format which belongs to the corresponding file types.

#### Many-to-many model

Some entities shown in the diagram are many-to-many relationship between two or three entities. The attribute_group_contain entity model the many-to-many relationship between attribute and attribute group entities, which means that an attribute may belong to many attribute group, and each group may have many attributes. The attribute_validator entity models the many-to-many relationship among Attribute, Document Profile and Validator, which means that the attribute in the DP should be verified using the associated validator. The required_attribute entity models the many-to-many relationship between Attribute and Filetype, which defines the required attributes for each file type. The recommended_attribute entity models the many-to-many relationship between Attribute Group and Filetype, which defines the recommended attribute groups for each file type. The interfaces used to manage a many-to-many relationship in the Admin Module has two listboxes listing the child components which can be multiple selected and assigned into a common container to create the relation.

### 4.2.2 Document Profile Tools

The DP tools implemented are also the basic one which can be applied into various DP.

### XML Schema tool

The XML schema tool allows the Administrators to manage standard metadata in the system. The Administrators can add a new standard metadata by uploading the XML schema file that describes the metadata. The *XsdReader* component can parse the file for the structure of the metadata and show it in a schema browser window, which utilizes JSTree for formatting the view of the metadata tree structure. When Admnistrators click to select an attribute or a content element shown in the schema browser, the path leads to it will be shown.

If the system can parse the XML schema file for the metadata structure, it is easier to create the interfaces for editing the corresponding the XML instance (document). In addition, the mapping attributes technique discussed in the section 3.6.5 can also be implemented as a method of standard metadata generation method from user-defined metadata, because the Administrators can define a mapping rule between an attribute the user-defined metadata into an attribute or content element of a standard metadata shown in its schema browser. The XML mapping technique can reduce the number redundant attributes in different standard metadata, so it will definitely improve user's experience.

### User metadata and Admission Policy

The only file requirement implemented is that all files submitted must be supported by the system. The system has a list of supported formats which belong to a list of file types. The system did not verify the information of file type and format given by Depositor, i.e. the Depositor can upload a format and tell the system a different one. The effort for detecting a correct format of a file is beyond the scope of this thesis. When the system is improved toward a working system, it may suggest the format for user based on the methods discussed in the section 2.2.2.

Metadata requirement is implemented through the definition of minimum metadata for each file type from the Admin Module. The *attribute method* discussed in the section 3.6.5 is used as the input method for user to enter metadata. The Aministrators define a list of required attributes for each file type, which will be shown in the metadata editting interface. In addition, the Administrators can also recommend some popular attribute groups that are often used to describe a file type. On the other hands, Depositor can also define their own attributes which are called "custom attributes".

### Validator

The Validator class implements the metadata validation feature in a similar manner of the model's properties validation method in Yii. A Yii's model object has a set of rules applied on a list of its properties (attributes). Each rule performs a check on the declared attribute list. The rule can be implemented by a method inside the model or by an instance of the class CValidator. Yii has a list of built-in validator for validating common attribute value rules such as required, numeric, integer, file, email. A rule can also has some parameters such as the minimum and maximum value for integer, the max length of a text field, etc. Similarly, the administrators of the system can select a set of validation rules, which are defined in the instances of Validator class, to assign to the existing attributes in the system. When user enter data into inputs, a dynamic model, i.e. an instance of the class NdfaDynamicModel, will be created. The list of attributes needed to be validated and their validation will be attached into the instance dynamically (e.g. at the runtime) to perform the same validation process as done in Yii.

As shown in the section 4.2.1, Validator class is the mapping model for the *Validator* entity. Each Validator instance will be transformed into a rule applied on the attribute defined in the many-to-many relation between the Validator and Attribute Entity. To define a new validator, the administrator should have some knowledge of Yii. If the new validator similar to the Yii's built-in validators, the administrator only have to supply the `name` and `extra_param` in the same manner of a rule in Yii. Similarly, if the validator is not available, the administrator has to define new child of the CValidator class to implement the validation before he can supply the corresponding name and params for the new validator, thus this validator solution is extendable and flexible for verifying metadata.

### User Description

The user description, or user guide of the DP can be defined in the field *user_guide* of the *Document Profile* entity. The system utilizes the CK Editor library to provide the WYSIWYG editor for administrator to edit administrator guide (for developing tools, metadata, requirement, etc) and the user guide that give users the all the needed information about the DP. The user guide will be published to all users through the *Information Module* of the system.

### Interfaces

The editting interface for a File Combination object consists a directory browser together with metadata and status of the files. From the browser, the user can upload, download, delete files, and create sub directory. The status of the metadata of the files are also shown on the browser. For user's convenience, the valid filled in metadata of a file can be copied into other files. In the metadata editting, the Depositors need to specify the file type for each uploaded file. Each file must be described using a predefined schema that has a set of required attributes and a set of popular (or recommended) attributes for that file type defined by administrator. The Depositor can add his own attributes if he want. The Depositor also have to set a publishing date which can be immidiate after the collection is accepted, or later on a specific day or after a period of time after his death.

The reviewing interface shows the summary of the submitting collection: the total number of files, the size of the collection, the result of the admission policy check and how many files have fulfilled the metadata requirement.

Although the interfaces are not implemented in the archive and digest subsystems, they are similar to the interfaces in the ingest system. For instance, the archive and digest subsystems also need the interfaces that allow archive owner and consumer to edit metadata, though they might not make changes on the old metadata. Similarly, they both need the interfaces that allow user to download the files inside the archive.

**Archive persistence and unpersistence**

Currently the demo system only implements the ingest subsystem, so the user metadata are persisted into a JSON string and written into files. The association between files and their metadata used in the demo is the directory convention technique discussed in the section 3.6.7. The files are stored in the "files" directory and their user metadata are stored in the "description" directory with the same structure and file names. All files of a collection are placed in a directory that uses the collection ID as its name. Similarly, all collections of a user are kept inside a directory that has the same name as User's username.

### 4.2.3 Information Module

Information Module is implemented by the HelpController of the system. The content in the module is provided by Adminstrators through the Admin Module. The page that shows the list of file types and formats loaded the information from system database. Similarly, the page shows the DP guide are loaded from the Document Profile entity. Other pages, which can be called "static pages" because it is rarely changed in a working system, only show the content prepared by Administrator with the help of the CK Editor.

### 4.2.4 Admin Module

The Admin Module has the interfaces for Administrators to manage the whole system. The tasks are done by modifying the database of the system. For simplicity, only one database is used in the prototype system. The database contains all the tables needed for the system's global modules and subsystems, each above entity will be mapped as a table in the database. The modification interfaces for database content are initialized using the Gii tool which implements the Object-Relational Mapping (ORM) technique mapping each entity (or table) of database into a model class the prototype. Gii also generates the relations among entities in the database by reading foreign keys defined in the database.

After models are generated, simple modification of database can be done by using the "CRUD" (Create, Read, Update, Delete) interfaces which can also be generated using Gii. Some entities can be modified using the default generated interfaces, but others obviously need some changes in rules and interfaces for particular use. For example, the auto-generated interfaces cannot be used to modify the relationship between model. The interfaces to modify the *Xmlschema* instances need a field to upload the schema file. In addition, the view interface need to be changed in order to show the structure of the XML document defined by the schema, etc.

## 4.3 Prototype Installation and demonstrating instruction

The prototype of the system is installed at the URL `http://arkiv.hiof.no/ndfa/`. It can also be installed again in another location by following the below installation instruction.

### 4.3.1 Installation instruction

- install LAMP, enable Apache's mod_rewrite module.
- configure some PHP parameters in *php.ini* that enable the script file execute longer and handle big file upload, such as *max_execution_time*, *max_input_time*, *upload_max_filesize*, *post_max_size*, *memory_limit*, etc. More information can be obtained on PHP's homepage [40].
- extract the file ndfa.zip which contains the directory *ndfa* of the prototype's source code in a http-accessible directory.
- extract Yii framework (version 1.1.12) directory into the parent directory of application source (so yii and ndfa are in the same directory).
- create a database name ndfa (which stands for Norsk Digitalt Folket Arkiv) and fill the database access information into the *main.config*.
- the SQL script used to initialize database structure is stored in *ndfa/protected/data/ndfa.sql*.
- set permission for these directories to be writable for web server process according to the files arrangement of the prototype specified at 4.1.3:
  - *ndfa/protected/modules/ingest/archive*: storing submitting collections.
  - *ndfa/assets*: empty at installation.
  - *ndfa/protected/runtime*: due to Yii's requirement.
  - *ndfa/xsd*: storing XML schema.
  - *ndfa/protected/modules/archive/storage*: storing archives.
  - *ndfa/protected/modules/digest/channels*: storing publishing documents.

### 4.3.2 Feature overview

The following features can be demonstrated in the prototype system:

- The interfaces of File Combination Profile can be demonstrated in the ingest subsystem. The interfaces are not available the other two subsystem. File upload is used for modifying components, and the *attribute method* mentioned in the previous chapter is used for modifying metadatawhich enable Depositor to enter information about the uploaded file using a user-defined metadata which has three set of attributes (required, popular, and custom attributes).
- For simplifying user input effort, user only have to supply a set required attributes for each type of document. Some attributes can have additional validation. Other additional attributes (custom attributes) might be added according to user's needs. The information can be copied into the metadata of other files for user's convenience.

- The prototype uses JSON as the temporary persistence format in the ingest subsystem. Each collection has an ID, which is also the name of the directory contained the collection's files and metadata. All collections of a Depositor are stored in the directory whose name is the Depositor's username. Metadata are associated with their files by directory convention. Each file in "description" directory stores the metadata for the corresponding file in the "files" directory.
- Access Control module checks for elegibility of Depositor, i.e. only an account with a valid Norwegian ID number is allowed to access the ingest subsystem.
- The content shown by the Information Module can managed from the Admin Module.
- The system can parse an XML schema file and visualize the structure of the XML document described by the file.

### 4.3.3 Demonstrating instruction

The layout of the page contains the needed information to navigate through the system. It has the link to ingest, archive and digest module and information module (help). It does not have the link to Admin module. To access the admin module, the administrator need to login into an administrator account type the URL of the system plus "admin".

#### Information Module

The information module can be accessed via user's browser without authentication. It has a set of pages which is easy to navigate to for showing various information of the system such as system overview, copyright agreement, system policy, user guide for the whole system (4 static pages), the list of supported file format, and user guide for each DP.

#### Admin Module

The administrator need to initialize the system by logging into the admin module, using the account name "admin" and password "admin" (which might be changed later on the above URL - `http://arkiv.hiof.no/ndfa/`). Only this account can access the Admin Module. Administrator can define the following information:

- A set of file type for File Combination.
- A set if file format supported by the system, which file type the format belongs to.
- All attributes used as user inputs which will be mapped into other metadata.
- Validation rules for each attribute in each DP.
- A list of required attributes used to describe each file type.

- A list of attribute groups and the attributes inside them.
- A list of attribute groups recommended to users for each file type.
- A list of metadata by uploading the XSD files and giving the information for the *Xmlschema* entity.
- Enter the information into the information module. The “Info Pages” link allow administrator to define the content of 4 static pages mentioned in the Information Module above. The list of file formats and the DP user guide can be editted by choosing the corresponding link on the menu of the Admin Module.

**Ingest Subsystem**

Visit the homepage and create an account in the system for use. The prototype does not implement any account verification methods. However, only the account named in a valid Norwegian ID number is allowed to access the ingest subsystem. After such account is created, the ingest module can be accessed after logging in.

Depositor can create a new collection to start depositing. Depositor needs to specify collection name, description and collection’s DP. After the collection is created, the depositor can turn into the review interface of the collection and edit it via editting interface by choosing “Edit collection’s content”. After finishing the content modification, users can turn back to review interface and check for colletion status. If the collections do not satisfy the admission policy, users have to edit the collection’s content and metadata until the admission policy is satisfied. Currently Depositor cannot submit the collection when it is ready.

To edit collection content, user need to upload files into the collection, and providing metadata for each child component. The “Check form” button enable user to check the metadata toward the validation rules defined by administrators. If the check is failed, the user can still save the metadata for later edition, but only valid data will be saved and copied to other files or child components. The component editting interfaces for File Combination and Image Album are similar. They allow user to add files into the collection and delete files from it. The File Combination editting interface also allows users to create directory inside the collection. The way to upload multiple files is to upload a zip file and check the extract checkbox in the interfaces. The interfaces allows user to copy the entered information to other files in the collection. User can select the files to copy information, or copy to other files in the same directory, or copy to all files in a collection.

## 4.4 Roadmap toward the first working system

The demo system has the basic tools that can be used to build up a working system. The following list is the remaining components needed to be developed in order to turn the prototype into the first working version.

### 4.4.1 System metadata and association rule

Other metadata that contain the information from the system must be defined and included into the system metadata of DP object. As discussed in the section 3.6.3, the recommended structing metadata is METS, together with a user-defined metadata that can contain all the needed information to make the archive become autonomous as discussed in the section 3.7.4. Obviously the user-defined metadata are defined by Administrators and may vary for each DP. However, the system metadata must have enough information to make the archive become autonomous. Therefore, the system metadata must include the following information:

- The information of the original owner and current owner, which are not only the name and the Norwegian ID number of them, but also their basic information and contact information.
- The date of submission, publishing date and status (whether it is published or not).
- The information about the DP such as the type of child component, the component-metadata association rules, the set of metadata used to describe child components.
- The structure of the system metadata in the package and the update history of the archive package.
- Some other system's information such as the version of the system, the authorities that manage the system from the accepted date.

### 4.4.2 Archive Package Tool

Obviously the archive package tool mentioned in the section 3.4.4 can only be implemented after system metadata are defined. Archive Package Tool implements the persistent methods and archive update mechanism for DP, so it should be built and integrated into its corresponding DP's interfaces. Following the guideline discussed in the section 3.6.8, the recommended persistent methods are to use the XML files to store the metadata, to use the directory convention to associate metadata files and object component's files, together with describing the association rules and the archive package structure using the METS metadata. In addition, the archive update mechanism should be implemented by recording change events into the event element of PREMIS metadata, the updates are not applied on the old components or metadata files, and the new version of metadata are created as the outcome the recorded event.

### 4.4.3 Finalizing File Combination Profile

After the system metadata are defined and the archive package tool of the File Combination Profile is implemented, the other components of the File Combination Profile can be implemented such as the mapping interface for mapping the user-defined metadata into the standard metadata. The interfaces of the DP that used in the archive and digest subsystem can be adapted from the interfaces in the ingest subsystem. Once the File Combination Profile is completed, the system will have many basic tools to implement other DP.

### 4.4.4 Popular Document Profiles

The system needs to create and implement various additional DP to allow users to submit other type of documents. The Image Album is the DP for one of the popular type of document. Similarly, the system also needs the similar DPs: Video Profile, Sound Profile, Simple Text Document Profile, Formatted Text Profile, Email Profile. The Static Web, Dynamic Web, Generic Software Profile mentioned in the section 3.4.2 should also be included because they are the most popular type of documents.

The system needs more type of DP for the better preservation and consuming form. While the File Combination DP can support various type of documents, it is a too generalDP that does not categorize its child components. The situation can be illustrated by a metaphor that a person has many types of things to preserve such as books, audio CD, movies DVD, album of images, and letters (all are in traditional form, not digital form). Although he might attach each object description paper on each above object and then put everything into a big box and preserve it, that might not be a suitable way to handle that problem. It might be better if he puts all of his books together in a box that fits them, and writes a description of all the books contained inside the box as well as the brief content of each book, and does the similar way with each other type of object, so they are easier to be searched and identified content later. The various DPs can be viewed as the different types of box for different types of document, while the File Combination Profile can be viewed as a big box can store many types of documents. By storing objects in the suitable containers, it is easier for the system, which is a computer software, to manipulate them in consistent and meaningful ways. For example, the system can build a channel just for display all the images in an Image Album Profile object or a channel that play the video files in an Video Profile object, which are similar to many current image and video sharing websites.

However, the Depositor should be able to choose which type of DP he wants to put his documents into. Therefore, the generic DPs such as File Combination Profile, Generic Software Profile, or a Multimedia Profile for storing all kinds of multimedia documents are still necessary in the system.

### 4.4.5 User Authentication

The user registration must be improved to verify that a Depositor account created belong to right owner, i.e. person who has the Norwegian ID number given as the username. A better solution for the registration is to integrate the MinID Common Login Portal into the system, which needs an approvement from the related Norwegian Governmental authorities.

### 4.4.6 System's logs

The Logging feature is obviously the essential tool for Administrator to ensure the system work without severe problems such as security and privacy violations, system crash, damaged storage, etc. The system needs to keep tracks of its user's activity, system's performance and behavior. The suggested solution is using the Yii's Logging feature. The feature also supports server's context information, performance profiling and SQL execution profiling. The logs can be saved into different forms as stated in the introduction about the logging feature at Yii's homepage [46]:

> Yii provides a flexible and extensible logging feature. Messages logged can be classified according to log levels and message categories. Using level and category filters, selected messages can be further routed to different destinations, such as files, emails, browser windows, etc.

### 4.4.7 Admin module

The Admin Module should be improved before the system is ready for depositing. The list of essential improvements are as followed:

- Essential system management tools for administrator to check system's status and performance (such as the above system's log function).
- DP's tools and other tools for system management activities such as backup tools, archive owner changing tool, channels configuration tools, etc.
- An access control method for administrator: the RBAC mechanism should be used to divide the roles of administrators into many different roles as discussed in the section 3.7.7. Some recommended roles for the system in the first working version are root administrator (held by the authority that manage the system) who has the access to system management tools, DP content developer (held by archivist) who defines the content, admission policy and preservation planning for the DPs in the system, infrastructure administrator who manages the server, storage, channels performance in the digest subsystem, and protecting the system from security attacks, privacy supervisor who can access the logs of the system to check for privacy violation, developer who implements the requests from other administrators and users.

### 4.4.8 Time ensurement module

The time ensurement module is suggested to be implemented using the Linux crontab feature which calls the PHP interpreter to execute the file in the archive subsystem that does the jobs specified in the section 3.7.3. It should also ensure that no archive will be published due to incorrect time information by comparing the system time to the a secured time source such as NTP time server pool.

### 4.4.9 Other components

Some other components must be implemented before bringing the system into a production environment such as the system needs to be secured before putting into production environment, the system policy and copyright agreement must be adapted according to the managing authorities, etc.

# 5 Result, Findings, and Future work

## 5.1 Finding and contribution

### 5.1.1 Result summary

The system architecture for the Digital Norwegian People's Archive is developed toward scalablability so that the system can scale to serve a national population of depositors and a public-available number of consumers. The system is also deemed to be platform-independent so various technologies and platforms can be used to implement. The implementation guideline for the system is also provided, which covers the important aspects ranging from user, metadata standards, open file formats, archive package structure, ownership and copyright policy, administrative tools, the basic workflows of defining archive, depositing, archiving, and consuming.

From above implementation guideline, a prototype is also made together with the roadmap toward a working system. The prototype can provide the readers a better understanding. Basic components of the system are built, which can be used as the reference for further implementation.

### 5.1.2 Findings and contributions

**Academic findings**

The main finding and contribution of the thesis is the scalable and flexible system architecture of the archive. In addition, various related issuesof the system has been identified and the suggested solutions are also given. The Document Profile concept proposed in the system is an important factor that make the system become flexible to work with many different types of document. Various different techniques are developed to scale the system such as using a distributed solution for better performance, applying data extraction techniques to give better consuming value and an extendable model for archive sharing through channels.

**Practical findings**

The system as the result of the research and design process toward a system that preserves and shares the hight potential cultural and historical value in the society. The implementation of the system will play an important role in the continuous archiving efforts in the Digital Era.

**Technological finding**

The technologies chosen to implement the prototype are free and open source. Using open source software and framework to built the system proves that open source technologies are capable to implement a large scalable and flexible system. In addition, the system also demonstrates the value of various open standards such as open file formats, METS, Dublin Core, XML, etc. Consequently, it supports the development of free and open intellectual value which should be shared freely and widely.

## 5.2 Conclusion and future work

This thesis should be seen as the first step in a progressing effort of preserving personal documents as cultural and historical heritage. In the in the digital era, digital documents are being the important supplementation to the the tradition form of documents, the sooner a public personal archive system is made available, the more risks of losing those digital documents are reduced.

There are several further research and work directions following this initial effort. First direction is the development of the specification of the system in more details, based on the above guideline, together with adding more related topic if needed. Developing a more detail prototype of the system from the system architecture and guideline will to refine and adjust the system specification toward the practical need after the first prototype is also a potential direction. More studies on the security, privacy protection, together with a structure of the administrative roles for the system are also needed.

Lobbying for public institutions to manage and run the archive is a needed action for making the system widely available. Lobbying for political willingness to make the archive become a national institution is also an important task to bring the system into practical use.

On the other hands, building different alternatives for a public personal archive system in a community that economic and political limitations exist, such as in developing country, can help the implementation of a public available personal archive becomes sooner.

# Glossary

Glossary

**AIP** Dissemination Information Package. 6, 7, 12, 30, 31, 40

**CDN** Content Delivery Network. 32, 33, 40

**DCMI** Dublin Core Metadata Initiative. 13

**DIP** Dissemination Information Package. 6, 7

**DNG** Digital Negative. 23

**DP** Document Profile. 21, 23, 25–34, 36, 38, 41–43, 47–50, 52, 53, 55–59

**DSEP** Deposit systems for electronic publications. 4

**DTD** Document Type Definition. 12

**EXIF** Exchangeable image file format. 15, 23

**HTML** Hypertext Markup Language. 9, 24, 46, 47, 50

**HTTP** Hypertext Transfer Protocol. 43

**HTTPS** Hypertext Transfer Protocol Secure. 43

**IDE** Integrated Development Environment. 12, 36

**IT** Information Technology. 1

**JSON** JavaScript Object Notation. 13, 53, 55

**LC** The Library of Congress. 32, 39

**METS** Metadata Encoding and Transmission Standard. 13, 37

**MIX** NISO Metadata for Images in XML. 15

**MODS** Metadata Object Description Standard. 15

**MVC** Model-View-Controller. 46, 48

**NISO** National Information Standards Organization. 12

**NPA** Digital Norwegian People's Archive. 1, 3, 7, 9, 12, 21, 33, 38

**OAIS** Open Archival Information System. 4, 6, 7

**OCR** Optical Character Recognition. 15

**ORM** Object-Relational Mapping. 53

**PDI** Preservation Description Information. 7, 8

**RBAC** Role-based Access Control. 42, 46, 59

**RDF** Resource Description Framework. 13, 24

**SGML** Standard Generalized Markup Language. 12

**SIP** Submission Information Package. 6, 7

**TextMD** Technical Metadata for Text. 15

**URL** Uniform Resource Locator. 24, 53, 55

**WYSIWYG** What you see is what you get. 46, 52

**XML** Extensible Markup Language. 12, 34–37, 48, 53

**XMP** Extensible Metadata Platform. 15, 23

**XSD** XML Schema Definition. 12, 34, 36, 48, 56

# Bibliography

[1] MC Amirani. "A new approach to content-based file type detection". In: *ISCC 2008* (2008). URL: `http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=4625611` (visited on 09/29/2012).

[2] *BankID*. URL: `https://www.bankid.no/` (visited on 09/29/2012).

[3] June M Besek, National Digital Information Infrastructure Program, and Preservation. *Copyright issues relevant to the creation of a digital archive: a preliminary assessment*. Washington, D.C.: Council on Library and Information Resources : Library of Congress, 2003, p. 17. ISBN: 1887334971. URL: `http://www.clir.org/pubs/reports/pub112/pub112.pdf` (visited on 09/28/2012).

[4] U.M. Borghoff. *Long-term preservation of digital documents: principles and practices*. Berlin, Heidelberg: Springer Verlag, 2006. ISBN: 9783540336402.

[5] T. Bray et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. URL: `http://www.w3.org/TR/xml/` (visited on 09/28/2012).

[6] CCSDS. *Reference model for an open archival information system (OAIS)*. 2012. URL: `http://public.ccsds.org/publications/archive/650x0m2.pdf` (visited on 09/29/2012).

[7] *CKeditor's homepage*. URL: `http://ckeditor.com` (visited on 09/28/2012).

[8] *Content Categories*. URL: `http://www.digitalpreservation.gov/formats/content/content_categories.shtml` (visited on 09/28/2012).

[9] *Copyright Act*. URL: `http://www.kopinor.no/en/copyright/copyright-act` (visited on 09/28/2012).

[10] RJ Cox. *Personal archives and a new archival calling: readings, reflections and ruminations*. Duluth, Minn.: Litwin Books, 2008, p. 418. ISBN: 978-0-9802004-7-8. DOI: `102459312`.

[11] Norwegian Ministry of Cultural & Scientific Affairs. *Legal Deposit*. 1989. URL: `http://www.nb.no/fag/for-utgjevarar-og-trykkeri/pliktavlevering/legal-deposit` (visited on 09/29/2012).

[12] Norwegian Ministry of Culture. *Acts and regulations related to Media and Culture*. 2008. URL: `http://www.regjeringen.no/en/dep/kkd/Documents/acts-and-regulations.html?id=313415` (visited on 09/29/2012).

[13] *Dodsfall*. URL: `http://www.skatteetaten.no/no/Alt-om/Folkeregistrering/Dodsfall/` (visited on 09/29/2012).

[14] David C. Fallside. *XML Schema Part 0: Primer.* 2001. URL: http://www.w3.org/TR/2001/PR-xmlschema-0-20010330/ (visited on 09/28/2012).

[15] *Feide - Felles Elektronisk IDEntitet.* URL: http://www.feide.no/ (visited on 09/29/2012).

[16] *Folket Registering.* URL: http://www.skatteetaten.no/no/Alt-om/Folkeregistrering/ (visited on 09/29/2012).

[17] *Formats, Evaluation Factors, and Relationships.* URL: http://www.digitalpreservation.gov/formats/intro/format_eval_rel.shtml (visited on 09/28/2012).

[18] David Giaretta. *Advanced Digital Preservation.* 1st ed. Springer, 2011, 532 s. ISBN: 9783642168086.

[19] Arne-Kristian Groven. *Trust strategies in long-term management and preservation of digital records.* Tech. rep. Norsk Regnesentral, 2010.

[20] ECMA Internationl. *ECMAScript Language Specification.* 2011. URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (visited on 09/28/2012).

[21] *IPTC Photo Metadata Standards.* URL: http://www.iptc.org/cms/site/index.html?channel=CH0089 (visited on 09/28/2012).

[22] *jQuery's Homepage.* URL: http://jquery.org (visited on 09/28/2012).

[23] *JSON homepage.* URL: http://www.json.org/ (visited on 09/28/2012).

[24] *JStree's homepage.* URL: http://www.jstree.com (visited on 09/28/2012).

[25] M Karresand and N Shahmehri. "File type identification of data fragments by their binary structure". In: *Information Assurance Workshop, ...* (2006). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1652088 (visited on 09/29/2012).

[26] Susan S Lazinger and Helen R Tibbo. *Digital preservation and metadata: history, theory, practice.* Englewood, Colo.: Libraries Unlimited, 2001, XXII, 359 s. ISBN: 1-56308-777-4.

[27] *LC Web Archives.* URL: http://lcweb2.loc.gov/diglib/lcwa/html/lcwa-home.html (visited on 09/28/2012).

[28] Christopher A. Lee. *I, Digital: Personal Collections in the Digital Era.* Society of American Archivists, 2011, p. 384. ISBN: 0838911552.

[29] WJ Li and K Wang. "Fileprints: Identifying file types by n-gram analysis". In: *Assurance Workshop, 2005* (2005). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1495935 (visited on 09/29/2012).

[30] *Library of Congress Standards.* URL: http://www.loc.gov/standards/ (visited on 09/28/2012).

[31] ST March and GF Smith. "Design and natural science research on information technology". In: *Decision support systems* (1995). URL: http://www.sciencedirect.com/science/article/pii/0167923694000412 (visited on 09/29/2012).

[32] CC Marshall. "Rethinking personal digital archiving". In: *D-Lib Magazine* 14.3/4 (2008). URL: http://dialnet.unirioja.es/servlet/dcart?info=link\&amp;codigo=2594837\&amp;orden=157310 (visited on 09/29/2012).

[33] *Media Ownership Act.* URL: `http://www.lovdata.no/all/nl-19970613-053.html` (visited on 09/28/2012).

[34] *Metadata Encoding and Transmission Standard.* URL: `http://www.loc.gov/standards/mets/` (visited on 09/29/2012).

[35] *Metadata Object Description Standard.* URL: `http://www.loc.gov/standards/mods/` (visited on 09/28/2012).

[36] *Model View Controller on Wikipedia.* URL: `http://en.wikipedia.org/wiki/Model\%E2\%80\%93view\%E2\%80\%93controller` (visited on 09/28/2012).

[37] PRESS NISO. “Understanding metadata”. In: *National Information Standards* (2004). URL: `http://www.niso.org/publications/press/UnderstandingMetadata.pdf` (visited on 09/29/2012).

[38] BJ Oates. *Researching information systems and computing.* 2006.

[39] Richard Ogley. *File magic numbers.* URL: `http://www.astro.keele.ac.uk/oldusers/rno/Computing/File_magic.html` (visited on 09/29/2012).

[40] *PHP’s homepage.* URL: `http://php.net/` (visited on 09/28/2012).

[41] R. Raudjärv. “Dynamic Schema-Based Web Forms Generation in Java”. MA thesis. University of Tartu, 2010. URL: `http://code.google.com/p/xsd-web-forms/downloads/detail?name=thesis.pdf` (visited on 09/29/2012).

[42] *Regulations for national registration.* URL: `http://www.lovdata.no/cgi-wift/wiftldles?doc=/usr/www/lovdata/for/sf/fd/td-20071109-1268-002.html#2-2` (visited on 09/29/2012).

[43] *The Creative Licenses.* URL: `http://creativecommons.org/licenses/` (visited on 09/28/2012).

[44] *The XML Path Language.* URL: `http://www.w3.org/TR/xpath/` (visited on 09/28/2012).

[45] Audun Vaaler and Børre Ludvigsen. *Open standards for graphic, image, sound, video on public websites.* Østfold University College, Oct. 1, 2010. URL: `http://media.hiof.no/diverse/fad/rapport_4.pdf` (visited on 09/29/2012).

[46] *Yii Framework Homepage.* URL: `http://www.yiiframework.com/` (visited on 09/28/2012).