Adaptable metadata creation for the Web of Data

Fredrik Enoksson

Doctoral Thesis No. 17. 2014
KTH Royal Institute of Technology
School of Computer Science and Communication
Department of Media Technology and Interaction Design
SE-100 44 STOCKHOLM, SWEDEN

ISBN 978-91-7595-330-4
TRITA-CSC-A 2014:17
ISSN 1653-5723
ISRN KTH/CSC/A--14/17-SE

Akademisk avhandling som med tillstånd av KTH i Stockholm framlägges till offentlig granskning för avläggande av teknisk doktorsexamen måndagen den 24 November kl. 13:15 i sal F3, KTH, Lindstedtsvägen 26, Stockholm.

Abstract

One approach to manage collections is to create data about the things in it. This descriptive data is called metadata, and this term is in this thesis used as a collective noun, i.e no plural form exists. A library is a typical example of an organization that uses metadata, to manage a collection of books. The metadata about a book describes certain attributes of it, for example who the author is. Metadata also provides possibilities for a person to judge if a book is interesting without having to deal with the book itself. The metadata of the things in a collection is a representation of the collection that is easier to deal with than the collection itself. Nowadays metadata is often managed in computer-based systems that enable search possibilities and sorting of search results according to different principles. Metadata can be created both by computers and humans. This thesis will deal with certain aspects of the human activity of creating metadata and includes an explorative study of this activity. The increased amount of public information that is produced is also required to be easily accessible and therefore the situation when metadata is a part of the Semantic Web has been considered an important part of this thesis. This situation is also referred to as the Web of Data or Linked Data.

With the Web of Data, metadata records living in isolation from each other can now be linked together over the web. This will probably change what kind of metadata that is being created, but also how it is being created. This thesis describes the construction and use of a framework called Annotation Profiles, a set of artifacts developed to enable an adaptable metadata creation environment with respect to what metadata that can be created. The main artifact is the Annotation Profile Model (APM), a model that holds enough information for a software application to generate a customized metadata editor from it. An instance of this model is called an annotation profile, that can be seen as a configuration for metadata editors. Changes to what metadata can be edited in a metadata editor can be done without modifying the code of the application. Two code libraries that implement the APM have been developed and have been evaluated both internally within the research group where they were developed, but also externally via interviews with software developers that have used one of the code-libraries. Another artifact presented is a protocol for how RDF metadata can be remotely updated when metadata is edited through a metadata editor. It is also described how the APM opens up possibilities for end user development and this is one of the avenues of pursuit in future research related to the APM.

Keywords: Metadata, Metadata Editing, RDF, Web of Data, Semantic Web, Linked Data, End User Development

Sammanfattning på svenska

För att underlätta hanteringen av olika slags samlingar skapas ofta beskrivande data om sakerna som utgör samlingen. Denna typ av data kallas metdata och används till exempel i bibliotek för att enklare kunna hantera boksamlingar. Varje bok har genom metadata blivit beskriven med ett antal attribut och egenskaper, till exempel titel och författare. Metadatan utgör således en representation av samlingen som oftast är enklare att hantera än själva samlingen och den möjliggör till exempel sökningar bland böckerna i ett bibliotek. När en intressant bok har hittats så är det även möjligt att använda metadatan för att kunna skapa sig en uppfattning om boken är av intresse eller ej. Metadata hanteras nu för tiden oftast i något typ av datoriserat system och den kan då skapas både av datorer och människor. Den här avhandlingen har ditt fokus på det senare och tar upp ett antal aspekter på denna aktivitet. Då mer och mer data produceras och görs allmänt tillgänglig så ökar även kraven på att metadata ska var lätt att tillgå och hantera. Därför har den semantiska webben, vilken även kallas Länkad Data, tagits i stort beaktande i denna avhandling.

Den semantiska webben kallas även ”Web of Data” på engelska och ska ses som en utbyggnad av den existerande webben. När metadata exponeras på den semantiska webben så blir kontexten global. Metdata som förut kanske endast används inom en organisation kan nu exponeras globalt och potentiellt få mer användare. Vad för metadata som då behöver skapas för en samling kan behöva ändras och kanske även hur den skapas. Denna avhandling beskriver skapandet av ett ramverk som kallas annotation profiles, som är en uppsättning av artefakter som utvecklats för att möjliggöra en anpassningsbar miljö för metadataskapande för den semantiska webben. Den viktigaste artefakten går under namnet Annotation Profile Model (APM), en informationsmodell konstruerad så att ett datorprogram kan skapa ett metadata formulär från en instans av modellen. En sådan instans kallas annotationsprofil, vilket kan ses som en konfigurationmekanism för en formulär-baserad metadata editor. En annotationsprofil kommer därför att göra det möjligt att ändra vad för metadata som kan skapas och redigeras, men utan att behöva ändra koden för datorprogrammet. Två kod-bibliotek som implementerar APM har utvecklats vilket utvärderats både internt inom min egen forskargrupp men också genom att intervjua utvecklare som har använt det i de verktyg de utvecklat. En annan artefakt som presenteras i denna avhandling är ett protokoll för hur RDF kan uppdateras över webben när ändringar görs till metadata uttrycket genom en formulär-baserad metadata editor. Hur APM öppnar möjligheter för så kallade slutanvändar-utveckling (End user development, EUD på engelska) beskrivs också och ses som en av de möjliga vägar att gå vidare med framtida forskning relaterad till APM.

Acknowledgements

I want to start by thanking my two supervisors Ambjörn Naeve and Olle Bälter. Ambjörn for always being a big source of inspiration and for providing ideas and generating projects that have helped to develop my thinking, both with regard to research and to life in general. I want to thank Olle for his structured guidance and help along the path that lead up to this thesis, for his patience, his positive mindset and for always pushing me forward.

The research that is presented in this thesis has taken place within the Knowledge Management Research Group (KMR), headed by Ambjörn. From this group I would first of all like to thank Matthias Palmér for always being contagiously enthusiastic, a source of inspiration, and for always having time for discussions. I would also like to thank Hannes Ebner for being able to answer all my difficult technical questions, being meticulous about most of the things he does and, last but not least, for having such a great sense of humor. Erik Isaksson is yet another member of our research group, and I would like to thank him for all our long discussions that have helped me in sharpening my thinking. A big thank you also to the other core members of the KMR-group, Mikael Nilsson and Fredrik Paulsson.

Much of the research that lead up to my licentiate thesis was carried out at what was known at that time as the Uppsala Learning Lab (ULL) at Uppsala University. I want to thank the former managing director of ULL, Mia Lindegren for providing a platform for the projects I have been involved with at ULL and I want to thank Anette Vikberg for all the help with the managerial issues in those projects . Furthermore, a big thank you also to the great group of people that worked at the Uppsal a Learning Lab during my time there. The projects at ULL were to a large extent financed by the projects LUISA and HematologyNet, and I want to acknowledge the EU-commission for the funding of these projects. I also want to thank the people that I have been working with in these projects, especially Eva Hellström-Lindberg, Thom Duyvené de Wit, Carin Smand and the rest of the great staff of the European Hematology Association executive office.

The research carried out after my licentiate thesis has to a large extent been financed by the ECE school here at KTH, and I would like to acknowledge and say thank you to Mats Herder for providing me with this opportunity. I would also like to thank my colleagues at the ECE school and the KTH library, especially the head of the PI unit, Elisabeth Mannerfeldt, and the rest of my colleagues at the PI unit.

I also want to thank the School of Computer Science and Communication (CSC) at KTH for financing part of the research and making it possible for

me to finish up this thesis as well as my licentiate thesis. I want to especially acknowledge professor Ann Lantz, the current director of the MID department and professor Nils Enlund, the former director of the media department. CSC is also the school at KTH where I have been enrolled as a PhD, student and I want to thank all my colleagues there and also express my gratitude to all of my fellow PhD students (both current and former). Among them I would like to mention Filip Kis for our work together on the last paper included in this thesis, and Björn Hedin for the discussions about methodology during the summer months of 2014 .

Most of all I would like to thank my family for just being so great! This includes my parents Bengt and Gerd, my older sister Linda, my oldest younger sister Cecilia and her partner Rafael and your delightful child Isabella, my youngest sister Edina and last but not least Maria and her partner Fredrik and the always delightful little Vincent.

Papers included in this thesis

Paper 1

Thalmann, S. & Enoksson, F. (2007) An approach for on-demand e-learning to support knowledge work. In Tochtermann, K. & Maurer, H. (Eds.): *Proceedings of I-KNOW 07*. 7th International Conference on Knowledge Management, Graz, Austria, September 5-7, 2007. Journal of Universal Computer Science. pp. 289-29

Paper 2

Palmér, M., Enoksson, F., Nilsson, M. & Naeve, A. (2007) Annotation Profiles: Configuring Forms to Edit RDF. In *proceedings of the International Conference on Dublin Core and Metadata Applications*, Singapore, August 28 – 31, 2007. pp. 10-21

Paper 3

Enoksson, F., Palmér, M. & Naeve, A. (2007), An RDF modification protocol, based on the needs of editing Tools. In Sicilia M.A., Lytras, M.D. (Eds.): *Metadata and Semantics*, Post-proceedings of the 2nd International Conference on Metadata and Semantics Research, MTSR 2007, Corfu Island, Greece, 1-2 October 2007.

Paper 4

Enoksson, F., Naeve, A. & Hellstrom-Lindberg, E. (2011). Using a hematology curriculum in a web portfolio environment. *Knowledge Management & E-Learning: An International Journal (KM&EL)*. 3 (1). pp. 84-97.

Paper 5

Enoksson, F. & Bälter, O. (n.d.) The activity of human metadata creation and the Semantic Web. Paper accepted for publication in the *International Journal of Metadata, Semantics and Ontologies* (ISSN: 1744-2621)

Paper 6

Enoksson, F. & Kis, F. (n.d.) Towards End-User Development for Metadata Creators. Paper submitted for peer-review.

Table of Contents

1. Introduction

As we humans try to live our lives we interact with the world around us in different ways. One thing we do is to establish some kind of order in our daily lives, for example in our workplace and at home. What kind of order that is does not necessarily need to make sense to outsiders. This could be how a collection of books are organized into bookshelves, which can appear as a random ordering to others, but might make perfect sense to the person that put the books on the shelf. The order of the books might represent an externalization of that individual's mind and way of thinking. Hence, order can be something individual. However, many of us prefer to make use of a clear and well-defined order that is easy for others to understand, and the reason can be to get a satisfactory feeling of having things put into a good order. This can be a part of their work or as part of a hobby or dedication to some cause. An excellent example of the latter can be found in the book “High Fidelity” (Hornby, 1995) that is about the fictional character Rob Gordon who works in a record shop. He is also a collector of vinyl records and as part of his hobby often re-organizes his records according to different principles. He engages in this activity not only because he likes to sort his collection in new ways, but also because of the love for music and the media that they are recorded on. He also seems to sort his collection when other things in his life are chaotic.

When it comes to collections that are meant to be used by a wider audience than your own record or book collection, the ordering is preferably done according to principles easily understood by others. Common approaches for ordering the things in the two example of collections above are alphabetical order of the author of a book or the group/artist of a record. This ordering is however not always good enough as someone searching the collection might only know the title of the book or record. Simply put, the right kind of information to search according to the ordering principle is not available. A physical collection is limited to be sorted according to one ruling principle, and to make the collections possible to search in other ways alternative approaches have been developed. Written (physical) index cards, containing descriptive information about each thing in the collection, is one solution that has been used in libraries. Normally, each thing in the collection is represented with a card, and this set of cards will represent the whole collection. Furthermore, each such set of cards allows sorting according to one principle. For example, where a library typically would have one set of cards each for author, title and subject-classification. Even though the cards have the same limitation to only be “sortable” according to one

principle, they are in many ways much easier to deal with than the whole collection. The information put on those cards we today refer to as metadata, and it is commonly stored on computer systems. Metadata in a digital form enables simpler management through the use of computers, for example indexing and searching in many different ways that do not rely on a single sorting principle.

1.1 Metadata, what is it good for?

As expressed above, metadata can be found e.g. in libraries and the main ideas behind metadata are often easy to communicate to anyone. However, between different domains and communities the exact meaning of the term "metadata" can vary. Domains focused around digital data look upon metadata as describing that data, like the size of a file, or the amount of successfully transmitted data-packets through a digital network. The metadata itself will in many of those cases be considered more interesting than the actual things being described, as the things themselves might not primarily be aimed at human consumption. In a library, metadata is used to describe books, i.e., things that are intended for human consumption. The metadata that is created for things in such a collection constitutes a representation of that collection. In this way the collection can be managed by handling that representation instead. This includes various search functionalities that can be enabled as well as possibilities to sort according to different principles. Also, the metadata description of a thing will enable the user of a collection to judge if that thing, possibly found through a search, is of interest or not.

Metadata is commonly described or defined as "*data about data*", which makes it easy to communicate the concepts and ideas behind the term. However, with a strict interpretation of this definition one can argue that metadata can only be about data and nothing else. Going further from that strict interpretation, one can continue to also look at a definition of data. Descriptions or definitions of data as "*individual pieces of information*" serve the purpose of being generic and to be widely accepted across many communities, but they can be too generic to be useful in certain communities. Using alternative definitions could serve a domain or community well, but that same definition would not automatically be useful in another community. From the definitions mentioned it can be concluded that metadata is information. And, in general you can say that how strict a term like data or metadata is to be defined in different communities should be guided by the usefulness of such a definition, i.e., what kind of purpose would an alternative definition fill? It is then of importance to state how the term is used and if its definition deviates totally from the widely accepted generic definition.

The most suitable definition of metadata in the context of this thesis is "*Descriptive data about identifiable things*" (Nilsson, 2010, pp. 11). First of all, this definition clearly says that metadata can be about any kind of thing, not just data, and that fits the view of metadata presented in this thesis. The definition also says that metadata is data that needs to be descriptive of the thing(s) that it intends to describe. Moreover, according to Nilsson, the use of the word data is to be interpreted as being machine-processable and also, to some extent, possible to interpret by machines. With this definition, metadata cannot be a picture as it would only be possible for humans to interpret. This is an important aspect when dealing with metadata interoperability and this definition was developed with that aspect in mind. That the thing being described is possible to identify is also one important aspect, since it should be possible for a machine to interpret if two things being referred to are in fact the same thing or not.

To conclude, metadata is useful for creating order in a collection of some kind. Metadata about the things in a collection provides a representation of that collection. A person who wants to search the collection can use the metadata for that purpose, and when it is the result of such a search, the metadata can be used to judge if a certain thing is interesting or not. An example of this can be a student searching for books about a certain topic in a library. The metadata enables that search and for the books that was found in the search the student can, with the help of the metadata, decide which of the books that are worth reading. For the persons that manage collections, like the books in a library, the metadata provides a representation of the collection that is often easier to deal with than the whole collection. In order to make use of the benefits that metadata provides it has to first be created.

1.2 Research objective and research questions

This thesis will focus on the human creation of metadata. When metadata is created it can be assumed that certain goals and purposes of how the metadata will be used have been considered. While for other goals and purposes, suitable metadata might not exist. Also, the metadata description of a thing can be viewed with several aspects in mind, but to cover all the possible future aspects of a thing in the metadata will never be feasible. So, when a thing is to be described with other aspects, the necessary metadata for discovering it has to be created.

How one type of thing is described will also change over time, and that means that metadata will eventually have to be updated. Hence, the metadata description about a thing will never be finished as it might have

to be extended. With the advent of a Web of Data, also referred to as the Semantic Web or Linked Data, a global scope has been established where the metadata both can be distributed and then discovered and consumed. Metadata stored in an isolated database can therefore be exposed into an environment where it is possible to link to other metadata. Such metadata descriptions exist on a *semantic co-existence* level and when the possibilities to link between metadata descriptions across the web a level of *semantic collaboration* has been reached. The levels of semantic coexistence and collaboration were introduced in Naeve (2005) and will be discussed in more detail in chapter 3. Part of the research objective (presented below) has been to create a technical environment for editing metadata that is exposed as a part of the Semantic Web, while at the same time it can easily be adapted.

The research presented in this thesis started with a project called LUISA (see below) where metadata editors for learning objects were developed. As the requirements upon what metadata elements that should be possible to edit was not fixed, a flexible approach was taken. This initial project shaped the research objective and questions presented below, with a perspective that assumes that the metadata is meant to be exposed on a level of semantic coexistence and collaboration, where the scope is global. Under this assumption the objective can be summarized as: *Investigate how an environment for human metadata creation can be constructed that is adaptable with respect to what kind of metadata that can be created.* This objective requires an understanding of how metadata is created by humans and includes the suggestion of possible solutions. Therefore the two main research questions have been formulated as:

Research question 1:

What are the main characteristics of the human activity of creating metadata?

Research question 2:

How can we design an adaptable environment for metadata creation, that works for the Web of Data?

When the work presented in this thesis started, the primary focus was on the technology around metadata and the Semantic Web. Thus, the primary aim at that time was to develop tools that supported the Semantic Web vision (Berners-Lee, Hendler and Lassila, 2001). This part of the work has been presented in my licentiate thesis (Enoksson, 2011). Some of that work is also included here as it relates to research question 2.

In the part of the research I have done after my licentiate thesis, other aspects have been considered and incorporated. The technology that had been constructed needed to be evaluated on how it was perceived by people having other functions or roles. Also, a deeper understanding for how metadata is actually created "in practice" had to be gained, i.e. what are the purposes and goals for the activity of creating metadata? This is reflected in research question 1.

1.3 Scope of this thesis

This thesis includes research I have carried out about metadata from 2006 up until 2014. The main approach is design-oriented, as described in chapter 2.2, i.e. artifacts have been constructed to solve problems related to metadata. The Annotation Profile Model (APM) is the main artifact constructed and described in this thesis. It has been constructed in response to answer the second research question and is covered in chapter 4.

In the work leading up to this thesis, the kind of metadata that has been considered concerns description of things that are primarily considered to be consumed or used by humans. In order to create useful metadata for those kind of resources, the context around them has to be understood, including the fact that the metadata is created by a human. The research has been carried out within several projects, and they are briefly described below:

- **LUISA (Learning Content Management System Using Innovative Semantic Web Services Architecture)** – A project funded by the European Commission within Framework Program 7. The consortium members of this project came from both industry and academia. The project was aimed at utilizing Semantic Web Services in order to suggest individual learning paths for different learners. This required an environment where metadata editors could be adapted for different situations and my main involvement in this project was related to developing the environment that lead up to the Annotation Profile Model.
- **Organic Edunet –** This project was also funded by the European Commission as a targeted project of the eContentplus program. It was aimed at organizing learning resources about organic farming into a semantic network of distributed repositories. The resources were exposed as a part of the Semantic Web through the use of a system of web portfolios

called Confolio (web-client) and a server back-end called SCAM. Both SCAM and Confolio existed in earlier versions but were developed further in this project[1]. These developments involved usage of the APM.

- **HematologyNet** – Another project funded by the European Commission through the Leonardo da Vinci Lifelong Learning Program. This project made use of Confolio and SCAM to structure learning resources in hematology. In this project a curriculum for hematology was included and expressed as a set of learning goals. The curriculum was used in two ways: 1) as metadata about a learning resource in order to express the intended learning goal, and 2) to express the competence (in hematology) of a learner (i.e. user of the system). This was expressed as metadata about the learner in two ways, the current competence and the desired competence, thus defining a "learning gap" in such a way that suitable learning resources could be found that could close the gap. This project provided another case where the APM could be evaluated. It was special in the respect that metadata had to be created both for the learning resources and for the persons using Confolio. More details about the project can be found in paper 4.
- **DISKA (Digitala Semantiska Kulturarvs Auktoriteter)** – A project within the cultural heritage domain, financed by the Swedish government agency Vinnova. The first part of the project was to conduct an inventory of so called *authority lists* at 24 government funded agencies within the cultural heritage sector in Sweden (i.e. museums, archives, libraries etc). In the second part of the project, a subset of lists from the inventory was selected to be exposed as Linked Data. An authority list is not considered to represent the collection, but instead provides a parallel reference list of values that is related to the collection. The most common example of authority lists are the list of authors in a book collection. The main idea in the DISKA project was to connect cultural heritage institutions through the commonalities found in the authority lists. How that could be achieved through the use of Linked Data was and important aspect of the project, which reflected a situation where more and more organizations want to expose their data as Linked Data. This presents a larger problem that could not be solved within this project, but it is an interesting problem for further research.

1 Do note that Confolio nowadays goes under the name EntryScape and SCAM under the name EntryStore. Ebner & Palmer (2014) describe the latest developments.

As can be seen in the description of the projects above, the research discussed in this thesis has been situated within the context of several domains, but within projects that mostly had their focus on learning.

In the work done within these projects special consideration has been taken of the idea of metadata interoperability. One aspect of metadata interoperability considers the possibility to combine metadata elements originating from two or more standards. This becomes important when dealing with learning resources as they can be of many different kinds, like a movie, a document or even a statue in a park. And, these resources can vary in their metadata descriptions, but they still need to be combined with metadata describing the learning aspects. Consider the example where a movie is found in a collection that is considered useful for learning purposes. The metadata description of the movie has to be extended with the learning perspectives, in order to make it discoverable in that way too. This could require a combination of two metadata standards, which is not always done effortlessly and there is no guarantee that two or more standards can be used together. However, if the standards involved are interoperable with each other, this can be achieved (see Nilsson (2010)).

A personal reflection I did fairly early, in the research leading up to this thesis, was that it seemed as if metadata standards had been developed with little or no consideration taken to the interoperability aspects. This is natural since metadata collections have historically been created in *semantic isolation*[2]. Metadata has traditionally been isolated in databases that were separated from each other, where each database only contained one or a few types of resources and only one standard was needed. Also, metadata standards are developed by people who are involved in a specific domain and therefore aim to fit the metadata to the need and purposes found there. As the people developing metadata standards are expert in their domain they are expected to know what should be possible to express with the standard. However, they are not necessarily experts on how the metadata should be expressed. As described in chapter 3 it cannot be assumed that only one metadata standard will cover all needs and requirements for metadata descriptions of a thing. Thus, a good practice is to choose, when possible, metadata standards that can be combined with other metadata standards. According to Nilsson (2010) metadata standards can be interoperable if they adhere to a common data model. Nilsson also argues that the best fit for a common data model that exist today is the data model of RDF[3]. Metadata expressed in RDF is in

2 For a description of this term see section 3.4.1
3 http://www.w3.org/RDF/

focus in this thesis as there seems to be no better alternative for a common data model. RDF is also the framework that has been developed to express data on the web, and the envisioned Semantic Web. Linked Data is one part of the Semantic Web that aims to interlink data across databases and enable discovery of data, metadata and resources across the WWW. Further discussions about metadata, interoperability, Semantic Web, Linked Data and related issues can be found in chapter 3.

1.4 Outline of this thesis

The chapters of this thesis are arranged in the following way. An overview of research related to metadata is presented in the beginning of chapter 2, with the intent to frame the research presented in this thesis. The two methodological approaches to research that have been utilized, the constructive research approach and the design science research approach, are presented in section 2.2. Section 2.1 gives an overview of activity theory, which was used in paper 5, in dealing with research question 1. The beginning of chapter 3 is aimed at providing a background of metadata, but it also includes a description of other roles that are involved. The problem of metadata interoperability is discussed in section 3.3 followed by section 3.4 that covers metadata being exposed at a global scope, like the Semantic Web and Linked Data. The artifacts developed are described in chapter 4. This includes the main artifact, the Annotation Profile Model. Chapter 5 contains a description and evaluation on how the two methodological approaches, constructive research and design science research, have been used. This chapter also includes the main result of paper 5, that was aimed at answering research question 1. Chapter 6 is dedicated to a discussion and a reflection upon the research performed and how it can be taken further.

As the work presented here is a continuation of the work I presented in my licentiate thesis (Enoksson, 2011) parts of the text from that thesis have been included here. Notably the sections 3.2-3.4 and sections 4.1-4.4. However, the texts have been reworked so they are not plain copies.

1.5 List of abbreviations used

AIO – Abstract Interface Object

AJAX – Asynchronous JavaScript and XML

APM – Annotation Profile Model

AT – Activity Theory

AUI – Abstract User Interface

CFI – Choice Form Item

CIO – Concrete Interface Object

CR – Constructive Research

CUI – Concrete User Interface

DCAM – Dublin Core Abstract Model

DCAP – Dublin Core Application Profile

DCMES – Dublin Core Metadata Element Set

DCMI – Dublin Core Metadata Initiative

DSP – Description Set Profile

DSR – Design Science Research

EUD – End User Development

FTM – Form Template Model

FUI – Final User Interface

GUI – Graphical User Interface

GFI – Group Form Item

GPM – Graph Pattern Model

HCI – Human Computer Interaction

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IEEE LOM – Learning Object Metadata (LOM) according to the Institute of Electrical and Electronics Engineers (IEEE)

IETF – Internet Engineering Task Force

IRI – Internationalized resource identifier

JSON – Javascript Object Notation

LOM – Learning Object Metadata

MODS – Metadata Object Description Schema

OWL – Web Ontology Language

PDF – Portable Document Format

RDF – Resource Description Framework

RDFS – Resource Description Framework Schema

REST – Representational State Transfer

SW – Semantic Web

SDQ – SPARQL Describe Query

SPARQL – SPARQL Protocol and RDF Query Language (recursive abbreviation)

SPARUL – SPARQL Update Language

TEL – Technology Enhanced Learning

TFI – Text Form Item

URI – Uniform Resource Identifier

UI – User Interface

WWW – World Wide Web

W3C – World Wide Web Consortium

XML – Extensible Markup Language

2. Research in relation to metadata

Metadata is created to provide good and useful descriptions about things. How good and how useful the descriptions actually are can be estimated in the evaluation of the metadata quality, which can include several aspects (described further in chapter 3). Much of the research and development related to metadata includes some direct or indirect aspect of metadata quality. The different approaches to metadata quality concerns both theoretical and practical matters, and a rough categorization of the metadata research, related to the research presented in this thesis, can be found below.

- **Automatic generation of metadata** – How can useful metadata be created through new technologies and applications in an automatic fashion. The project AMeGA (Greenberg, Spurgin and Crystal, 2005) was formed in order to evaluate the current automatic metadata generation functionalities and also to investigate what aspects of metadata creation that is most suitable for automatic creation. Rodriguez, Bollen and Van De Sompel (2009) describes automatic generation of metadata through associative networks of described resources and Matsou and Ishizuka (2004) describes an algorithm for extracting keywords in a document.
- **Metadata interoperability** – The possibility to compare, link between and even merge metadata for two or more collections, and to be able to combine metadata elements from various standards in a single metadata description. Questions related to interoperability was early addressed by Waugh (1998) and more recently by Nilsson (2010).
- **Construction and development of vocabularies (and ontologies)** – This kind of development is for example done within a community that has a need to describe things (within that community) in certain specific ways. As a part of their work they might need to create new vocabularies or they might need to further develop existing ones. Gangemi, Fisseha, Pettman and Keizer (2002) describes the development of an ontology in order to describe resources in the fishery domain and Sánchez-Alonso et al. (2008) the engineering of an ontology for organic agriculture. A vocabulary consist of a set of concepts that are used as elements or values in a metadata description. How the

concepts of the vocabulary relate to each other can also be expressed, as well as how they relate to other concepts and other vocabularies.

- **Development of new models to represent metadata** – Models on different levels to represent metadata, for example the CIDOC Conceptual Reference Model (Doerr, 2003) and the Dublin Core Abstract Model (Powell, et al., 2007).
- **Visualization of and interaction with metadata** – Research related to how metadata can be manipulated, for example through some kind of metadata editor, and also how metadata can be visualized and presented. See for example Klerx et al. (2004) where metadata is visualized to provide access to learning resources, and Greenberg et al. (2004) who describes the evaluation of a metadata creation application.

Note that these categories are not mutually exclusive and are only meant to give a context for the research described in this thesis. A more thorough list of categories can be found in Hunter (2003).

Research about metadata is often driven and motivated by current or foreseen needs. For example, automatic creation of metadata is useful when little or no effort can be put into manually creating metadata. Automatic metadata creation can also assist a metadata creator for creating certain parts of the metadata, while the time saved not having to create that part can be spent on the parts that require more careful consideration. The possibility to automatically create metadata is often the result of research carried out within different disciplines of computer science, for example Rodriguez, Bollen and Van De Sompel (2009) that creates associative networks of resources, where metadata that is associated with resources that have a rich metadata description is propagated to resources with poor metadata descriptions. Another example is found in Matsou and Ishizuka (2004) on how keywords can be extracted from text-documents.

Metadata interoperability can be considered on several levels. For a basic level it can be to use standard set of elements and terms, thus become interoperable with other metadata using that same set. If the metadata is to be consumed only by humans, this can be considered to be good enough. A level of machine semantic interoperability (Nilsson, 2010) is however needed when machines are to process and consume the metadata, for example, when a machine should interpret the full metadata description about a thing when elements from two or more standards have been used, and also compare metadata descriptions originating from different sources. The different levels of metadata

interoperability are described further in chapter 3. An actual need to combine elements from two or more metadata standards appears when resources are being reused and re-purposed in new situations. A typical example of this can be found in the Technology Enhanced Learning (TEL) domain, where the term *Learning Resource* (alternatively *Learning Object*) is often used. A Learning Resource is considered to be a thing that is or can be used in learning situations. Moreover, since a Learning Resource can basically be any kind of resource a new combination of metadata elements is needed when the existing metadata for the resource has to be extended with metadata about the learning aspects, that reside in other metadata standards. The resource can of course also be further re-purposed into a situation where even more metadata elements from other standards can be added "iteratively" or "recursively" in order to improve the metadata quality.

A natural focus when developing new metadata standards is to provide a good set of metadata elements and values that will describe a certain type of thing in a useful way. As described above one metadata standard is not always enough to provide a good enough description, which would need a combination of metadata elements from two or more standards. Two metadata standards can of course be adapted to be interoperable with each other whenever the need occurs. But, instead of having to arrange this in a post-hoc manner, it can be arranged in a pre-hoc manner, as suggested in the doctoral thesis by Nilsson (2010). This means that metadata standards are developed to be interoperable with each other "by default". As Nilsson (2010) argues, a pre-hoc arrangement requires that the metadata standard adheres to a common data model on how the metadata is expressed. The metadata interoperability problem has also been addressed by Waugh (1998) who also describes a possible solution.

The creation of new vocabularies and ontologies includes new kind of concepts and constructs that will be used by a community. The intention is to provide better ways to describe an artifact. As mentioned above one example can be found in Gangemi, Fisseha, Pettman and Keizer (2002) where an ontology has been developed to be used for the fishery domain. To be considered useful, the vocabulary or ontology has to be used in practice by the intended community and through the uptake the validity can be evaluated. A thorough development of a vocabulary or ontology should also include an analysis of the feedback that can be gathered from the community followed by an analysis that should lead to an update of the vocabulary or ontology.

Research on visualization and interaction with metadata can for example be concerned with how to present large sets of metadata and how such presentations can be utilized for searching (Klerkx et al., 2004). This kind

of research usually involves observations about the interaction that are evaluated both in a quantitative and a qualitative manner. A quantitative evaluation can include a comparison on how many errors in the metadata that can be traced back to using a certain interface compared to another. A qualitative evaluation can include methods used within HCI research (Lazar, Feng and Hochheiser (2010)), for example interviews and user observation on how people interact with the metadata creation tool. An example of this can be found in Greenberg et al. (2003) who describe an evaluation upon the usability of a metadata application.

The research presented in this thesis falls into the category of *visualization of and interaction with metadata* presented above. It has been carried out in situations where *metadata interoperability* has been considered important. However, the research is also closely related to the category *Development of new models to represent metadata*. And, much of the thesis deals with metadata creation done by humans, that is strongly related to research on *Automatic generation of metadata*. Moreover, plans for future research include how to also support *Construction and development of vocabularies* in the process where metadata is created.

In the rest of this chapter the theories and methodologies that have been used in this thesis are presented. Activity theory is the theory used in the attempt to answer the first research question. The approach taken to answer the second research question is through constructing artifacts that has been iteratively evaluated. This approach is characteristics of *design science research* and especially *constructive research*.

2.1 Metadata creation and Activity Theory

One of my assumptions is that the explicit goal of metadata creation is to create metadata that as well as possible describes the resources at hand. Moreover, in the research carried out leading up to this thesis a better understanding of the practice of metadata creation was considered important, since my contributions to answer of research question 1 are focused on understanding this practice. Furthermore, my contribution to research question 2 (described in chapter 4) has bearing on the work situation for people creating metadata. Therefore, in order to gain descriptive knowledge about their practice, a number of interviews have been conducted with people that were at that time or had earlier been working with creating metadata. When the study was prestructured, activity theory (AT) was utilized as an underlying framework and it was also used when the data from the interviews was analyzed. The details of the study can be found in paper 5, but AT is here briefly described here as

one of the theories that I have used and it is presented with some of the result of the study in order to be able to give examples.

Activity theory originates from an attempt to establish a Marxists psychology in the Soviet Union. The original ideas were developed by Vygotsky (1962) and Leontév (1978) and they have been developed and extended further over the years. AT was introduced into Scandinavian psychology by Engeström (1987) and it has also been introduced and gained popularity within the field of Human-computer interaction (Bödker, 1992; Kaptelinen & Nardi, 2006).

Activity theory aims to get an understanding of the human mind and how it relates to the rest of the world. Humans engage in activities that include the things in the world and it is considered to be impossible to fully understand these intricate relationships if they are separated from each other. Hence, they have to be studied together. This study is carried out by focusing on activities, which are seen as meaningful interactions with the world. Kaptelinin, Nardi and Makulay (1999, pp. 28) describe activity theory as resting on the ideas that "*(1) the human mind emerges, exists and can only be understood within the context of human interaction with the world; and (2) this interaction, that is, activity, is socially and culturally determined*".

Each activity contains a subject, that is the human, that is working with an object, that is something in the world. Kaptelinin (2005) points out that Leontév used the Russian word *predmet*, which in many cases translates to object in English. Kaptelinin further say that predmet "*often means the target or content of a thought or an action"* (Kaptelinin, 2005, pp. 6). Thus, the object can be something that should be carried out or achieved. The activity is mediated through a mediating artifact, often referred to as a tool. Furthermore, an object has a relation an *objective* of an activity, which are the reasons why a subject is carrying out the activity. Furthermore, the same object does not need to correspond to the same objective. Engeström (1987) extended the description of activities by taking more of the surrounding context into a so called activity system, that can be seen in figure 1 below.

A *subject*, i.e. a human works with some kind of thing in the world referred to as an *object*. The activity will hopefully then result in some kind of outcome. Seeing metadata creation as a human activity, the object, when interpreted as a translation of the Russian word predmet, is to create metadata. The objective of that activity seemed from the result the study in paper 5 to correspond to enabling search-possibilities for the actual and potential users of a collection. The activity is carried out using a metadata editor, which corresponds to an instance of *tools* in figure 1. And in activity theory, a *tool* can be any thing that can be used to support

the *subject* in achieving the *objective*. The *community* consists of the other roles involved in the activity, but are not necessarily aware of the *objective* of the activity or have a full understanding of the goal. In this study, the data indicated that the community can be other people creating metadata. The community is related to the objective through the *division of labor,* i.e. what roles different persons in the community have. The *subject* is related to the community via the *rules* of the activity that can be explicit or implicit norms. In metadata creation an explicit norm would be a metadata standard and an implicit norm could be a common understanding on how to interpret the value of a metadata element when a standard is unspecific.

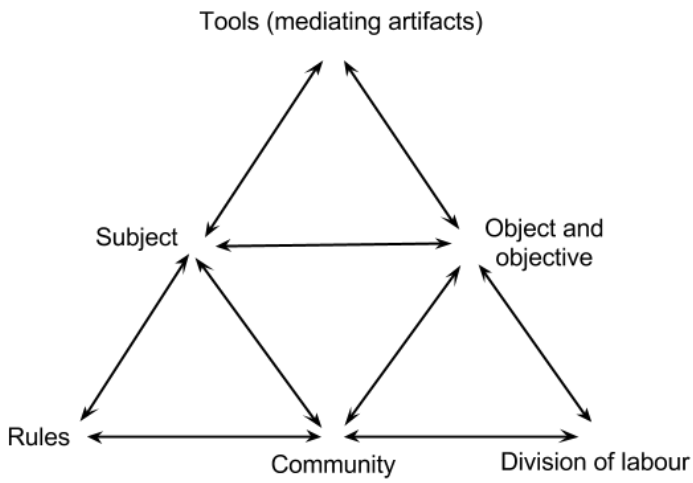

Figure 1: Model of an activity system.

In the examples given above some of the results of the study presented in paper 5 is revealed. How activity theory became useful when searching for an answer to research question 1 is discussed in chapter 5.1.

2.2 Constructive Research and Design Science Research

In this chapter two approaches to science and research is presented: the Constructive Research approach (CR) and the Design Science Research approach (DSR). Piirainen and Gonzalez (2013) concluded that CR and DSR have much in common and both are described here (in an overview manner) as both approaches have been used in the research leading up to

this thesis. This chapter will end with a discussion on the similarities and differences of DSR and CR.

Natural science research aims to explain and predict different phenomenas in nature and the result will eventually produce theories about how the world around us works. The result of this kind of research is descriptive. In contrast, both DSR and CR include the building of an artifact or construction early in the research process, that is intended to solve a relevant (practical) problem. March and Smith (1995) states that design science is aiming to "*create things that serve human purposes",* where the things (artifacts or constructions) that are created are artificial in the sense that they are new and innovative constructions or artifacts created by humans. In the construction process knowledge will be gained on how things should be created or designed (Dodig-Crnkovic, 2010) to solve the problem, but also to generate knowledge on how to approach the problem. This will naturally be knowledge that is of a prescriptive nature. Also, parts of the evaluation can be done on how the introduction of the new artifact or construction improve (or worsen) the current situation? However, knowledge that are of a more descriptive nature can also be gained. For example, by studying the new situation that might occur once the artifact is used. Thus, the evaluation part of the process will generate knowledge on how well the artifact works and can also include investigations of phenomenas that occur once the artifact is utilized.

2.2.1 Constructive Research

Constructive research is described in Kasanen, Lukka and Siitonen (1993) and also in Lukka (2003) as a methodology where innovative constructions are created with the aim to solve a real world problem. Lukka (2003) describe CR in the following way:

> *"The constructive research approach is a research procedure for producing innovative constructions, intended to solve problems face in the real world and, by that means, to make contribution to the theory of the discipline in which it is applied. The central notion of this approach, the (novel) construction, is an abstract notion with great, in fact infinite, number of potential realizations. All humans artifacts - such as models, diagrams, plans, organization structures, commercial products and information systems designs – are constructions"*

CR is further described to commonly originate from a situation within a company or organization where the construct to solve a particular problem is developed, demonstrated and initially used within that

organization. The research procedure also includes a theorizing part on how the problem could be solved in principle, i.e. also outside of the situation where the construct has been developed and used. CR is (by Lukka (2003) and Kasanen et al. (1993)) presented within the context of management accounting, that is described as a field which is applied and practical. It is also argued that CR is common in technical sciences. Dodig-Crnkovic (2010) further argues that this method is also often used within computer science research, but rarely included in the methodological discussion. Kasanen et al. (1993, pp. 246) divides the constructive research process into the following phases (where the order may vary):

1. *Find a practically relevant problem which also has research potential.*
2. *Obtain a general and comprehensive understanding of the topic.*
3. *Innovate, i.e. construct a solution idea.*
4. *Demonstrate that the solution works.*
5. *Show the theoretical connections and the research contribution of the solution concept.*
6. *Examine the scope of applicability of the solution.*

A depiction of CR can be found in Figure 2. This figure is a modified version of the depiction found in Kasanen et al. (1993), but here also expanded to include an iterative process.

As expressed in phase 1 above, the problem considered should be relevant and also have research potential. Lukka (2003) argues that CR is aimed at shortening the gap between research and practice in management accounting so that if the research process is successful it can bring something directly back to the participating organizations. Phase 2 is aimed at finding out what kind of knowledge that exists around the current problem. Has someone attempted to solve this problem before, and if so, what was their approach? How successful were those previous attempts? How have similar problems been addressed, and how successful were they? What kind of relevant theories exist? Those are the kind of question necessary for this phase. During phase 3 the building of the construct takes place. This naturally becomes the most critical phase, if the design of the innovative construct fails there is little reason to continue (at least from a practice point of view). However, the design can often be performed in an iterative manner with correction of errors found

when demonstrating the construct. This should be carried out during phase 4 and iterated until the construct solves the problem addressed in a satisfactory way.

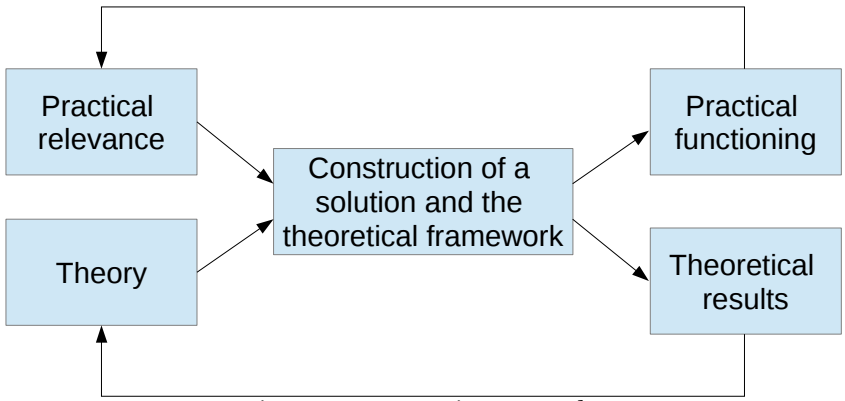

Figure 2: Constructive Research.

Phase 4 might also be critical because the new construct might have to be "sold" to the organization that is meant to use it in practice. Thus, it is important to make sure that the organization in question can put enough effort into this phase. Phase 5 is also devoted to finding the connections with and contributions to research. This includes an analysis of how well the construction has solved the problem at hand in the organization and to what extent the construct would work in other organizations with the same or a similar problem. It is also important to find the connections to existing theories and also, if appropriate, to create new ones. In phases 3-5 the artifact is both constructed and evaluated and it is here that new knowledge can be generated about the problem and also on how to construct a solution for that kind of problem. But, even if the construct fails to solve the problem at hand, new knowledge should still have been generated in the process (Lukka, 2003 and Dodig-Crnkovic, 2010). One important indicator when examining the scope (phase 6) is the uptake of the construct in other organizations. However, the reason for the uptake might depend on many things, for example if the researcher has been good at "selling" the construct, which might in itself be rather poor but in spite of this solved the problem better than any existing solutions. On the other hand the solution might be very strong but it was constructed within a domain where changes take a long time.

2.2.2 Design Science Research

What in CR is referred to as a construct is referred to as an artifact in DSR (Piirnanen and Gonzalez, 2013). The term artifact will be used for both these concepts in the rest of the in this text. DSR will here be described within the context of information systems research, where it has been situated by several authors (see for example Hevner, March, Park and Ram (2004); March and Smith (1995); Iivari (2007); Bratteteig (2007)). Hevner et al. (2004) state that much of the research in the information systems can be characterized by the two paradigms *behavioral science* and *design science*. They further argued that both paradigms are fundamental to research in information systems and explain the differences between the paradigms in the following way:

> "*The behavioral-science paradigm seeks to develop and verify theories that explain or predict human or organizational behavior. The design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts*" (Hevner et al, 2004, pp. 75)

The design science paradigm is considered by Hevner et al. (2004) to be a problem solving paradigm, where the central part is the creation of an artifact. DSR within information system research concerns the creation of an IT artifact, broadly defined as "*constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practice), and instantiations (implemented and prototyped systems)*" by Hevner et al. (2004, pp. 77). In this definition four different types of artifacts exist *constructs*, *models*, *methods* and *instantiations*. These artifacts were also identified in March and Smith (1995), along with two processes of DSR, to *build* and *evaluate* the artifacts. The process of building the artifacts has to be executed on existing knowledge and theories. Moreover, the evaluation of the IT artifact is carried out on the basis on it's actual usefulness within one or more organization and therefore DSR can be considered to be applied. Furthermore, evaluation of the IT artifact can include studies of a behavioral nature, on how the artifact is perceived and used and how it has impacted an organization. Such studies can in turn verify or falsify existing knowledge and theories.

Hevner et al. (2007) present a set of guidelines for how to conduct and evaluate Design Science Research. These guidelines are listed and summarized below:

1. *Design as an artifact* – One result of conducting DSR is the artifact, which is constructed to solve a relevant problem within some domain. The artifact may be a construct, model, method, or

an instantiation and has to demonstrate feasibility. However, the artifact does not necessarily have to be an actual instantiation.

2. *Problem relevance* – The purpose of creating the artifact is to solve a relevant problem within some domain. DSR is therefore often carried out in collaboration with practitioners within this domain.
3. *Design evaluation* – The created artifact has to be evaluated with respect to utility, quality and efficacy, but also with respect to completeness, simplicity, elegance, understandability, and ease of use (March and Smith, 1995). Depending on the nature of the problem, the methods of evaluation can vary between observational, analytical, experimental, testing, or descriptive. Since design is also an iterative process, the evaluations will provide feedback to the design process.
4. *Research contribution* – DSR has to clearly contribute knowledge to the domain of the designed artifact, including design construction knowledge and/or design evaluation knowledge. In DSR the contribution will often be the artifact itself, which will show a new innovative solution for a relevant problem, but the contribution can also be new methodologies for evaluation.
5. *Research rigor* – In what way is the research process conducted? Rigorous methods have to be applied both during the construction of the artifact as well as during the evaluation of it. This concerns questions such as: Have relevant existing theories and best practices been considered and used? And, if not, has the theory or practice in question been disregarded for a good reason?
6. *Design as a search process* – DSR can be seen as the search for an artifact that will solve a particular problem in a new and innovative way. In it's nature the process is both heuristic and iterative. The boundaries of possible solutions will be discovered along with the process itself and they will be refined with each new iteration based on the available means, for example other preexisting technology.
7. *Communication of research* – The results of DSR have to be communicated both to practitioners and their managers. The managers need to understand to what extent resources are needed to utilize the new artifact. The practitioners will need an understanding on how the artifact works and also enough information to create an implementation.

For DSR to count as research and not just as consultancy work guidelines 4, 5 and 7 are important. Iivari (2007) also argues that researchers should not just try to solve any problem that appears in practice, since what first can be seen as a problem can be in fact be a conglomerate of problems. Instead the problem should be carefully selected as one of the sub-problems from that conglomerate. Iivari (2007, pp 52) further argues that DSR is about potentiality: "*A new idea or artifact may provide totally new opportunities to improve practice long before practitioners recognize any problem*". Thus, the problem that the artifact will in fact solve is one that has not yet appeared in practice, but is predicted by the researcher.

As described earlier in this chapter both CR and DSR involve the construction or building of an artifact, that is aimed at solving a relevant problem. This is probably the most obvious similarity between these two approaches. Piiranen and Gonzalez (2013) have made a thorough comparison between the two and they conclude that DSR and CR are compatible, with regards to their main features. It is also argued that CR to a certain extent can be positioned as a subset of DSR, but not the other way around. The greatest difference found in the comparison is that DSR does not require an instantiation, which CR does. In DSR contributions on other levels are recognized as valuable, where as in CR the level of practical relevance is stressed and part of the evaluation also requires an instantiation.

One example of CR or DSR within metadata research can be found in a position paper by Waugh (1998) that describes an early view on the metadata interoperability problem and proposes a solutions that is technically justified. Since Waugh (1998) was early in addressing this problem the scope of applicability was very complicated to examine. An approach similar to CR and DSR can be found in Greenberg, Crystal, Robertson and Leadem (2003) where the development of a metadata editor (application) is described including one iteration to improve the proposed solution. The problem at hand had direct relevance to practice and the applicability of the solution therefore became possible to examine. The two papers do however not refer to CR or DSR. In the PhD thesis by Nilsson (2010) CR has been applied to the problem of metadata interoperability. The relevance of the problem is throughly explained and the artifact created is a model of how to represent metadata in order to achieve machine-processable semantic interoperability. The work is grounded in a set of requirements for such interoperability, that were found during many years of practice. The model created was also a contribution to the theory of how metadata interoperability can be achieved.

In chapter 5 it will be described how both DSR and CR has been utilized in order to answer research question 2. This will be carried out by showing, for each guideline of DSR, how it has been followed together with how each phase of the CR has been approached.

3. Metadata

This chapter will describe perspectives on metadata that underpins the work presented here. Some of the numerous aspects of metadata interoperability are included and these aspects are related to the scope within which the metadata is accessible. This chapter also includes explanations of some terms used in chapter 1 and 2

A coherent set of metadata that exists about a thing is commonly referred to as a *metadata record* for that thing[4]. Each metadata record is expressed through one or more metadata *elements* that can take on one or more values[4]. The value of an element can be another resource or a textual value or a *reference* to either of these items. Examples of elements used in a metadata record for a book can be *Title* and *Author,* where the value of element *Title* can be the text-string "High Fidelity" and the value of the element *Author* can be a reference to the metadata record that describes the author Nick Hornby. What kind of information that is expressed using the elements *Title* and *Author* is probably obvious to people skilled in English. But a proper label for the element should also be accompanied with a description of how the element is intended to be used. Metadata elements are defined and described in m*etadata standards,* which can also define restrictions on the allowed set of values for an element.

Nowadays metadata is commonly stored and managed on computerized systems. Computers are good at detecting that the same element has been used to describe two or more things. That enables quick searches and sorting of result lists. A result list from a search in a library can for example be easily sorted by title or by the last name of the author as the sorting principle. In contrast, humans are better at interpreting the meaning of the metadata elements and their values.

3.1 Metadata as information

According to the definition given in the introductory chapter, metadata is a kind of data. However, the metadata record about a thing is meant to provide information and metadata can therefore can be considered to be information. This thesis will not discuss an overarching theory of information. Rather, information is here considered "tautologically", i.e., to be something that informs. Furthermore, individual pieces of data might not always have to make sense on their own, but combined together they should. Consider for example, a newspaper publishing data

4 http://dublincore.org/documents/usageguide/glossary.shtml

about the score of a football-game. For that information to make sense it has to include at least 1) The names of the two participating football-teams, 2) the score, and probably also 3) when the game took place. Thus, the three individual pieces of data together provide valuable information to someone interested in knowing the score of a particular football game, while the individual pieces provide little information considered in isolation. However, when using only these three pieces of data for the football-game, it is assumed that the receiver of the information knows that the name of the two teams correspond to two football teams. Thus, in order to interpret data some assumptions may have to be made about the receiver of the information. Moreover, in order to provide information, the provider has to consider which assumptions that can be made about the potential receiver.

Extrinsic characteristics (Utility)
Completeness – Do the set of elements used to describe a thing provide a good enough description? How many of the metadata elements in a metadata record have values assigned to them?
Conformance to acceptance – Does the metadata for a collection meet the expectations of the community of users of that collection?
Logical consistency and coherence – Has the metadata for a collection been created in a consistent way? For example, have names of persons and places been consistently expressed?
Accessibility (intellectual) – Is metadata expressed in a way that is not unnecessary complicated?
Intrinsic characteristics (Quality)
Accessibility (physical) – Are there any physical barriers for accessing the metadata? Have proper formats for the metadata been used?
Accuracy – Are the values provided factual and not obviously false? Is the metadata free from typographical errors?
Timeliness –Does the metadata describe the current state of the resource, i.e. is it up to date?
Provenance – Did the metadata creator possess good knowledge of the topic at hand? What guidelines have been followed? Was the metadata created in an automatic manner or was is created manually?

Table 1: Aspects of metadata quality (and utility) adapted from Bruce and Hillman (2004).

For metadata this means that it is the combination of metadata elements used in a metadata record that together will form the information that is provided. A metadata element and its value separated from the metadata record will most often not provide a good enough description for the intended receiver of the information, unless there are several underlying assumptions that hold true. However, not every possible aspect is feasible to consider when metadata is created, so the chosen set of elements used are the ones that are considered to be both feasible and to provide a useful description. Thus, metadata will be created with certain receivers in mind, receivers for whom certain assumptions can be made. How useful of a description the metadata will provide will therefore depend on the receiver and thus it will be context dependent (Sutton, 2008). Bruce and Hillman (2004) describe seven aspects of metadata quality. These aspects were further considered by Sutton (2008) who argued that they included both quality and utility aspects and categorized each of the aspects in this way. The different aspects of metadata quality are described in Table 1,where they have been divided into the two categories quality and utility. A more thorough description of these aspects can be found in Paper 5.

3.2 Creating metadata - creating information

As described earlier, creating metadata about a thing is creating information about it. When a person creates metadata, several other roles are often involved as well. The list below serves as an overview of the most important roles that have been used in this thesis.

1. **Metadata standard creator:** this role creates and defines metadata elements for a standard by describing the meaning of the elements, how they should be interpreted and what their intended use should be. The standard creator can also restrict the value-space for an element in a valid metadata record and also define a number of values and their descriptions. A standard is normally not created by a single person, but rather by a working group that has certain aspects and a certain community of users in mind. For example, the IEEE LOM[5] standard was developed to describe learning aspects of things and the standard *MPEG-7*[6] was developed to describe audio/video resources. A related task that will fall under this role is the maintenance of the standard, for example driving discussions about the changes in the definition of the elements in order for the standard to remain up to date.

5 See IEEE Computer Society (2002) in the reference list
6 See ISO/IEC 15938-2 (2002) in the reference list

2. **Application profile developer:** the task of this role is to combine a set of metadata elements from one or more standards that is suitable for a certain community, application or any other context or situation where metadata is used. The outcome is an application profile that defines the set of valid metadata records according to the profile. The application profile can introduce stricter usage of the elements than the standard they originate from dictates, but it cannot change them in any other way. The term “application profile” will be further described in chapter 4.2.2.
3. **Metadata record creator:** this role involves creation of metadata records about a resource. A metadata record creator will also edit metadata records they need to be updated. Since metadata can be used to describe several different aspects of the resource, it could require several different authors to be responsible for creating and maintaining different parts of the metadata.
4. **Programmer:** the metadata record creator needs ways to interact with metadata that is stored digitally and it is the role of the programmer to provide this. The programmer role can also include the responsibility to comply with the specified metadata standards that are used. Moreover, since different authors edit different parts of the metadata, it would fall into the tasks of this role to create different editors of metadata for different metadata record creators. This role also involves maintenance of the program and applications that are used to interact with the metadata.

3.2.1 Approaches for metadata editing

Consider an example where a metadata record creator wants to edit the values of the elements *title*, *subject*, and *author* of a book. For that purpose a metadata template has been provided that could look something like what is shown in figure 3 below. This kind of template looks like a form for entering data and therefore this kind of metadata template is in this thesis referred to as a form-based metadata editor. The form-based approach to editing metadata is described in paper 3. The labels (or fields) of the metadata element are shown to the user and a way to create or manipulate the corresponding value is provided. In figure 3 this is done either through a text-field or a choice mechanism like a drop-down menu.

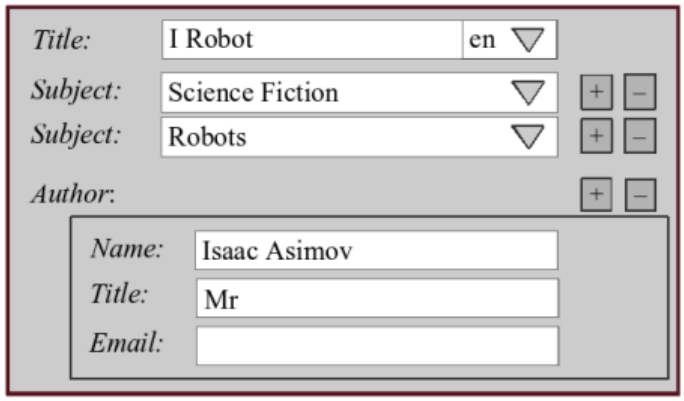

Figure 3: An example of a user interface for editing metadata

As mentioned above, figure 3 shows that the title can be changed through a text-field and that the language of the text provided can be set using a drop-down menu. Only one value for the element labeled *Title* is possible to provide, whereas it is possible to give more than one value for the element labeled *Subject*. In the example, two values have been entered and a third can be added by pressing one of the plus-buttons to the right of the chosen value. Doing that creates a new entry below the one where the plus-sign was pressed. The value can only be chosen from a restricted set of alternatives, and this choice is performed via a drop-down menu. It is also possible to add more than one author for the book, and for each author, the information on the name, title and email can be provided.

In general a form-based interface can be constructed in three different ways according to how adaptable they should be to changes in what metadata that can be edited:

- **Fixed metadata editors** have little or no adaptability as they are hard-coded to support a specific set of metadata elements. Such editors are useful when the metadata that is used never or rarely changes.
- **Configurable metadata editors** can be adapted to changing needs regarding what should be possible to express in the metadata. These changes are achieved by changing a configuration. Such configurations are typically developed by experts on compliance with specific metadata standards and/or domain-specific extensions.

- **Generic metadata editors** can edit any metadata by more or less exposing the underlying syntax of the metadata records to the end-user.

Such a fixed metadata editor is hard-coded, a change in the editor would require programming efforts in order to adapt the form-based metadata editor. To use a generic editor that more or less exposes the underlying data syntax for the metadata record would pose the requirement that the metadata author should know to properly how to express the syntax properly and in a correct manner. This might be possible for very specific users, such as the editor of MARC21[7] metadata in libraries, as described in paper 5. But, for the average user, a graphical user interface that provides more guidance was considered to provide the best alternative[8], and configurable metadata editors was the approach chosen to answer my second research question. A configuration mechanism for metadata editors was one of the artifacts that was constructed, by the author and this is described in detail in section 4.3.

3.3 Combining metadata using different standards

As mentioned in section 1.4, the combination in one metadata record of metadata elements from two or more standards is a situation that often appears in practice. We will now go more into depth regarding the kind of problems that are involved in this kind of combination and relate them to relevant efforts to achieve metadata interoperability.

Waugh (1998) argues that existing metadata records might have to be extended with new metadata as new standards emerge since this provides new ways to describe a particular thing (see aspect timeliness in table 1 on page 26). In the same manner Nilsson, Palmér and Naeve (2002) argue that metadata can be used to express information that is not objective, but has to be interpreted in a more or less subjective context, and thus the information expressed will be of a subjective nature. From there it is argued that all possible metadata records for a thing cannot be covered in one standard. In practice this means:

- It should be possible to edit the metadata records about a resource during the lifetime of the resource,
- it should be possible to effortlessly combine elements from existing metadata standards with elements from new metadata standards as they emerge.

7 http://www.loc.gov/marc/bibliographic/
8 Notably in the projects LUISA and Organic Edunet mentioned in section 1.3

As mentioned above, the IEEE LOM is a metadata standard developed for describing so called *Learning Objects*. It was the first metadata standard developed for an educational context that was acknowledged by a large standardization organization (IEEE). The term Learning Object indicates that the resource being described using IEEE LOM is meant to be used in a learning context. More specifically, the IEEE LOM standard defines a learning object as: "*any entity - digital or non-digital - that may be used for learning, education or training*". A general reflection to consider is what falls outside this definition? I.e., what is not a Learning Object? With such a wide definition it becomes apparent that any entity that could be used in an educational context could be described with educational metadata, perhaps using IEEE LOM. But, such an entity (thing) may already have other kinds of metadata, which then would have to be combined with elements from the IEEE LOM standard to describe the intended learning aspects. Hence, one standard is not enough to fully describe all the relevant aspects of the thing. This point was elaborated in paper 1, where IEEE LOM is proposed to be used in combination with other metadata standards. However, to combine the IEEE LOM standard with other standards is possible, but has been shown to be problematic when it comes to the interpretation of the resulting metadata record (Nilsson, 2010, pp. 35-37).

Nilsson, Johnston, Naeve and Powell (2007) give an example with a metadata record that combines elements from the IEEE LOM standard with metadata elements from the Dublin Core Metadata Element Set (2010), abbreviated DCMES. This could be achieved by combining these standards on a syntactical level since both of them can be expressed in XML. Hence, the XML document holding the combined metadata can be contained within the same description and this would provide a combination on the syntactic level. Two cases can be considered here:

1. An IEEE LOM records exists and metadata using the DCMES standard should be added.
2. A metadata record using DCMES standard exists, and should be extended with some metadata elements from IEEE LOM

With a metadata description using both IEEE LOM and DCMES both of the options above would be relevant. It would then be expected that a computer application processing and interpreting any of those two descriptions could detect them as being identical. However, this is not possible without having explicit information about the two standards encoded into the application. So even if the syntax used is the same, the two possible combinations into one metadata description would not always be interpreted as identical. Even if rules can be encoded in the application, a new set of rules need to be encoded when a third standard

is to be used for the metadata description. Nilsson (2010, pp. 35-36) also gives a similar example where IEEE LOM and the MODS standard are to be combined. Here the standards can also be combined on the syntactical level, though the computer processing and interpretation of their meaning is not always consistent with the meaning expressed using the separate standards. And unless certain special rules are applied, only metadata elements from one standard can be interpreted by the machine. To conclude, these two examples show that the combinations are possible on a syntactic level, thus the two descriptions are syntactically interoperable. However, problems occurs when the resulting metadata record is to be interpreted by a machine. Thus, the two exemplified combinations are *not* semantically interoperable on a machine processing level. One approach to deal with this type of interoperability when creating a new standard is to deal with it once the problem occurs (i.e. post-hoc coordination). Another option is to make the standard as widely interoperable as possible (i.e. pre-hoc coordination). Nilsson and Naeve (2010) present two approaches to handle the interoperability aspects:

- **vertical harmonization** – defined as "*interoperability on different levels within a given set of standards*".
- **horizontal harmonization** – defined as "*interoperability based on interoperability across standards*".

Vertical harmonization is a limitation on the scope that is considered for interoperability, while the horizontal harmonization would provide interoperability across any standards and thus enable combinations of element from different metadata standards to be performed rather effortlessly. Nilsson (2010) argues that the way to achieve horizontal harmonization involves the use of a core model of how metadata is represented and he concludes that the only existing framework that can serve as a candidate for a core model is the one given by RDF[9] (Resource Description Framework).

Attempts have been made to express the IEEE LOM standard in RDF (Nilsson, Palmér, Brase, 2003). This has turned out to be problematic and the work was continued by the DCMI-IEEE LTSC Taskforce[10], but this effort now seems to have been abandoned.

Configurable metadata editors would make it possible to change the form-based metadata editor without having to change the code of the underlying application. For such a configuration to be as generic as possible, it should aim for interoperability across any standards, i.e.

9 RDF provides one way to express metadata and is described further in section 3.4.2.
10 http://dublincore.org/moinmoin-wiki-archive/educationwiki/pages/DCMIIEEELTSCTaskforce.html

horizontal harmonization. As stated above, RDF is currently the best (i.e. the most interoperable) choice of representation and it would allow an effortless combination of elements from several metadata standards. RDF prescribes *how* to express the metadata, but *not what* should be expressed (i.e. what elements that should be used). The configuration mechanism was therefore developed primarily for RDF and metadata that can expressed using that standard.

3.4 Exposing metadata in new environments

The Internet is based on a stack the protocols that include TCP[11] and IP[12], on top of which the part of the Internet called the WWW is constructed using another protocol called HTTP. Through the use of these techniques, several possibilities have opened up that we nowadays perhaps take for granted. For example, it is possible to easily upload and share and also view pictures on different an image-services, such as Instagram, Flickr or Picasa. Moreover, the WWW is commonly used for transferring documents in different media-formats that are meant to be consumed by humans, such as HTML and PDF . For example, a library that is exposing their library catalogue for searching would typically do that through a search interface and a result list generated with HTML which include the metadata intertwined with information about how to present it. To also publish the data separately (i.e. without information on how to present it) seems to become more and more common[13] [14]. Data that is published is not forced to follow a certain format, and can be published in various "non-committing" formats like CSV[15], XML[16] and JSON[17].

To expose metadata, or data in general, on the web for everyone to use can potentially have many advantages. As it is exposed to the eye of the public, someone will hopefully use it. Moreover, it can also be used in situations that are new, which in turn can reveal errors. Of course, when the metadata is used in new situations its quality cannot be guaranteed, but, on the positive side what information is missing will become apparent. As long as that information is fed back from the users of the metadata this can help to improve it. Hence, through the usage, the quality and utility of the metadata will be revealed. Users can also start to combine one metadataset with another metadataset, which can lead to

11 https://www.ietf.org/rfc/rfc793.txt last visited 21st Sept 2014
12 http://tools.ietf.org/html/rfc791 last visited 21st Sept 2014
13 See https://www.data.gov/ for governmental data,last visited 10th Sept 2014
14 See http://data.nobelprize.org/ for data about nobel prize laureates, last visited 21st Sept 2014
15 http://tools.ietf.org/html/rfc4180 last visited 21st Sept 2014
16 http://www.w3.org/XML/Core/ last visited 21st Sept 2014
17 http://json.org/ last visited 21st Sept 2014

discoveries of connections between metadata records that no-one knew existed before since the records were separated from each other.

The founder of the web, Tim Berners-Lee[18] , wrote about a vision of a Semantic Web in an article in Scientific American (Berners-Lee, Hendler, and Lassila, 2001). This web was presented as an extension to the existing WWW with data that was machine processable both syntactically and semantically, and where the machines could interpret and to some extent also draw conclusions from the data. One of the first step taken to realize the vision was to create RDF as a standard to represent data on the web, a standard that makes it possible to connect data across the web. This is described in section 3.4.2. Efforts and discussions about realizing the vision of the Semantic Web have lead to the coining of the term "Linked Data" (also discussed in chapter 3.4.1). The Linked Data "movement" makes use of the capability of RDF to connect parts of metadata in one record to metadata in another record.

One aspect of exposing metadata is that it can end up in contexts where the assumptions behind the creation of the metadata are no longer valid. Exposing the existing metadata on the WWW can be a good start by showing what kind of information that is available, and to consider what "hidden" assumptions that are involved. In this thesis this issue has been identified, but it was considered too large to include. However, it provides rich opportunities for future research about metadata.

3.4.1 Semantic Web and Linked Data

The activity to realize Tim Berners-Lee's vision of a Semantic Web is now coordinated by the World Wide Web Consortium (W3C), which also describes the Semantic Web as "*a web of data*"[19]. Having a Web of Data can be compared to the present WWW that forms an interconnected web of documents related to each other. The meaning of the links between the documents cannot be explicitly expressed, and instead it has to be interpreted by the human who takes part of the documents. For the Web of Data the links between the data express some kind of meaning, i.e. how the linked pieces of data are semantically related to each other. In this way different meanings of data can be expressed in a machine processable form. The Resource Description Framework (RDF, see chapter 3.4.2) was developed for this purpose in order to enable the creation of a Web of Data.

18 http://www.w3.org/People/Berners-Lee/ last visited 6th of August 2014
19 http://www.w3.org/RDF/FAQ last visited 21st Sept 2014

Web Ontology Language[20] (OWL) is another standard developed for realizing the possibilities of the Semantic Web. By using OWL it is possible to express rich and complex relations between things. This can be formalized knowledge in the form of e.g. a taxonomy of animals, where it, for example, can be expressed that giraffes and beavers are mammals, and all mammals are animals. These kinds of formal knowledge expressions are aimed at enabling machines to reason and infer conclusions. Such knowledge representations are also referred to as ontologies.

Linked Data is a term that has emerged from the discussions held and initiatives taken as efforts towards realizing a Semantic Web. Bizer, Heath, and Berners-Lee (2009, pp 1) describe the term Linked Data as "*a set of best practices for publishing and connecting structured data on the Web*". Linked Data is thus the part of the Semantic Web that aims at connecting data across the web, i.e. creating the Web of Data. Linked Data enables one or more metadata elements in a metadata record to have values that point to a metadata description of a thing that resides somewhere else on the web. An example could be a Learning Resource that is a composition (aggregate) of other Learning Resources from two or more databases.

The linking of linked (or, rather, "linkable") data is achieved through using RDF, which in turn make use of the Uniform Resource Identifier[21] (URI), a web standard used to uniquely identify some thing in the web environment. A normal web-address, like http://example.com/ is an example of a URI, and metadata expressed in RDF can use URIs to uniquely identify the thing being described. It should be noted that all URIs are not resolvable, i.e. when a request is sent to the URI you are not guaranteed to get a reply. For Linked Data, Berners-Lee (2006) argues that the URIs used should be resolvable. It is also recommended to use URIs that are based on the HTTP[22] schema.

Naeve (2005) described three different semantic levels that can be put into relation to how metadata is exposed with respect to its possibilities to be linked to:

Semantic isolation - metadata is exposed internally to an organization. The metadata can also be exposed to the outside world, but without possibilities to link to other resources outside of the dataset, and without possibilities to link into the metadata records from descriptions in external datasets. The metadata records thus lives in isolation from other metadata records in other datasets.

20 See http://www.w3.org/2001/sw/wiki/OWL (visited 21st Sept 2014) and chapter 4.2.1
21 http://www.w3.org/TR/webarch/#identification last visited 21st Sept 2014
22 https://tools.ietf.org/html/rfc2616 last visited 21st Sept 2014

Semantic coexistence - metadata records from different datasets are exposed so that linking between them is possible. However, the possibility to link between datasets has not yet been utilized. Thus, the metadata records can be considered to be “linkable”.

Semantic collaboration - The metadata records are linked together in various ways. This is the situation in Linked Data when the value in one metadata record is the description of another thing residing in another metadata record. The metadata element through which the link is expressed includes a meaning, i.e. it can be interpreted semantically.

At what level a specific set of metadata is situated depends on the scope. For example, within a company, the level of semantic collaboration might have been reached, though compared to the outside world the company's metadata could be considered to exist on the level of semantic isolation. The Web of Data (i.e. Semantic Web and Linked Data) corresponds to the level of semantic coexistence and collaboration with a scope that is global. For the Web of Data it is not predetermined what data that will be connected, rather this is supposed to happen opportunistically. If that is supposed to happen in an effortless manner the interoperability has to be pre-coordinated, which corresponds to the horizontal harmonization described in section 3.3. Since RDF is the description standard used to achieve the Web of Data and also the best model for achieving horizontal harmonization it is this way to express metadata that has been in focus in this thesis.

3.4.2 RDF – Resource Description Framework

RDF is a recommendation from the W3C on how to express data on the web and it was developed as a first step towards a Semantic Web. Through its data model it prescribes how information is expressed as well as the syntaxes that can be used in doing so (see RDF, 2014).

Looking at RDF from a high level perspective it, can be seen as a directed graph with nodes and arcs between some of the nodes. When using RDF the simplest non-empty graph is called a *triple* and consists of two nodes and one (directed) edge between them. A triple is an atomic part in the RDF data-model, which mean that a node cannot exist on its own. Hence, any valid expression in RDF is a set of triples. The directed edge that goes from one node to the other is called a *predicate* or a *property*. The node from which the edge starts is called the *subject* of the triple and the node that the edge points to is called the *object* of the triple. A set of triples is called an *RDF graph* and this graph can be connected since the subject of one triple can also be the object of another triple.

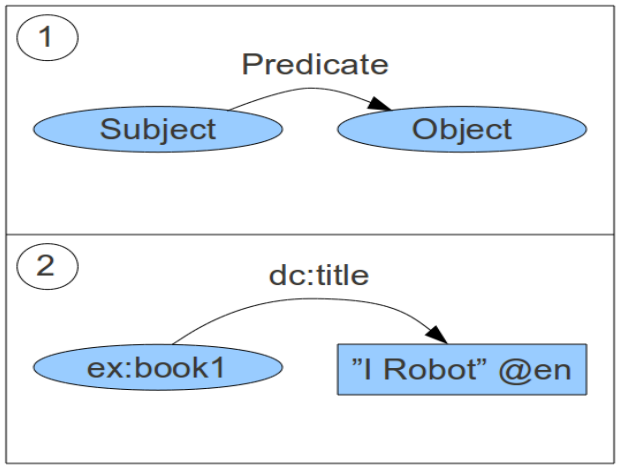

Figure 4: Part 1 shows a triple with a subject and an object as nodes, and the predicate as an edge that connects the subject to the object. Part 2 shows the triple that expresses the title in the example editor in figure 3. Here dc:title is an abbreviation of the IRI http://purl.org/dc/terms/title that is used to identify the property, ex:book1 is short for the example-IRI http://example.com/book1.

The basic construct of RDF is illustrated in figure 4. This example shows how the element *title* of the book in the example editor in figure 3 is represented in RDF.

RDF expressions

In an actual RDF-expression, the different parts of a triple are required to be expressed in a certain way. These requirements come from the need to identify certain parts of a triple in order to see if they are also part of any other triples, and to check if the triples form one or more connected graphs. For example, when a subject in triple A is also the subject of triple B it is necessary to somehow be able to recognize that these two subjects are the same. Here two cases can be considered: 1) The subject of a triple is only to be used within the current RDF-graph or 2) the subject should be possible to be (re)used or should be able to be connected to another RDF-graph, beside the current one. In the first case, the subject only needs to be uniquely identifiable within the current graph, and for that purpose RDF has a construct called *blank nodes*. For the syntaxes that can express RDF it is a requirement that blank nodes are supported, but how this is done is left to the ones who define the syntax. For the second case, a standard URI (Uniform Resource Identifier) is used. This

construct is defined and described by IETF (Internet Engineering Task Force) as "*...a compact sequence of characters that identifies an abstract or physical resource*" (Berners-Lee, Fielding, Masinter, 2005). A simple example of a valid URI is *http://www.example.com/*, so a normal web-address is a valid URI. Berners-Lee et al. (2005) describe the rules that express which sequences of characters that form valid URIs. Hence, when triple A and triple B are not stored in the same database it can easily be detected that the subject of these triples are the same, since they use the same URI as identifier.

The updated RDF specification uses Internationalized Resource Identifiers[23] (IRI) which is a generalization of URIs that allows for universal character encoding (Unicode). A valid URI can only be expressed with the US-ASCII character set (Berners-Lee, Fielding, Masinter, 2005). All URIs are IRIs, but all IRIs are not URIs. In order to be in line with the latest RDF specification IRI will be used instead of URI in the rest of this thesis.

To create a valid RDF expression, one must follow the rules given below (Cyganiak, Wood and Lanthaler, 2014):

- The only way to express a **predicate** is to use a IRI.
- The **object** of a triple can be a blank node, a IRI or a string of characters, which is called a *literal*. The objects of the two triples in part 1 of figure 6 are example of literals.
- The **subject** of a triple can either be a IRI or a blank node.

Since IRIs can become very long, they are usually abbreviated when expressed, as for example in figure 4 where the IRI *http://purl.org/dc/terms/title* is presented as *dc:title*. Here the part *dc:* is the prefix that corresponds to a so called *namespace* IRI, which in this case is: *http://purl.org/dc/terms/*. The IRI that identifies a resource is also called the *name* of the resource.

When RDF expresses metadata, blank nodes are commonly used for grouping of values such as the author information in figure 3. How this description is expressed is shown in figure 5, where the property *dc:creator* points to a blank node, which in turn points to the values for the properties *foaf:name* and *foaf:title*. The prefix *foaf:* corresponds to the namespace IRI *http://xmlns.com/foaf/0.1/*. The FOAF (Friend Of A Friend)[24] vocabulary specification provides a set of properties that can be used for metadata about a person. Throughout this thesis the prefix *ex:* is used for the purpose of giving examples, as in figure 5. If needed, it can be

23 http://tools.ietf.org/html/rfc3987 last visited 21st Sept 2014
24 http://xmlns.com/foaf/spec/ last visited 21st Sept 2014

expanded to *http://example.com/*, which is a IRI reserved for the purpose of giving examples.

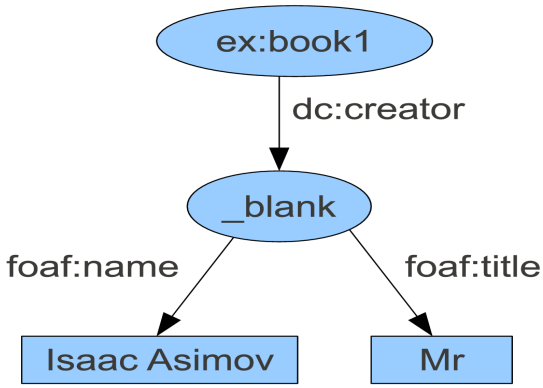

Figure 5: Example usage of a blank node.

In figure 3 the underlying RDF expression presented and edited after the label “Subject” in the metadata editor can take two forms. The common characteristics is that the values come from a restricted set that the user is allowed to choose from, typically via a drop-down menu. What is hidden from the user is how this is actually expressed in RDF. The value can either be a IRI or a predefined set of strings, and the two possible resulting RDF expressions are shown in figure 6. Part 1 shows the situation when the selected values are from a set of predefined string values, and part 2 when the possible values are from a set of IRIs.

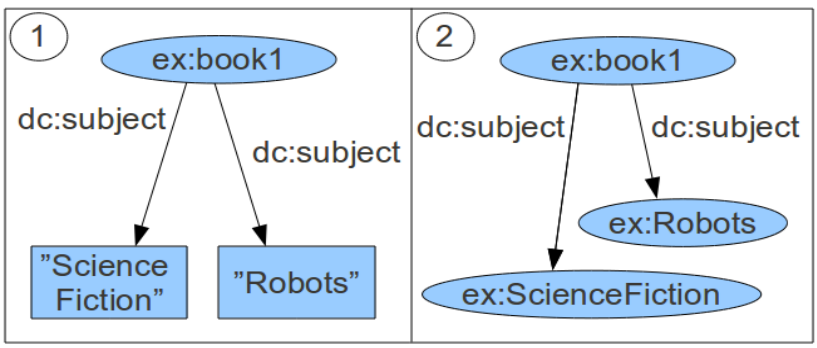

Figure 6: The two possible ways to express the triples when "Subject" in the editor of figure 3 is edited, as a string in part 1 or as a URI in part 2.

Note that in order to express more than one “Subject”, the property *dc:subject* is used multiple times, creating more triples with the same subject.

4. Annotation Profiles – a framework for adapting form-based metadata editors

In the previous chapter metadata was described with some of the possible approaches for how to enable metadata creation via a graphical user interface. That chapter also included important aspects of metadata interoperability and described the relation between metadata and the Web of Data. RDF was also described, in an overview manner, as an enabler for the Web of Data. This chapter describes the artifacts created to answer research question 2. The artifacts enables form-based metadata editors to be created for metadata expressed in RDF and the metadata editing forms can also be changed without having to update the source code of the surrounding application. The artifacts have been developed by my colleagues and me which resulted in a framework we have termed *Annotation Profiles*.

The Annotation Profiles framework and its different parts are described in this chapter. The most important item is the *Annotation Profile Model* (APM), a model for form-based metadata editors. A valid instance of that model is called an *annotation profile* and is a representation of a metadata editing form, which holds all the information needed for a machine to create that form. The main reason for developing the APM was to create a simple way to update a metadata editor in an application without having to change the source-code. Changes to the metadata editing form can instead be achieved by changing the annotation profile that the metadata editing form is created from. Thus, the APM enables a configuration mechanism for metadata editing forms. The APM is described in paper 2 and this chapter will provide an overview.

The metadata editing form naturally dictates what metadata the user can create and the development of the APM did not intend to include a new type of interface to how metadata is created, but rather to improve the flexibility when the elements possible to edit in the form have to be changed. As described in paper 6 the APM has the potential to empower the metadata record creator to change the form-based metadata editor in that manner.

This chapter is structured in the following way: Section 4.1 describes the requirements for a configuration mechanism for metadata editors, which constitutes the requirements for the design of the APM as well as the information an annotation profile requires. Parts of the required information might already exist and be expressed through various sources. These kind of sources are described in section 4.2 with respect to how they can provide information to an annotation profile. Chapter 4.3

gives a detailed description of the APM, which is followed by chapter 4.4 that describes how RDF can be edited over the web with the help of an annotation profile. Chapter 4.5 describes the relation of the APM to the two ideas of 1) automatic and semi-automatic creation of the user interface, and, 2) End-user development. Finally, section 4.6 presents some of the use-cases for the APM and also the evaluation that has been carried out with people using the artifacts of the framework.

4.1 Requirements on a configuration for form-based metadata editors

A basic requirement on a configuration mechanism is that it should contain enough information for a metadata editing form to be generated from it. This includes 1) what metadata that should be possible to edit and 2) in what way it should be possible to edit it. Are values edited as free-text or are they chosen from a restricted set of values? The identified requirements are presented in paper 5 and there divided into the following categories:

- **Completeness** – includes support for editing arbitrary well-formed RDF triples. Any kind of valid RDF expression should be editable. The configuration mechanism should specify which statements that are editable and thus indirectly which ones that are not.
- **Structure** – includes cardinality constraints on elements as well as their ordering in the resulting form. A direct correspondence between the graph structure and its presentation in the form should not be enforced. For example, it should be possible to hide a complicated graph-structure with intermediate resources from the end-user. And, it should be possible to introduce "cosmetic groupings" of statements, even though this construct does not exist in the underlying RDF graph-structure.
- **Interaction** – includes hints on how the end user is supposed to choose values from a restricted set, e.g. check-boxes, radio-buttons, dropdown menus, or search-dialogs. It also includes mechanisms for string validation according to datatypes, control of auto-complete mechanisms etc.
- **Presentation** – includes multilingual labels and descriptions to aid the user in deciding how to edit. Font, color, indentations, borders, and everything else that has to do with appearance is also included here.

How these requirements are met in the APM is described in section 4.3. However, as part of the information requirements can be found in related initiatives and standards section 4.2 provides an overview of what information each of them can contribute to an annotation profile.

4.2 Related initiatives and standards

Different kinds of standards and initiatives exist that can hold information which can be directly reused in an annotation profile. What kind of information these standards and initiatives can provide will be described below. It will also be explained what is missing in order to fulfill the requirements expressed in section 4.1.

4.2.1 RDF Vocabularies and Web Ontology Language, OWL

RDF Schema (Brickley & Guha, 2014), commonly referred to as RDFS, is a specification used to express so called RDF vocabularies. An RDF vocabulary is also expressed in RDF and provides a set of classes and properties. *Classes* represent a formalized approach to give names to sets of resources that share common qualities. *Properties* complement classes by providing names to specific (and hopefully separate) aspects of these resources and how to encode them and their values as datastructures. This turns RDF Schema into a type system for RDF with a set of classes and properties, much like the type system in an object-oriented programming language. The commonly abbreviated namespace for RDFS is *rdfs*[25], and it will be used in the examples to follow.

The top-class in RDFS is a resource identified with *rdfs:Class* as its (abbreviated) IRI. The property *rdfs:subClassOf* is in a RDF triple to "*...to state that all the instances of one class are instances of another* " (Brickley & Guha, 2014). To indicate that a resource is an instance of a certain type (or class), the property *rdf:type* is used. In a similar fashion, for properties, a top-class exists for properties with the IRI *rdfs:Property* and the property *rdfs:subPropertyOf* is used "*...to state that all resources related by one property are also related by another*" (Brickley & Guha, 2014). An example of this kind of relationship is the property *dcterms:creator* which is declared as sub-property of *dcterms:contributor* (DCMI Usage Board, 2012). This means that if someone is the creator of some kind of resource, that same person is also considered to be a contributor to that resource. Moreover, RDF Schema has defined the two properties *rdfs:domain* and *rdfs:range* that are used for restricting the possible triples that can represent the value of a certain

25 The full namespace URI is http://www.w3.org/2000/01/rdf-schema#

property. These are properties used on instances of the class *rdfs:Property* the following way:

- The two triples *P1 rdf:range C1* and *P2 rdf:domain C2* states that:
    - *P1* and *P2* are instances of the class *rdf:Property* and,
    - *C1* and *C2* are a Classes,
    - triples using property *P1* will have only instances of class *C1* as object
    - triples using property *P2* will have only instances of class *C2* as subject.

The triple *dc:creator, rdfs:range, dc:AgentClass* restricts the object in the triples when *dc:creator* is the property. The value in those triples need to be instances of the class *dc:AgentClass*. This means that RDFS can be used to declare how a property can be used, i.e. what type of values that it can have, partly restrict the allowed triples.

Web Ontology Language, OWL (Hitzler, Krötzsch, Parsia, Patel-Schneider & Rudolph, 2009) is an extension of RDFS, used to define ontologies. An ontology can be seen as a kind of vocabulary that has more intricate semantics between the classes and properties. OWL provides a way to restrict the cardinality of a property, i.e. how many times the same property can be repeated for the same resource. This can be expressed by using *owl:onProperty* together with the properties *owl:maxCardinality* and *owl:minCardinality* that each should have a non-negative integer as value. The property *owl:cardinality* can also be used when the maximum and minimum cardinality are the same. Restricting a triple by stating that the values come from a certain class can be done by using the property *owl:onProperty* in combination with the properties *owl:someValuesFrom* and *owl:allValuesFrom*.

The constraints upon an RDF expression offered by RDFS and OWL are restrictions on properties. Hence, for a property, the allowed object and subject of the triple can be specified by declaring a certain class as domain or range. For example, looking at part 2 of figure 6, if the values for the property *dc:subject* are to be from a set of IRI:s, they can be selected in the application after checking if they fall under the restrictions of the range. Alternatively, the combination *owl:onProperty* and *owl:allValuesFrom* can also be used to provide the selection. The information expressed when using *rdfs:range* and *owl:onProperty,* as exemplified here, also tells us that the value cannot be a literal. By using the cardinality restrictions in OWL, the application knows how many

times a property can be repeated, and this can be reflected in the metadata editor by inactivating buttons for adding or removing values for this property. It is important to note that OWL and RDFS only restrict how a property can be used, not the set of properties that can be used.

4.2.2 Application Profiles

An Application Profile provides a way to specify which properties and values that are allowed for a valid metadata record. The term Application Profiles was defined in an article by Heery and Patel (2000) as ”*schemas which consist of data elements drawn from one or more namespaces, combined together by implementors and optimized for a particular local application*”. This means that different metadata elements from different standards are combined into one metadata record. In the case of RDF this means that an application profile would express the set of properties that are allowed for a certain metadata record, possibly together with a restrictions on these elements. This is not always equivalent to the restrictions expressed using RDFS or OWL , since an application profile can express even stricter restrictions. For example, if the restriction in RDFS states that *P rdfs:range C1*, the application profile can state *P rdfs:range C2* if and only if the class *C2* is a subclass of *C1*. This means that the expression in RDFS and in the application profile are kept consistent with each other.

An application profile is usually defined in a document that states what properties to use and how these properties are intended to be used in order to form valid metadata records and descriptions (see for example Powell, 2005). Possible uses of application profiles include comparing how the resources of two separate, but yet similar, collections are described, i.e what properties do they have in common and are the value-sets the same for these properties?

Dublin Core Application Profile and the Singapore Framework

The Singapore Framework (Nilsson, Baker & Johnston, 2008) was developed by the Dublin Core Metadata Initiative[26] (DCMI) as a standard for how a Dublin Core Application Profile (DCAP) (Coyle & Baker, 2009) is to be documented. According to the Singapore framework, such a profile consists of the following components:

- **Functional requirements** – a mandatory document that describes what the application profile is designed to support.

26 http://www.dublincore.org

- **Domain model** – a mandatory document that defines the basic entities of the application profile and their relationships. The main purpose of this document is to define what kind of entities that exist and how they are related.
- **Description Set Profile** – a (mandatory) machine-processable part of the profile that defines the set of valid metadata records. It provides a way to express constraints on a metadata record.
- **Usage guidelines** – an optional document that further describes how the profile is to be applied, i.e. how the properties are meant to be used.
- **Encoding guidelines** – an optional document that describes if the metadata is encoded with a specific syntax.

DCAP is based upon the Dublin Core Abstract Model (DCAM) (Powell, Nilsson, Naeve, Johnston & Baker, 2007). However, the data models found in RDF and DCAM are compatible, and therefore a DCAP can be utilized when the metadata at hand is expressed in RDF.

A *Description Set Profile* (DSP) defines the set of valid metadata records (Nilsson, Miles, Johnston & Enoksson, 2007) in a machine processable syntax, through the notion of templates. A DSP consists of one or more *Description Templates*, that each expresses constraints on the resource that is to be described. Furthermore, a Description Template consists of one or more *Statement Templates* that each expresses that a certain property is included together with the constraint on that property. Thus, statement templates restricts what properties that are allowed to be used in a triple. A statement template can be of two different types:

- **Literal statement template** – which is used when the value is restricted to be a literal. For example, the value of the property *dc:title* in figure 3 is a literal,
- **Non-literal statement template** – which is used when the value is not a literal. In RDF, an IRI or a blank node are examples of non-literals.

It is also possible to constrain the values in a statement template by expressing different constraints which are comparable with *rdfs:range* in RDFS.

An experimental wiki-syntax where DSPs can be expressed has been developed by the author (Enoksson, 2008). It is an extension of a syntax for the MoinMoin wiki engine[27]. The idea is to use the normal wiki-syntax

27 http://moinmo.in last visited 21st Sept 2014

that also contains a way to express the different parts of a DSP. Hence, the normal wiki-syntax can be mixed together with the DSP-syntax, and from that single source it is possible to generate both an HTML-document, that contains the functional requirements and the domain model, as well as the machine-processable DSP represented in XML. Hence, both the formal machine processable expression and the parts intended to be read by humans can be mixed into the same source documentation.

When creating a DSP, the restrictions on a property must be consistent with the related restrictions given by RDF Schemas or OWL ontologies. DSP provides a machine processable way to define the set of properties to be used for the metadata record, which could not be done by RDFS or OWL alone. DSP was designed to define a set of valid metadata records, but a metadata editor presented to a metadata record creator might have to exclude some of those properties. DSPs are therefore not entirely suitable to be used as a configuration mechanism for metadata editors. Furthermore, a DSP cannot express a predictable order of elements (in a resulting metadata editor) as well as no possibilities to express where the editing should take place when a deeper graph structure of triples are involved. Thus, complementary information is required. Golder et al. (2010) mentions such a complementary declaration of information when using DSPs to create metadata editors, but no further documentation about this has been found.

The Dublin Core Metadata Initiative (DCMI) has initiated the RDF Application Profile Task group[28] which started in the beginning of 2014. This group is reviewing the requirements of application profiles for metadata expressed in RDF. As stated above, the data models for RDF and DCAM are compatible, so the work done for DCAP can be reused within this task group. The task group is also gathering use-cases and discussing related initiatives, in order to avoid doing work that already has been done. Two of these initiatives are worth mentioning here, Resource Shapes[29] and Shape Expression[30]. The first one is a high level RDF vocabulary that specifies the shape of RDF resources. The shape of an RDF resource is defined as a "*description of the set of triples it is expected to contain and the integrity constraints those triples are required to satisfy*". Shape Expression is fairly similar and is intended to function as a constraint language for RDF, such as XML Schema[31] and

28 http://wiki.dublincore.org/index.php/RDF-Application-Profiles last visited 21st Sept 2014
29 http://www.w3.org/Submission/2014/SUBM-shapes-20140211/ last visited 21st Sept 2014
30 http://www.w3.org/2013/ShEx/Primer last visited 21st Sept 2014
31 From the W3C http://www.w3.org/standards/xml/schema (last visited 21st Sept 2014)

RelaxNG[32] that can define restrictions for XML. Both of these initiatives are under development and from the documentation that is available they will not contribute with any further information to an annotation profile than what a DCAP would do. These initiatives are therefore not compared or described further here.

4.2.3 Fresnel

Fresnel is a display vocabulary for RDF intended to be used for displaying RDF metadata in a user interface (Pietriga, Bizer, Karger, & Lee, 2006). There does not seem to be much activity around the initiative apart from the creation of another vocabulary called PRISSMA (Costabello, 2013). The PRISSMA vocabulary is developed to display RDF on mobile phones, and reuses several constructs from Fresnel. Displaying and editing metadata are in some ways similar and a Fresnel declaration contains information that can be used in an annotation profile.

A reasonable requirement for a configuration mechanism both for viewing and editing RDF metadata would be that only the parts that should be displayed or edited in a certain situation are shown in the resulting user interface. And, when using Fresnel, the set of properties to display can be declared to be a subset of the properties that are declared for example in a DSP. The information that Fresnel can contribute to the Annotation Profile Model is:

- the selection of which triples to display. This is done by declaring a set of properties in a so called *Lens*,
- how the set of selected properties and property values should be displayed. In Fresnel this is declared in a *Format*.

A DSP can be compared to a Lens in the sense that they both make a selection of triples. The Format in Fresnel will in turn express how the selected triples are to be displayed. A configuration mechanism for an editor would need to use something similar to a Lens in order to select which triples that should be possible to edit . A DSP might not be enough since it is a reasonable requirement that not all the properties of a DSP should be exposed in the resulting editor. However, all Lens constructs do not uniquely declare which properties to display. For example, a Lens could be declared to include all triples for a resource, or via another construct list several alternative properties to present before the most appropriate one is chosen. This flexibility is not suitable for an editor, since it needs to know explicitly which properties to use since it has to create the structure of the metadata. Since a Lens cannot always provide

32 http://relaxng.org/ last visited 21st Sept 2014

an explicit construct for how the structure of the metadata (i.e. triples in RDF) should be created, a Lens can only be reused in a configuration mechanism with certain restrictions. The Formats used in Fresnel can provide some information on how the metadata should be edited. However, since the purpose is to express how metadata should be presented some default assumptions would have to be made when Fresnel is used for editing purposes.

4.2.4 A full representation of the form-based metadata editor

The existing standards and initiatives presented in sections 4.2.1-4.2.3 are not aimed specifically at metadata editing, but with a combination of them it is possible to come close to the requirements expressed in section 4.1. However, in order to fulfill all those requirements, additional information has to be declared.

To recapitulate what has been concluded above, by using a DSP it is possible to define the set of valid metadata records, and by using Fresnel it is also possible to select a subset of elements of a valid record and declare how it should be displayed. To use these techniques in order to create a form-based metadata editor would in some cases be possible. Fresnel also gives the possibility to select what part of the RDF graph that should be displayed, and that information can be reused in a configuration for metadata editors. However, some assumptions have to be made about how the metadata is to be handled in the resulting metadata editor, and this will not always correspond to the requirements that a user would have. For example, is the value being edited through a free-text or from a restricted set of values[33]. And, as previously stated, Fresnel does not in all cases explicitly declare what graph-structure to create.

Hence, in order to construct a configuration mechanism for metadata editors, information both from the DSP and from Fresnel can represent information that is possible to re-use. However, relying on pieces of information represented in various places and standards would probably result in a model that becomes complicated to manage. Instead, the approach chosen was to enable relevant information to be extracted into a representation of the full model of the form-based metadata editor.

33 i.e. should a text-field, or some kind of choice mechanism be displayed, like drop-down menus, radio buttons etc.?

4.3 Annotation Profile Model – information model for RDF metadata editors

The APM is an information model that, when instantiated, can be processed by a machine in a way that enables the machine to create a form-based metadata editor. The information model was designed to be independent on any particular syntax, but to be possible to be expressed in several syntaxes such as JSON[34] or XML. An annotation profile can be created with information from different sources, like RDF Schema, DSP etc described in section 4.2. However, when such resources are not available an annotation profile needs to stand on its own, so the syntaxes used for expressing an annotation profile also need to be able to express those parts that can be found using other sources.

The APM was developed by me and my research group[35] in order to contribute to a flexible metadata authoring environment for RDF. It was also the main artifact developed to answer research question 2. The model is described from a technical perspective in Palmér, Enoksson & Naeve (2007) and also in a research paper (see paper 2). Here the model will only be described in terms of its most important concepts. The APM can be divided into two submodels: 1) The *Graph Pattern Model* works on a data-level to express what metadata that should be editable and 2) The *Form Template Model* that expresses and restricts *how* the metadata can be edited. An instance of the Graph Pattern Model is called a *graph pattern* and an instance of the Form Template Model is called a *form template*. In the examples given in this chapter, the graph pattern and the form template are expressed using two separate syntaxes. However, me and my colleagues have developed a way to express both of these models together in JSON, though this syntax is not further included in this thesis.

4.3.1 Graph Pattern Model

The Graph Pattern Model (GPM) is used for capturing existing triples in the RDF graph that is to be edited, and, when needed, also to create new triples. Hence the GPM is responsible for the completeness requirement given in the list of section 4.1. This includes defining what metadata elements that can be edited and then, indirectly, what metadata that should not be possible to edit in the resulting form-based metadata editor. This is done by restricting which triples and combinations of triples that will be affected. Hence, the role of the GPM is to capture the triples that already exist, as well as to act as a pattern to create new triples. Capturing existing triples that should be edited can be carried out

34 http://www.json.org last visited 21st Sept 2014
35 http://kmr.csc.kth.se/wp last visited 21st Sept 2014

by querying the RDF graph. Moreover, when new triples are created, a query can act as a template for the new triples or combination of triples that should be created. A simple example query can be found in figure 7 where the title for the book displayed in figure 3 should be edited, Something that involves only one triple. The IRI for the book is `ex:Book1` and the property is `dc:title` so the object of the triple or, equivalently, the value of the property, needs to be either captured in the existing graph, or created if that triple does not exist.

```
< ex:Book1 dc:title ?T >.
```

Figure 7: Simple graph pattern (in a simple made up query language) using the property `dc:title`.

Matching triples in an RDF graph can be performed by iterating over the triples of the graph, and, when found, a binding is made from the variable `?T` in the query to the matching object. Exactly how this is done will depend on how the system managing the RDF graph is implemented. If no matching triples exist in the graph, the query will act as a template on how to create new triples. So, when the title of the book is edited in the user interface, the value that is edited by the user will be bound to the variable `?T` and it will be possible to create the full triple. Thus, the same query that was used for capturing existing triples can also act as a template for creating new triples. The term graph pattern is borrowed from SPARQL (Prud'hommeaux & Seaborne, 2008), a query language for RDF. A graph pattern is made up of one or more triple patterns. The graph pattern given in figure 7 consists of one triple pattern. The variables in a graph pattern also play an important role as the bridge between the Graph Pattern Model and the Form Template Model which is described in section 4.3.2. To capture deeper structures, like intermediate blank nodes, variables can be inserted in both object and subject position in the graph pattern like variable `?B` in the graph pattern in figure 8 below.

```
< ex:Book1 dc:creator ?B >
   < ?B foaf:name ?N >
 < ?B foaf:title ?Title >
   < ?B foaf:email ?EM >
```

Figure 8: An example graph pattern with a deeper structure.

This graph pattern includes four triple patterns that are connected through the variable `?B`. Since `?B` is in subject position in three of them it can only be bound to either a blank node or an IRI. When variable `?B` cannot be bound to a value, a new value needs to be created. However, the given graph pattern does not specify if a blank node or a IRI should be created. This requires the graph pattern to not only deal with triple-patterns, but also to make it possible to specify requirements on the type of the node, which could be expressed something like this `< nodeType(?B, URI) >`. When capturing triples, such a restriction will only bind `?B` to a URI (or in line with the new RDF specification this should be an IRI). The corresponding situation appears if the node type is specified to be a blank node. However, the Graph Pattern Model does not require the node type to be specified, but instead defines a default choice for every situation. In the graph pattern above, where `?B` will be bound to a variable that is the subject of one triple and the object of another, the default choice is specified to be a blank node. The variable `?N` and the variable `?Title` will only be bound to a value in the graph if `?B` has already been bound to a value. The default node type for `?N` and `?Title` is *literal*, since, according to this graph pattern, they appear only in object position.

A graph pattern like `<ex:Book1 dc:subject ?S >` would match a literal, but can be changed in two ways: 1) by expressing the node type for `?S`, as mentioned above or 2) by adding a so called *constraint triple pattern*, i.e. a triple pattern with the only variable in subject-position. Going for option 1, the possible matches would be every possible IRI, whereas option 2 would restrict the value to be either a blank node or a IRI that matches the constraint. An example of such a graph pattern is displayed in figure 9.

```
< ex:Book1 dc:subject ?S >
< ?S rdf:type ex:ExampleClass >.
```

Figure 9: Example of a graph pattern with a constraint triple pattern.

Note that the same kind of information that the constraint triple pattern holds can be specified in an RDF vocabulary by using the property *rdfs:range*. For example, in an RDF vocabulary provided to the annotation profile the triple *dc:subject rdfs:range ex:ExampleClass* would provide the same information as the constraint triple pattern in figure 9.

As seen in this example, the graph pattern is expressed as a query that also restricts which triples and combinations of triples that can be modified. In the examples given above, the IRI *ex:Book1* is the resource being edited. In order to simply reuse graph patterns they are made to be more generic, and a variable is placed there instead of an explicit IRI. When the graph pattern is applied, that variable is then replaced by the IRI. The following requirements for the graph pattern were detected during the development phase:

- One variable will act as the starting point.
- The graph pattern needs to be fully connected through the variables. This means that the subject position of a triple pattern can only be represented by a variable.
- Closed loops are not allowed in the graph pattern, and it will therefore correspond to a tree-structure.

As previously stated, the Graph Pattern Model was inspired by the SPARQL query language which is a recommendation from the W3C. However, SPARQL, or any other RDF query language are not fully suited as a dedicated syntax, since they are too complex and include capabilities that are so complex that it cannot be concluded what triples to create. The RDF query languages also allow closed loops in their graph patterns.

4.3.2 Form Template Model

The *Form Template Model* (FTM) supports the interaction, presentation and structure requirements given in section 4.1. Included in the model are ways to connect to the metadata that is to be edited through the graph pattern. An instance of a FTM is called a *form template* and contains information about the resulting form. A form template can be broken down into a list of *form items* and according to the FTM there exists three different types:

- **Text Form Item (TFI)** – used for:
  - plain text, possibly providing a language or datatype.
  - IRI as a plain text-field.
- **Choice Form Item (CFI)** – used for:
  - choosing a value from a predefined set of IRI:s
  - choosing a value from a predefined set of a literals.

- **Group Form Item (GFI)** – used for
  - intermediate resources (such as blank nodes in RDF) and their further properties.
  - groups of other form items.

  Group form items accomplish this by having other form items as children.

In the example metadata editor displayed in figure 3, the title of the book can be edited in the first text field. In front of the text-field, the label '*Title*' has been added, in order to provide the user with hints about what value that is expected to be provided in this text field. Also, there is no plus sign next to the text-field which tell us that only one value is allowed. These are some of the attributes that can be specified for a TFI. The label '*Subject*' and the dropdown menu next to it is an example of what can be generated from a CFI. An example of a GFI is provided after the label '*Author*', which in this case is "held together" by a blank node in the RDF structure. The form template is not locked to a certain syntax. For example, RDF has been used in one implementation whereas the latest implementation is using the JSON syntax. XML provides another alternative syntax and the example below can be used to generate the example editor in figure 3. The XML-attribute *vref* is used for references to variables in a graph pattern.

```
<FormTemplate xmlns="http://kmr.nada.kth.se/APtags#">
   <Group max="1" vref="X">
      <Description lang="en">The resource being edited</Description>
      <Text max="1" vref="T">
         <Label lang="en">Title</Label>
         <Description lang="en">A title of the resource</Description>
      </Text>
      <Choice vref="S">
         <Label lang="en">Subject</Label>
         <Description lang="en">A subject classification</Description>
      </Choice>
      <Group vref="B">
         <Label lang="en">Author</Label>
         <Description lang="en">An author expressed in foaf</Description>
         <Text max="1" vref="N">
            <Label lang="en">Name</Label>
            <Description lang="en">The full name of a person</Description>
         </Text>
         <Text max="1" vref="Title">
            <Label lang="en">Title</Label>
            <Description lang="en">The title of a person.</Description>
```

```
        </Text>
        <Text max="1" vref="Email">
           <Label lang="en">Email</Label>
           <Description lang="en">An email to the person</Description>
        </Text>
      </Group>
   </Group>
</FormTemplate>
```

In this XML-expression, the overarching element *FormTemplate* contains one and only one element *Group,* that is a GFI which contains all the other form items directly or indirectly (a GFI can be contained in another GFI). The element *Group* here has the attribute *vref* with the value *B,* which is a reference to the variable `?B` in the graph pattern. Furthermore, the element *Group* contains another element *Label* that holds the label to be shown to the user, which in this case is '*Author*' with a value that is given in English. The label for each form item can also be given in several languages by repeating the element with the translated value and the attribute *lang* set to the correct language code. The value of the element *Description* is used in the resulting editor to further guide the user regarding what value to fill in. Typically it can pop up when hovering over the label with the cursor. The element *Description* can also be repeated in different languages in the same manner as for the element *Label*. Any element *Text* corresponds to a TFI that also contains the elements *Description* and *Label* that are used in the same way as for the element *Group*. The same applies to the element *Choice* that corresponds to a CFI. Both the elements *Choice* and *Text* are required to reference a variable in the graph pattern through the attribute *vref*.

Some of the *Text* elements have an attribute called *max*, which expresses the maximum cardinality. For the *Text* element having the value *T* for attribute *vref*, the attribute *max* is set to 1 which means that at most one value is allowed for that property. This is reflected in the example UI shown in figure 3 where no new fields for the title of the book can be added. These kinds of cardinality constrains can also be found in OWL descriptions, as described above in section 4.2.1.

Form Item attributes

The examples above only show the value for some of the possible attributes[36] for the different form items. The attributes that apply to all the types of Form Items are listed below[37] :

36 The term “attribute” here refers to the generic term and not to XML-attributes
37 The list is duplicated from Palmér et al., (2007)

- **Label** – A short text string that will appear first in the form item presentation to guide the user regarding what value to provide. The text string can be provided in several different translation.
- **Description** – A descriptive text that typically appears as a tool-tip[38] or is accessible via a help button. It may be translated into several languages.
- **Variable-reference** – A reference to a variable in the graph pattern.
- **Minimum cardinality** – The minimal number of occurrences of the corresponding form item in the form model relative to the parent form item.
- **Maximum cardinality** – The maximal amount of occurrences of this form item in the form model relative to the parent form item.
- **Preferred cardinality** – The preferred amount of "ready-to-fill-in" form items. Hence, when initializing the form model for some RDF metadata, this is the desired number of form items. If existing data yields a lower number of form items, additional (empty) form items are generated.
- **Cardinality** – Setting this attribute is equivalent to setting the min, max, and preferred cardinality to the same value.
- **Class** – This attribute assigns a class name (or set of class names) to an element, similar to HTML. The intention is to provide hooks for styling, for example by using Cascading Style Sheets[39].

Some attributes that are specific to the choice form item and text form item are listed below:

- **Value** – A predefined value to be presented in this form item. If a variable is referenced, this value is added to the variable binding set.
- **Enabled** – *True* if you are allowed to use this form item for editing, *false* otherwise. If this attribute is not specified, the default value is set to *true*.

38 A hint or help that appears in a GUI, typically used when the user hovers with the pointer or cursor over an item in the GUI.

39 http://www.w3.org/Style/CSS/ last visited 21st Sept 2014

Yet another attribute, called *Choices*, can be applied to the Choice Form Item. The value for this attribute is a list of possible choices that are presented in the resulting form-based metadata editor . This construct is used in the situation where the set of values are given explicitly instead of being extracted from a vocabulary. Each value can be either a literal or a IRI. Labels for each value in different languages can be provided and. A description that describes the value further can also be added, which enables means to provide guidance for the metadata record creator.

Additional attributes

In the examples given so far, the mechanism provided for the end user to make a choice is through a drop-down menu. The underlying reason is because the basic FTM only provides a simple listing of the possible choices. An alternative way to present the choices is to sort them into a some kind of hierarchy, but this is not supported in the basic FTM. However, it is allowed to add other attributes than those described in the basic FTM. If this is done, it is required that implementations can ignore these attributes and still have access to enough information to be able to create a form. No other restrictions are imposed, but with this possibility, additional local features for specific implementations can be provided.

4.3.3 From annotation profile to Form Model

As previously described, an annotation profile can be separated into a graph pattern and a form template. The information in the annotation profile can be derived from different sources, such as RDF Schemas, OWL ontologies etc. When implementing support for the Annotation Profile Model in code the input is one annotation profile, the RDF graph to edit and the IRI[40] of the resource whose metadata should be edited. The following approach has been taken:

1. Apply the graph pattern to the RDF graph that contains the metadata that should be edited. Create variable bindings from the variables given in the graph pattern to the corresponding parts of the RDF graph.
2. Apply the form template to the set of variable bindings. Since the variables are also referenced in the form template, a connection to the part of the RDF graph to modify is given through the variable bindings.
3. The result of step 2 is called a Form Model (Palmér et al., 2007) and from that model, the information needed to create an editor is provided.

40 Internationalized Resource Identifiers, as described in chapter 3.4.2

In the implementation that has been developed, these steps have been implemented and turned into a code library. The kind of "organizational refactoring" involved in such an undertaking leads to modularization of code and enhances the possibilities for simple reuse in different applications. Further aspects of reusing code in relation with the annotation profiles framework are described in section 4.5.

4.4 Remote editing of RDF

RDF was developed for representing data on the web as well as to realize the Web of Data. The Linked Data movement prescribes how the Web of Data should be realized through IRIs that are resolvable. In the LUISA project, a requirement to edit the RDF metadata on a remote basis occurred when we soon found out that the metadata editor and the metadata storage could not run on the same machine. Thus, a solution for editing metadata across the web had to be developed. This solution is described in paper 3 and is also presented in an overview manner in section 4.4.1 below. The solution represents another artifact constructed in order to answer research question 2. However, another solution has also been implemented, and it is also described in section 4.4.2. The scope of each of these solutions is limited in the following way:

- The metadata is always represented in RDF.
- The updates are performed by a user through a form-based metadata editor in some kind of client application.
- The client application and the web-server holding the RDF metadata are communicating in a stateless manner, such as in the protocol HTTP. Hence, the communication can be done over the web.

Apart from giving a description of the two solutions, this section will also discuss the fitness of each solution from the perspective of certain situations.

One solution would be to transfer the whole RDF graph back and forth between the editing client and the remote repository. However, this is not feasible for several reasons. One of the problems considered in the LUISA project was consuming too much bandwidth. For example, the RDF storage used in the Organic Edunet project, storing over 800000 triples[41] (around 80 Megabytes), would perhaps not present a problem to transfer with the high speed Internet connections that exists today, but this

41 The metadata about these resources can be found at http://oe.confolio.org (last visited 21st Sept 2014)

solution would consume unnecessary bandwidth. It is possible to reduce the consumed size, as will be shown in the following sections. The remote RDF storage would also have to be locked for other updates during the metadata editing in one client.

Two solutions were developed with the two following approaches to how the number of triples to transfer could be reduced:

1. Use the graph pattern of an annotation profile, since it provides the information about which triples the form-based metadata editor will affect. This approach basically tries to only deal with RDF triples that will be affected by the form-based metadata editor. These triples are extracted and transferred to the application and then updated through the form-based metadata editor. The resulting triples are then sent to update the remote repository.
2. Use a construction called *named graphs* for RDF and only deal with the subgraphs of one named graph at a time.

The solution that used approach 1 is described in paper 3, where it is shown how the triples that need to be retrieved can be found by making use of the graph pattern of an annotation profile. The solution developed with approach 2 requires that the RDF graph is divided into subgraphs in advance. These subgraphs are called *named graphs* (Carroll, Bizer, Hayes, Stickler, 2005) since each triple in the subgraph is connected to one IRI that identifies the subgraph. This approach works well under certain conditions that are identified and discussed below.

4.4.1 Reusing the graph pattern of an annotation profile

As previously stated, a graph pattern defines which triples that can be affected in a metadata editing form. Together with the IRI that identifies the resource to be edited, a query can be constructed that will extract the triples that are affected by the form. The general principle behind how to detect which triples to include on the remote storage is to traverse the given graph pattern and find all the variables that occur in subject position of a triple pattern. Assume this is true for a variable `?X`. If `?X` also matches a node in the RDF-graph, the triples to include are the ones that have this node as subject and share the properties defined for `?X` in the graph pattern.

To extract the triples in this way is enough to enable the metadata editing on the client side, since all the triples that can be affected are included. However, it is not enough when the graph is to be updated on a remote repository, because of blank nodes that can exist in the RDF graph.

Unfortunately, blank nodes are not required to be identified in the same way across applications or systems dealing with RDF. Thus, how a blank node is identified might have changed by the time the RDF graph is received by the remote repository. Before this is explained further, the whole process of editing an RDF metadata record in this way needs to be explained. The steps are presented below, where it is assumed that we have access to the IRI identifying the resource which metadata that is to be edited as well as an annotation profile for this editing process:

1. The client sends a request containing a query to the remote repository. The query is generated from the graph pattern of the annotation profile and includes the IRI of the resource to edit.
2. The requested triples are extracted at the remote storage.
3. The extracted triples are returned to the client. Once received the annotation profile is applied, and a form is created and presented to the user.
4. When the user has finished editing, the updated graph is sent back to the remote repository, along with the specification of how the triples were extracted during step 1.
5. Step 2 is performed again to calculate the triples to remove, followed by a removal of those triples from the RDF-graph.
6. The updated triples are merged into the RDF graph.

In certain situations, the way to extract the triples described above can corrupt the graph after step 5 and 6 have been performed. This is due to the fact that when the updated graph is returned to the server there is no way of knowing if the same identifier for blank nodes have been kept. How things can go wrong is seen from this example: Consider three triples with the same blank node in the subject position. In step 2 only two of those triples are extracted, since it is only these two triples that can be affected in step 3. Assume that the two triples with the blank node as subject are not changed in step 3. However, the identifier for the blank node might have changed. This mean that after step 6 has taken place, the triple that was not extracted in step 2 might have another blank node in subject position than the two triples that were extracted. Thus, the RDF graph has changed in a way that was not intended and it has therefore been corrupted.

The proposed solution for this problem is to extract *all* the triples that have a matched node as the subject, instead of just the triples matched by the graph pattern itself. These extra triples will be left untouched by the

form, and included in the updated graph that is sent back. Furthermore, these extra triples are removed in step 5 and then inserted back again in step 6. Therefore, the correct triples are kept intact and the graph will not become corrupted. Extracting all the necessary triples, as described above, can be done through a SPARQL Describe Query (abbreviated SDQ) (Harris and Seaborne, 2013). Moreover, since repositories that work with RDF often have support for SPARQL, no new code needs to be written for extracting the triples. It should however be noted that it is not strictly defined what an SDQ should return, and different implementations can therefore return different sets of triples (see section 16.4 in Harris and Seaborne (2013)). In order to extract the set of triples as described above, the Concise Bounded Description[42] should be returned for every explicit IRI, as well as for all variables in the query that match a node in the graph.

The SDQ is then created from the graph pattern by simply removing all variables that only appear in object position. This is because the node in the graph that these variables will match will be included anyhow, and we do not need triples where that node is the subject. An example of an SDQ and its corresponding graph pattern is given in figure 10.

< ex:Book1 dc:creator ?B >
< ?B foaf:name ?N >
< ?B foaf:title ?Title >
< ?B foaf:email ?EM >
	DESCRIBE
	<http://www.example.com/Book1> ?B
	WHERE
	{ OPTIONAL {
	<http://www.example.com/Book1>
	<http://http://purl.org/dc/terms/creator> ?B
	. }
	}

Figure 10: To the left a graph pattern and to the right its corresponding SPARQL Describe query.

One case that is not covered by the suggested solution is the case when the same blank node is the object of more than one triple. In this case all those triples are not necessarily extracted in step 2 and thus not removed in step 5, and this can corrupt the graph. However, what seems to be the

42 http://www.w3.org/Submission/CBD/

common practice for blank nodes is to only have them appear in the object position of a triple, making the case just discussed appear seldom or never.

The main benefit of using the approach outlined here for editing a graph remotely is that it will work with any RDF graph, with the exception mentioned above. Moreover, the triples that should be extracted, transferred and removed are indirectly specified in the graph pattern. On the downside, a specific interface has to be implemented on the server in order to support the removal and update procedures. An update problem can occur if two users (A and B) are editing the same resource and if they both are in step 3 at the same time, using separate editors. If user A finishes first with steps 4-6, the metadata will be saved in a correct manner. However, when user B finishes, the graph extracted in step 5 will not correspond to the one extracted in step 2, since this subgraph has been changed by user A.

The problem occurs since information about when a set of triples was last updated is not kept, and the server will therefore only save the update that it has received last. This is a limitation of the solution, which could be overcome by including the original graph extracted in step 2 in the request performed in step 4. In step 5, the graph extracted from the describe query can then be compared to the graph extracted in step 2, i.e. it can be determined whether or not they are to be considered to be equal. In order to determine the equality of two RDF graphs one also needs to consider whether or not they are isomorphic. If any two graphs (not just RDF graphs) are isomorphic, the number of nodes must be the same in both graphs as well as the number of edges[43]. Algorithms that check for graph isomorphism usually make (more or less) qualified guesses on a mapping of nodes, and then check if the edges are the same or not, i.e. if the graphs are isomorphic or not. If there are *n* nodes involved, the number of possible guesses is *n* faculty[44]. In the worst case all these guesses have to be made, but if a mapping finds that the graphs are isomorphic, no further possibilities proposed have to be tested. The number of guesses can be reduced further by looking at certain characteristics of the nodes. For example, for each node check how many other nodes it is adjacent with. So, if a certain node has three adjacent nodes, this node will only be suggested in a mapping to a node in another graph that has the *same* number of adjacent nodes. In the specific case of considering two RDF graphs to be equal, the mapping also needs to take the IRI of that node might be identified with into consideration. A mapping for a node identified with a IRI has to be identified with the same IRI in both RDF graphs. In case such a mapping can be performed

43 Correspond to properties in RDF
44 Expressed in mathematical notation as *n!*

for all nodes identified with a IRI, the algorithm will continue to find a mapping for the blank nodes by guessing. After that, the edges/properties of the RDF graph are checked. An algorithm that finds out if two RDF graphs are considered to be the same has to make blind guesses on the mapping for the blank nodes. The blank nodes are included within a triple where the edge/property is identified with a IRI. Hence, the blank node might also be adjacent with a node identified with a IRI, and in this way the number of guesses can be reduced.

In summary, the time it takes to decide if two RDF graphs are considered to be identical depends on how many blank nodes that are are involved. But, the number of guesses can also be reduced, depending on in which triples the blank nodes occur. Thus, a conclusion to draw from this is that comparing the two RDF graphs in step 5 takes extra time, but how much depends on the RDF graphs at hand. Each practical setting will also have to evaluate the seriousness of the problem of overriding an update of the RDF graph. This seriousness will also have to be weighed against the frequency with which this situation will occur. It should also be noted that graph patterns cannot contain loops, which increases the likelihood that the extracted RDF graphs also do not contain loops. Moreover, for graphs that are tree-structures, the complexity of checking for graph isomorphism is *O(n)* (Hopcroft and Tarjan, 1973 ; Hopcroft and Wong, 1974) . This means that the time it takes to check for graph isomorphism grows linearly with the number of nodes involved.

4.4.2 Using Named Graphs

An alternative way to extract the triples that are to be transferred from the remote repository is to divide the full RDF graph into smaller subgraphs in advance. These subgraphs should correspond to the triples that can be affected i.e. to the subgraphs transferred and replaced in step 2 and 5 described in the previous section. A construct for RDF called *named graphs* can be used to create such subgraphs on the server side. As the name of the construct suggests, a specific name is given to the set of triples found in the RDF graph that forms the named subgraph. In this solution the identifier of one named graph must be an IRI, and in order to know which named graph it belongs to each triple in the named graph will be associated with this IRI. Hence, all the triples of a named graph can be extracted by sorting out which triples that belong to a certain named graph.

When editing an RDF graph remotely using Named Graphs, the same steps as described in the previous section are applied. However, in step 1 only the IRI of the named graph needs to be included in the request. For step 2, the named graph is extracted, even though it might contain triples

that are not necessary to include. In step 5 the specification of which triples to remove is done by using the IRI of the named graph. In step 6 the updated named graph is inserted, and it is assumed that all new triples belong to the named graph.

The main benefit of this solution is that the subgraph to extract is defined in advance. This also means that the update can be performed by replacing the named graph with the updated one, and no query-expression has to be used in order to determine which subgraph to replace. Another benefit is that the repository can hold information about when each named graph was last edited. This means that the scenario with user A and B in chapter 4.4.1 does not have to end in that the changes of the user who saves last are the ones that will be preserved. This problem can be solved in the following way: in the response sent to the client in step 3, a timestamp is included with information about the time when the graph was fetched, and that same timestamp is sent together with the updated graph in step 4. The repository can then compare the timestamp with the time of the last update of the named graph. If it has not been changed since it was sent to the client, the update is allowed. However, if changes have been made of actions have to be taken to not overwrite these changes. For example, the repository can reply to the user with a warning or error message that other changes have been made and they might be overridden.

The solution that uses named graphs was designed for the case where one named graph covers the description of one resource, i.e. one named graph correspond to one metadata record. As described above it is assumed that blank nodes cannot be shared between named graphs, since the identifiers of a blank node might change when one named graphs is transferred, changed by the client or returned to the repository. And, in one of the first publications that mentions named graphs it is stated that "*Graphs may share URIrefs but not blank nodes*" (Carroll et al., 2005, pp 248). Named graphs were first included in the latest set of RDF specifications, where the notion of RDF Datasets (see Cygniak et al., 2014) is also introduced. An RDF Dataset is a set of RDF graphs with at least one *default graph* and zero or more named graphs. According to Cygniak et al., (2014) blank nodes can be shared across the graphs in an RDF dataset. Thus, the solution presented here is limited with regards to how blank nodes have to be handled, since it was created before the introduction of this construct.

4.5 Approaches to creation of user interfaces

Myers and Rosson (1992) and Daniel et al. (2007) claim that much of the efforts developing applications is spent on the user interface (UI). A poorly designed UI will probably make the user unsatisfied with the application at hand, which is a good reason to spend resources on this part of the application. It is of course of interest to be able to reduce the resources spent when the budget for developing the application is a limiting factor. This is one of the reasons to pursue research into how UIs can be generated, or easily adapted to new contexts. How the APM relates to this kind of research is discusses in general terms. The discussion includes automatic adaption of UIs from one platform to another as well as the possibility to empower the end-users to adapt the UI in different ways. This is referred to as End-user development (Ko et al., 2011; Nardi, 1993).

4.5.1 Cameleon framework, different levels of the user interface

The Cameleon Reference Framework is a generic classification system for the different levels of computer systems that includes UIs, which can be found in (Calvary, et al., 2003). This framework includes the following four levels for the User Interface (UI)[45]:

- **Task and Concepts** – A description of what task that should be performed in order to reach the goal of the user. This includes a description of the domain model, for example the underlying data that the will be manipulated through the UI.
- **Abstract User Interface (AUI)** – An abstract representation of the UI in terms of interaction spaces that are independent of interaction modality (haptic, graphical, vocal, etc.). The AUI is made up of Abstract Interface Objects (AIO).
- **Concrete User Interface (CUI)** – A representation of a concrete UI that is independent of the platform. It is built from Concrete Interface Objects (CIO) that represent abstractions of actual UI components that the user can perceive, such as images, buttons etc.
- **Final User Interface (FUI)** – The actual user interface presented to the user. It consists of source code that realizes the CUI on a specific platform.

Research related to this framework includes how vertical transformation

45 See also figure 11

upwards and downwards to nearby layers can be achieved and formally expressed (see Meixner et al., 2011). A vertical transformation upwards in the hierarchy is referred to as an *abstraction* and going down in the hierarchy is called *reification*. For example, an abstraction is performed when a CUI representation is generated from the FUI, and a reification is performed when a transformation from the AUI level to the CUI level is carried out. Horizontal transformations between different representations on the same levels are also possible, and they involve transformations between different contexts of use. An example of a horizontal transformation on the CUI level is to transfer from a HTML representation to a WML representation (a kind of HTML developed for mobile phones[46]).

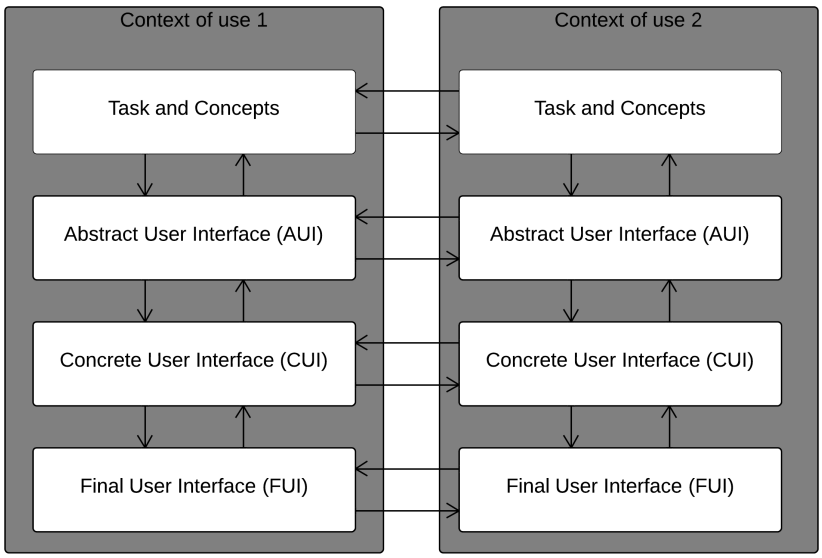

Figure 11: The levels of the Cameleon framework.

The framework can be used to describe scenarios such as the following: The UI of a desktop application has to be adapted to also work on a mobile phone. The FUI of the desktop application is then abstracted to a representation on the CUI level and then transformed into another CUI representation for the new platform. From this representation, a reification of the CUI is carried out in order to generate the FUI for the mobile phone.

46 http://www.wapforum.org/what/technical.htm last visited 21st Sept 2014

The goal of spending less effort on the UI will be achieved once transformations can be achieved with less effort than actually writing the source code for the UI of the new platform,. However, the Cameleon framework can also serve as a starting point for documenting the interface. Moreover, having documentation with a representation on each level and/or the transformations between them will also enable quick implementation of a UI for other platforms or modalities.

The Annotation Profile framework was developed independently of the Cameleon framework, since the latter was discovered later on in the research process. However, the concept of Annotation Profiles can be mapped into the Cameleon framework. For the highest level, *Task & Concepts,* the task considered for annotation profiles is of course the task of metadata creation. The graph pattern model of the annotation profile is part of the domain model considered on this level, since the graph pattern model is used to capture the metadata records. The APM was developed to be used on a computer with the standard interaction modality setup of screen, keyboard and mouse. Other interaction modalities were not considered, and no actual representation of the AUI level exist for the APM. The form template of an annotation profile is a representation of a form-based metadata editor on the *Concrete User Interface* level, since an annotation profile contains *Concrete Interface Objects*, expressed in a way that is independent of both platform and programming language. Furthermore, annotation profiles are not restricted to be expressed in any specific syntax, which enables horizontal transformation on the CUI level. Code implementing the concept will create the form-based metadata editor presented to the user, which is in turn mapped to the *Final User Interface* level.

The APM enables adaption of form-based metadata editors by changing the representation on the CUI level. Code that implements the APM can be modularized into a code library that can be reused by application developers. Two proof-of-concept implementations of such a code library have been implemented in the two programming languages Java (now abandoned) and Javascript, where the latter is called RDForms[47]. One reason to provide such a code library is to make it easier for a system developer that is developing an application that includes a metadata editor. As the system developer can reuse the code library, less time will have to be spent on developing code for how metadata in RDF should be managed. The author's licentiate thesis (Enoksson, 2011), describes a set of developers types involved in the development process around code libraries for the APM. In summary, these types are described in the following way:

47 https://bitbucket.org/metasolutions/rdforms/ last visited 10th Oct 2014

- **Annotation Profile Model code library developer** – implements a code library for the APM, like RDForms, in a specific programming language.
- **User Interface developer** – A system developer that writes the code for the final user interface of an application. The task of this role can be performed by reusing a code library.
- **User Interface Integrator** – Integrates the form-based metadata editor into the specific surrounding environment by tailoring its look and feel.

One of the goals when designing the APM was to increase the usage of RDF. When the development around the APM started, RDF was something that few programmers knew about. In order to increase the use of RDF it was also considered useful to provide a simple way for developers to create an editor for RDF metadata .

4.5.2 End-User Development

The idea behind end-user development (EUD) is to empower the user to easily adapt an application to her or his specific needs or preferences. For example, this can be concerned with what buttons that should be available in a UI, what colors these buttons should have. Lieberman et al. (2006, pp. 2) define EUD as "*a set of methods, techniques and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact*". A common example of EUD is the spreadsheet (Fischer et al., 2004; Mehandjiev et al., 2006), such as Microsoft Excel and Open Office Calc. In such programs, the user is allowed to express what calculation that should be performed. Although a professional developer might not call this activity system development, it shows the idea behind EUD, which is that all the requirements on a system might not always be possible to fulfill, and therefore the end users should be empowered to customize the application in order to fit it to their own needs.

Lieberman's definition of EUD is fairly wide and can cover many approaches on how the end-user development can be carried out. The spectrum ranges from text-manipulation, such as creating macros or scripts inside a program and editing a configuration file, to different types of graphical interaction such as drag-and-drop to design a customized UI.

Among the many ideas that came up while developing the different parts of the Annotation Profiles framework was to enable simple ways to adapt a metadata editing form. In order to change a form-based metadata

editor, a system developer often has to be consulted. For applications that have been developed utilizing the APM, it will be possible to adapt metadata editors, with respect to what metadata that is editable. Thus, changing an annotation profile will modify or extend the application, i.e. the use of the APM can be considered to enable EUD. The kind of roles that an end user developer can have is described in Enoksson (2011) as:

- **Annotation profile author** – create new profiles when needed. This can be the metadata record creator that adapts his or her own form-based metadata editor. It can also be a person whom the metadata record creator consults, person that is specialized on metadata and Annotation Profiles.
- **Annotation profile facilitator** – defines the requirements on the form-based metadata editor and indirectly on the annotation profile. This role could of course also be a metadata creator. A person filling this role would also be responsible for properly managing the annotation profiles for reuse and would also have to know where new ones that have been constructed by others can be found.

There are several potential ways to enable EUD for the roles above. However, how to do it will depend on what kind of freedom the end-users should be offered. It is already now possible to edit an annotation profile through a text-based interface. This gives a lot of freedom but requires knowledge about the syntax used to express an annotation profile. There are several errors that can occur in such an environment. For example if the structure of the graph pattern is incorrectly expressed incorrect metadata will be created. Another option to enable EUD would be a graphical editor where a metadata editing form can be assembled by combining different building blocks. In paper 6, a suggestion for such an application is outlined, where each building block corresponds to a form item in the APM. Prototypes of other applications where an annotation profile can be created have been developed and used within my research-group, but they have not been used and evaluated by others.

4.6 Use cases, development and evaluation

As previously stated, the initial ideas when developing the Annotation Profiles framework was to enable effortless changes to form-based metadata editors. The idea was then developed further and lead to the APM, which, when used in practice, can be seen as a configuration mechanism. From this work emerged the foreseen types of roles that were presented in section 4.5.

Much of the development of the concept and the first implementation of a code-library was carried out within the LUISA[48] project. The project as a whole was aimed at utilizing so called Semantic Web Services (see Studer et al., 2007) that would help users find suitable learning objects to achieve their learning goals. One academic and one industrial use-case were considered in the project. A tool for creating metadata about the learning objects for the use cases was needed in the project. The metadata elements and the set of values to be used were developed during the project. Furthermore, the metadata creation tool had to be designed to be able to cope with changes that occurred in what parts of a metadata record that should be editable.

The design of the APM was carried out in this project, as well as the first implementation of a code library in Java. That code library was developed to work in a setting where the application ran on the same machine as the RDF was stored. However, a tool that could handle editing an RDF repository not located on the same machine (i.e. only accessed remotely) was required, and this is the situation described in section 4.4. The approach developed within the LUISA project was described in section 4.4.1. This application was web-based but had to make use of a proxy-server that both fetched the RDF metadata from the remote storage and then generated the metadata editing form in HTML. When the user submitted the changes, a representation of the metadata editing form was sent to the proxy-server which then had to extract the RDF metadata and send it to the remote RDF repository. With this approach it became possible to edit RDF metadata remotely trough a web-browser. An alternative approach that was considered was to locate the functionality of the proxy-server in the browser. At that time we came to the conclusion that the web browser would use too much computing power to manage the RDF data in the client. This was the main reason that this approach was not considered further at that time.

The Annotation Profiles framework was further developed within the EU-funded projects Organic Edunet[49] and HematologyNet[50]. In the Organic Edunet project, a web-portfolio system was developed, aimed at managing learning resources for organic agriculture. The web-portfolio system was developed to be generic and consisted of two parts, 1) the server-application, called SCAM[51] (Ebner et al. 2009; Ebner and Palmér, 2014) and 2) the client-application called Confolio running in a web-

48 See http://cordis.europa.eu/ist/kct/luisa-synopsis.htm last visited 21st Sept 2014
49 See http://www.organic-edunet.eu/ last visited 21st Sept 2014
50 See http://www.hematologynet.eu/ last visited 21st Sept 2014
51 Nowadays also with the product name EntryStore (http://entrystore.com, last visited 21st Sept 2014)

browser[52]. It should be noted that SCAM was built as a generic web-application that would handle any kind of metadata descriptions in RDF, whereas Confolio is only one out of many possible implementations using SCAM as its repository of data. Moreover, since Confolio was built to be able to manage any kind of metadata in RDF, both SCAM and Confolio were highly useful within the project HematologyNet. In this project, the web-portfolio system managed metadata about learning resources in hematology. A central part of the project was the hematology curriculum (published by the European Hematology Association, latest version is described in Ossenkoppele et al., 2012). The curriculum was designed to be used for self-evaluation of skills and competences in hematology and was aimed at physicians in training to become specialists in hematology. The web-portfolio made use of the curriculum in two ways, 1) providing a set of values when classifying learning resources with respect to what part of the curriculum they were intended for, and 2) as a way to express both the current and the desired levels of skills and competences in hematology to a user of the web-portfolio system. The web-portfolio system then had information to conclude which learning resources that were suitable for a learner to engage with in order to reach the desired levels of skills and competences. More details of how the web-portfolio system was used in HematologyNet are provided in paper 4.

As a separate part of Confolio, a second implementation of a code library was developed in Javascript. It is nowadays named RDForms. The approach to remote editing of RDF metadata remotely is the one presented in chapter 4.4.2. Through the use of RDForms it became possible to configure the form-based metadata editors used in a Confolio installation.

As far as we know no other implementation of a code-library for APM exist. The developed code libraries have been licensed as Open Source under LGPL[53] and have been made publicly available. RDForms was made publicly available first on Google Code[54] and then on Bitbucket. Different versions of the code library are available for download, since the code has been improved and bugs have been corrected. At the time of writing version 2 has the largest amounts of downloads with a current count of 555 since it was made available in March 2013. There is also a Google group that has been created for discussing questions related to RDForms[55]. This group currently have 10 members, where 6 are outside of our group, the KMR group, which developed RDForms. As part of a

52 Nowadays also with the product name EntryScape (http://entryscape.com, last visited 21st Sept 2014)
53 http://www.gnu.org/licenses/lgpl.html last visited 21st Sept 2014
54 https://code.google.com/p/rforms/ last visited 21st Sept 2014
55 https://groups.google.com/forum/#!forum/rforms-discuss last visited 21st Sept 2014

continuous evaluation the author has asked all the people that have posted questions about how to use RDForms to the emailing-list if they are willing to be interviewed. So far, three persons have been interviewed. The interviews included questions on how they experienced working with RDForms, how they used it, if and how they find it useful etc. The results of these interviews have been reported in paper 6 and are briefly described here.

Case 1: This case included a university student in computer science that, at the time of the interview, was employed for a limited time by that same university. He was given the task to investigate if RDForms could be used to edit metadata in an application that manages metadata for the scientific publications at the university. The task was assigned to him by his boss, who had discovered RDForms, found it interesting, and wanted to see if it could be integrated into their application. The student felt that it was fairly easy to get something working. However, when the limited employment ended for the student, the task was also abandoned and RDForms never became a part of the system that managed the metadata in this case.

Case 2: In this case the interviewee was a system developer at a company that develops different kinds of tools to be used within the context of Semantic Web. The manager of the interviewee had found RDForms when searching for something that could be used to present metadata. The interviewee was therefore assigned the task to see if and how the metadata presentation could be achieved through RDForms. Thus, the code-library was here used only for presenting metadata. In this case RDForms was also considered to be easy to use in order to create something that worked. The interview also included a demonstration of the application into which RDForms had been incorporated. In this demo application, metadata records was presented with over a hundred elements. The application was developed for researchers within cultural heritage and humanities, and they makes note and suggest changes to the metadata records.

Case 3. Here the interviewee was a system developer that had been working in this profession for about ten years. RDForms was being used in an application where metadata about legal documents from the European Commission should be editable. The application included a complex system where metadata was imported from several systems into a central repository that represented all metadata in RDF. The need to edit that metadata lead to the task of developing a solution where RDForms was used. Before the decision to continue with RDForms was taken, two other options were considered. RDForms was chosen, since it was independent of the storage solution for the RDF metadata, which was

considered an advantage. The construction of the annotation profiles used was carried out by gathering information from multiple sources[56] on how a metadata record was allowed to be constructed.

At this stage no definite conclusions can be drawn from the limited number of interviews. However, they indicate that the APM and RDForms have some relevance "*in the wild*" and outside of the use-cases of the research group (the KMR group) to which the author belongs. There is also certain relevance in the fact that a code library can be created that is used by other programmers, who have been given the task to create an RDF metadata editor. Through the use of RDForms, they have in fact opened up the possibility for EUD. However, as stated in paper 6, that possibility will probably only become popular when a GUI is provided to create and edit and an annotation profile.

56 Such possible sources was described in section 4.2

5. Connecting the dots

The introductory section expressed the research objective and the two research questions of this thesis. Section 2 described the three different approaches that have been used in order to find answers to these research questions, namely *constructive research*, *design science research* and *activity theory*. This chapter will provide an answer to the research questions by 1) describing how activity theory was utilized in order to give an answer to the first research question, and 2) describe how the guidelines of design science research and the steps of constructive research have been followed in order to answer the second research question.

5.1 Using activity theory in order to understand the human activity of creating metadata

In order to answer the first research question the qualitative study presented in paper 5 was performed. The study was not aimed at metadata creation in one specific domain nor was it aimed at a specific type of metadata record creator. Instead, the study was carried out with the purpose of being exploratory across domains and to include persons that had some kind of practical experience with creating metadata. Activity theory, as an established theory, then provided a framework to find the main characteristics of the activities involved. The *extended activity system* (figure 1, page 16) was used for that purpose.

The study was performed by interviewing thirteen people who were, or previously had been, working with creating metadata. The interviews were semi-structured, i.e. questions were prepared in advance but additional question were asked, depending on the answers from the interviewees. The questions of the interviews were constructed to let the interviewee reflect about certain aspects of the process of metadata creation. Some of these aspects were introduced with the intention that the answers could be related to, i.e. seen through the lens of, activity theory, which deals with issues such as what tools were used, what is the purpose of the activity, what official and unofficial rules exist around the activity, and who else is involved in the activity? Twelve metadata record creators were interviewed in a physical meeting or over the phone, and the interviews commonly lasted for one hour. All these interviews were recorded, while the thirteenth interview was done via an email-exchange (at the request of the interviewee). All of the interviews were summarized and analyzed and the result is described below in summary while the details can be found in paper 5.

When analyzing the interviews clear indications were given that on a high level the *objective* of the activity metadata creation is to enable search possibilities for different kinds of users. It is however not always clear who these user are. One interviewee working at a library reflected that metadata in libraries is often created in order to be understood by other librarians. Thus, the actual user of the metadata can be colleagues or other people creating metadata. In relation to the objective of the activity, several of the interviewees mentioned that they learn a lot while creating metadata, and that was seen as a motivational factor that made the work interesting.

The study also indicated that the main *tool* used for actually creating metadata is some kind of form-based metadata editor. However, other tools are used as well in different sub-activities in order to know what value to input. In these kind of sub-activities a web browser or a telephone could be utilized in order to find certain necessary information, for example finding the correct spelling of the author of a book in a library. The interviewees were also asked to reflect about automatic metadata creation, and to fully trust such a procedure was not considered suitable without any human intervention for the collections that the interviewees were involved in. Utilizing automatic approaches to provide suggestions in the metadata editor was considered a good help by many of the interviewees, especially if the suggestions were good and actually saved time. The possibility to copy metadata from similar resources was a suggested functionality that could potentially avoid repetitive work when metadata for several similar resources were created. This was emphasized by one interviewee as a suggestion for preventing the activity from getting tedious.

The kinds of *rules* that were brought up during the interviews revolved around the guidelines and rules about how metadata should be created. For example, how a book should be classified, or how the name of an author should be transliterated. This is related to the quality aspect “consistency and coherence” in table 1. The interviews gave answers that varied a lot. One interviewee described a workplace where few rules and guidelines were given and little effort was spent to improve the situation. Other interviewees described that they had regular meetings where the guidelines and difficult cases were discussed.

The *community* and the *division of labor* were more difficult to detect in this kind of broad study. However, examples were found where other people worked with creating metadata, sometimes in other organizations such as a publisher of books that provides metadata about their books before it is sent to the library.

In the study, activity theory was used in order to detect and analyze the main characteristics of the activity of creating metadata. This was achieved through interviews, and the results were found to be on a rather general level, since the study included people from different domains and varying backgrounds. In order to get a deeper understanding of the activity of metadata creation, more detailed studies have to be carried out and they should probably focus on one domain at a time. The study presented in paper 5 provides a basis for what aspects that would be interesting for further research. And, even if the paper perhaps provides few surprises to people working with creating metadata, it can still be useful by explaining the activity to people working with metadata in other ways.

5.2 Summary and reflection on the use of Design Science Research and Constructive Research

This chapter will go through the guidelines for DSR introduced in chapter 2.2 and describe, in a summarized manner, how they have been followed. For each guideline a connection is also made to the corresponding steps of CR. For each guideline some reflections are also included.

Problem relevance (DSR guideline 2)

In CR the problem relevance is considered in phase 1, where the importance of practical relevance is stressed.

The problem at hand had been recognized prior to the LUISA project, when applications using RDF data were developed within my research-group. The RDF data used in the applications had to be edited manually and which data elements that needed to be edited changed fairly frequently, at least during the development phase. Several different editors were also needed in those applications. In order to avoid writing new code each time a change occurred, the idea of a configuration mechanism for the RDF data editor emerged. The idea was included in the proposal of the LUISA project, where metadata editors were to be developed for the two use-cases of the project, one academic and one industrial. At that time it was not known exactly what type of metadata that would be used, but with the experience from development in earlier projects, the possibility to change a configuration to meet new needs in the metadata editor was taken into account. The problem detected was one that potentially can appear for programmers that are involved with creating form-based editors for metadata in RDF. By the start of the LUISA project (March 2006), RDF was not that widely used. The problem of creating RDF metadata editors was therefore perhaps not that widely considered so we aimed to create a solution where the

programmer could create an RDF metadata editor without knowing very much about RDF. This was carried out with the potential side-effect that the usage of RDF would increase. Another desired side-effect was that the metadata record creators would be enabled to change the metadata editors that they were using.

One of the problems that appeared within the LUISA project was that RDF metadata had to be updated on a remote basis, since it could not be assumed that the medatata editor always ran on the same machine as the storage module for the RDF metadata. At that time, the kind of solution that existed for updating RDF data remotely was a suggestion for a standard called SPARUL[57], that would extend the SPARQL standard. The update procedure of SPARUL consisted of a declaration of which triples to remove and which ones to insert. In a SPARUL declaration of which triples to remove it is not allowed to include blank nodes explicitly, though this can be done, but in a fairly complicated manner. We had an idea that we thought could provide a solution that was simpler and also easier to implement, and where the intended scope was the case of a user editing metadata in a form-based metadata editor. This was in contrast to the scope of SPARUL that was developed as a general solution for updating RDF on a remote basis.

The two problems presented above were actual problems, that was encountered in a project and a solution was required. The artifacts were developed to provide a solution that would be as generic as possible, so that it could be applied in similar situations. The problems considered were also seen as problems where the relevance might increase in the future. This estimate was based on the assumption that the Web of Data (i.e. the Semantic Web) would be realized and widely utilized and therefore frameworks and tools would be needed for managing such data (i.e. RDF).

Design as an artifact (DSR guideline 1)

Phase 3 of CR includes the building of a what in CR is called a construct. The construction process itself can be seen as the design of an artifact in DSR.

From the problems that have occurred, a set of artifacts were designed, aiming to solve the problems at hand. The Annotation Profiles framework can be considered to include several artifacts, including:

- a **model** in the form of the Annotation Profile Model (APM).

57 Now a part of the SPARQL standard, http://www.w3.org/TR/sparql11-update/#updateLanguage (last visited 21st Sept 2014)

- two **instantiations** in the form of the two implementations of the model in two different programming languages.
- two **constructs** in the form of a dedicated syntax to express an annotation profile and a remote update procedure for RDF data.

The APM was developed as an information model for the purpose of representing a form-based metadata editor where an instance of the model, i.e. an annotation profile, would be expressable in a machine processable way. The model was required to be designed so that an annotation profile could hold enough information for a machine to be able to create a form-based metadata editor from it. Each annotation profile can also be considered to be a configuration mechanism for a form-based metadata editor, and the updates made to an annotation profile should have the desired effect without having to change and recompile the code of the application at hand. It was also considered a good design practice to not restrict an annotation profile to be expressable only in a particular syntax.

In chapter 4.1 four categories of necessary requirements on the design were described of the APM. The actual design of the APM resulted in a division into two sub-models, the Graph Pattern model and the Form Template Model. The completeness category included the requirement that all possible expressions of RDF should be possible to edit and this was included in the Graph Pattern Model. The other categories (presentation, interaction, structure) were included in the Form Template model. The two artifacts in the form of code libraries that represent the APM were also developed as a way to enable easy reuse of code when metadata editors are to be constructed. The dedicated syntax has not been described in this thesis, but it deserves at least to be mentioned, since it is an artifact that has been constructed, and that corresponding representation of an annotation profile is the one used in the RDForms code library.

Another constructed artifact was the protocol for updating an RDF repository remotely. This artifact relies on existing initiatives found in standards[58] to know what RDF graph to extract and what triple to remove in the update steps. The artifact was made specifically for the situation where metadata was edited in a form-based manner. A set of requirements were set up for the design of the construct (see paper 3), requirements which were then adhered to in the construction of this artifact.

58 Such as the Describe query in SPARQL

Design evaluation (DSR Guideline 3)

In CR it is considered important that the construction (or artifact) actually is able to solve the actual problem, something which also has to be demonstrated, preferably in a realistic setting. This requirement is expressed in phase 4 of CR and constitute one type of evaluation. In phase 6 of CR, the scope of the solution should be examined, and this is another type of evaluation.

Two versions (=instantiations) of the APM have been implemented in the programming languages Java respectively Javascript. Different syntaxes to express the annotation profiles were also used in these implementations. In LUISA, the implementation was created in Java, and metadata was created for learning resources that were to be used in one of the test-cases within the project. The implementation in Javascript is nowadays called *RDForms* and it has been incorporated in the application *Confolio*. This application was used in the two projects *Organic Edunet*[59] and *HematologyNet*[60] and this implementation of the APM made Confolio easy to adapt to the kind of resources that were to be managed in those projects. During the projects, the utility of the model was evaluated through gathering feedback from the users and revising the design and implementation accordingly. This is further described below in guideline 6. Looking at table 2 in Hevner et al. (2004, pp.86) the evaluation carried out here corresponds to testing. The artifacts were part of a larger system that was used in several projects, a fact that enabled errors to be detected and quickly fed back and finally corrected.

The evaluation of the design of the APM, the construct of a dedicated syntax (to express annotation profiles) and the implementations of code-libraries were carried out in situations where we made use of the artifacts we had constructed. The end-users could then make indirect use of the APM since the form-based metadata editors were created from annotation profiles. Me and my colleagues within the KMR-group inbetween us took on the roles as programmers, annotation profile creators and facilitators. In order to further test the usefulness of the designs, it was deemed necessary to find cases outside of our own research group that used the artifacts. To enable this to happen, the code libraries that were developed were released as open source for anyone to reuse. As described in section 4.6 there are three cases known to us where the code library RDForms has been used by other developers. The people we have found that have used RDForms have all been interviewed about their usage of it. According to table 2 in Hevner et al. (2004) acquiring such knowledge can be considered to be observational evaluation. It is

59 See http://oe.confolio.org last visited 21st Sept 2014
60 See http://hematology.confolio.org last visited 21st Sept 2014

hard to say what information the number of downloads of the RDForms code library actually reveals. It has not been possible to find out how many of these downloads that led to actual usage and how many downloads that only lead to testing. It has also been impossible to establish how many programmers faced with the task to create a form-based metadata editor for RDF that have actually found out about RDForms.

The design evaluation of the construct for how to edit RDF metadata on a remote basis was carried out through testing. The artifact was implemented in code and included in a system that was used for editing metadata. This was an activity within the LUISA project and some errors were detected but they could be resolved. Later on, after the LUISA project, the approach of using named graphs was utilized instead.

Research contribution (DSR Guideline 4)

This design principle corresponds to phase 5 of CR, and it includes demonstrating the theoretical connections and the research contribution.

The contribution of DSR is commonly the artifacts themselves (Hevner et al., 2004). The artifacts presented in this thesis have contributed to new knowledge withing research about metadata in situations where metadata is exposed in a context of the Web of Data. As was shown in section 4.2, the initiatives and standards that exist do not provide the information needed in order to establish the functionality of the APM. The artifacts of the Annotation Profile framework demonstrated both that a configuration mechanism can be constructed, and also how this can be done. Later on it was discovered that the initial concept also could be expanded to include ideas of empowering end-users through end-user development (see paper 6). The APM could also be situated within the Cameleon framework, as described in section 4.5 together with a set of new roles that were deemed to be important for the future, when the usage of annotation profiles hopefully will increase.

The remote editing approach demonstrated that there was an alternative to using SPARUL when the metadata concerned is to be edited through a form. It also showed that the theoretical important and difficult case of deleting the correct blank nodes could be solved for this particular case.

Research rigor (DSR Guideline 5)

The design of the artifacts was grounded in the use of existing standards and initiatives relevant for metadata as well as for the Semantic Web, as described in sections 3.4 and 4.2. The design was also related to the idea of metadata interoperability, specifically to the possibility of creating combinations of metadata elements in so called application profiles. A

guiding principle in this process was to try to reuse as much as possible from the standards and initiatives that already existed. When guided by such a principle, it could be shown that the design was in line with what had been achieved before and in what ways it had been developed further. Such improvements include: 1) using SPARQL for expressing a graph pattern and 2) using both XML and RDF for the form templates, and 3) the dedicated syntax that was developed is expressed using JSON.

The common use of form-based approach for metadata creation was assumed in the process of developing the APM. At that time this assumption was not really questioned, and therefore it became the approach that I chose for this kind of activity. Hence, this approach underpins the study presented in paper 5, which also reports on finding out how the activity of metadata creation actually was carried out. The data gathered from this study further indicates that the underlying assumptions were true to a fairly high degree.

When the solution for remote editing was designed, the communication between the client (holding the metadata editor) and the server (holding the RDF metadata), was decided to be stateless. This meant that the server would not have to keep any information about the state of the client. The solution was inspired by the principles of REST as outlined by Fielding (2000, pp. 76-105), according to which the web was designed to function. It was also considered important to utilize existing web technologies as much as possible, with the exception of not using SPARUL for the update part of the server. The solution presented in this thesis was considered to be simpler, and it did make use of the SPARQL standard.

Phase 2 of CR involves gathering of information in order to obtain a general and comprehensive understanding. According to Piiranen and Gonzalez (2013), the main difference between research rigor in DSR and CR is that the literature about DSR stresses the importance of previous knowledge more than the literature of CR does.

Design as a search process (DSR Guideline 6)

The design of the artifacts were grounded in existing knowledge as described under guideline 5 above. Since much of the work with designing the artifact was part of a project that needed a solution, the work provided possibilities to evaluate and improve the artifact in situations of actual usage. For example, the implementation of the code libraries could be tested in an environment with real users. Bugs and errors that were not caught in the development phase would then appear as feedback from the users. This also provided an indirect test of the APM, since actual metadata used in practice was edited using the code

that implemented the model. One of the things discovered during the first actual use of the APM, as not having been thoroughly enough considered, was the different types of interaction and the complexity involved in handling all the different kinds of choice items were rendered as drop-down menus. This problem was later reconsidered, and the possibility to use radio-buttons was included. A way to present the possible values for an element in “a hierarchy tree” was also developed, something that requires information about how the possible values should be sorted into a hierarchy. Another desired feature that appeared in the Organic Edunet problem was the possibility to copy metadata from one record to another. This feature was acknowledged as useful, but deemed to be outside of the scope for the APM, and therefore it had to to be implemented in the surrounding application.

The first implementation of the APM into a code library was created using Java. The code library was used in an application that had to be installed on the local computer, and the application was built with the assumption that direct access to the RDF metadata was granted. As described in section 4.4, this assumption did not hold true and later a solution for remote editing of RDF metadata had to be developed. Moreover, the described solution was implemented as a web-application that made use of a proxy-server before the metadata editor could be presented to the user. Each time metadata was saved, the metadata editor had to be reloaded into the web-browser. Avoiding web-pages to be reloaded in the browser for each request to the server is a problem that is solved by the use of AJAX and related technologies. The first mentioning of AJAX was in an article published in 2005, one year before the LUISA project started (Garrett, 2005). Thus, AJAX was not used in the LUISA project since it was simply too new. The second approach to remote editing, developed later and using named graphs, (see chapter 4.4.2) makes use of AJAX when used in Confolio.

It was considered to be a good design choice for the APM not to enforce a specific syntax in order to express an annotation profile. This was in the spirit of RDF, which is also not locked to a specific syntax. However, this “syntax freedom” when describing the ideas behind the APM seemed to cause confusion. Therefore a dedicated syntax, using JSON, was developed.

The example discussed here aim to demonstrate that the artifact was not finished after the first development step but had to reconsidered and redesigned iteratively.

Communication of research (DSR Guideline 7)

Apart from communication by publishing research papers, there were several reports written in the LUISA project about the APM and the use of annotation profiles. These were technical reports aimed both at programmers but also at technical managers in order for them to understand the artifacts. The code libraries developed were also released, and published on the WWW as open source code for anyone to discover and use. The documentation of the code, together with examples on how to use it, was also published along with the code itself. Moreover, a Google group for the code library was set up where questions and issues related to RDForms can be discussed.

5.3 The contributions to research and community around metadata

Much of the research presented in this thesis is of an applied nature. As described in the previous section, artifacts have been constructed and evaluated. A study aimed at discovering the main characteristics of the activity of creating metadata has also been performed through interviews with people working as metadata record creators. Moreover, ideas on how to empower the metadata record creator to manipulate the metadata editor, via end-user development, has been presented.

As previously stated, the study of metadata record creators in paper 5 was meant to be explorative, but also to uncover the "cross-disciplinary" characteristics of metadata creation in more than one domain. The findings in that paper provide a basis for further research around these activities. To the knowledge of the author, no other attempt has been made to frame the activity of creating metadata within activity theory.

The attempt to design an adaptable metadata editing environment lead to a set of artifacts presented in the previous sections. Much of the documentation and code developed have been made available to the public, and hence also to the metadata community. This contribution is intended to make it easier to both create and update a form-based metadata editor. Moreover, this kind of configuration mechanism can shorten the turn around when there is a need to update certain parts of a metadata editor. In general, the research presented here can be seen as an extension to the research that has been made about application profiles (that were described in section 4.2.2).

The latest code library developed, RDForms, has also been used by system developers. From those cases, a contribution to the metadata community has been made. Furthermore, an attempt to empower the

metadata record creator has been done, by enabling end-user development as described in paper 6. Whether or not this is a reasonable approach for the editing needs of the rapidly evolving Internet calls for further research.

6. Final reflections and discussion

The approach to the research reported in this thesis is of a type that intervenes with the world. That is, I as a researcher have not just observed the phenomenas in the world but I have also constructed/designed something that potentially can change something in the world. Or, at least it can provide one or more artifacts that constitute a better solution to a certain problem. Performing research in this manner involves several difficulties. For example, the situation that the artifact is primarily constructed for can impact the evaluation of the artifact. The fact that an artifact that was designed for an organization could not solve the targeted problem might be caused by other problems within the organization, problems that were out of control of the researcher. In such cases, the artifact might solve the same kind of problems in similar organizations. It could of course also be the other way around. The artifact that actually solved the problem in the situation for which it was developed, though it cannot be shown to solve similar problems outside of that situation. Another potential problem is described in Kasanen et al. (1993, pp 252) where the constructed artifacts work so well that the participating organization wants to keep the artifact as a business secret, which can delay publication of the results.

In the introductory sections of this thesis metadata was introduced and the problem of interoperability was brought in focus. The reason why this problem is considered to be important is that an adaptable metadata creation environment should make it possible to edit any kind of combination of elements. RDF is then considered to be the best option, since its data model, according to Nilsson (2010), is the best choice for a common data model that metadata standards should adhere to in order to achieve the level of machine semantic interoperability. This means that metadata elements from several metadata standards can be combined in a single metadata record, and the complete record can also be interpreted by a machine. The adaption possibilities in the metadata editing environment enabled by the APM consists in controlling what metadata elements that can be created, and edited. The APM is dependent on the RDF data model in order to effortlessly combine elements across different metadata standards.

A final reflection about myself and my development as a researcher: When I started the work that led up to this thesis I was a person that mainly focused on technology and thought that if we could only improve the technology, many of the problems found in relation to it could be solved. I still think that technology is an important part, but I have found new ways to locate the actual problems and I have realized that they

cannot always be solved by technical development. This kind of change is reflected in this thesis, especially when compared to my licentiate thesis (Enoksson, 2008). The latter focused more or less only on the technology, while this thesis I have also tried to investigate and understand the reality into which I have intervened. Paper 5 was one result of this expanded research scope, where I looked into how metadata is created in practice and the study presented there would ideally have been one of the first studies carried out and also the first paper written for this thesis. The smaller study presented in paper 6 was also performed in order to gain knowledge and understanding on how the artifact was used and how useful it actually is.

6.1 Future work

From my point of view the Web of Data provides a very powerful ways to semantically connect various sources of information. For example, connecting metadata records across collections, that previously have been isolated. Such connections have the potential to enrich the context of the metadata records. How different collections and different domains were established reflect a collective view of how the world of descriptions should (or could) be partitioned. Moreover, the kinds of metadata created for a collection reflect the aspects that are seen as important for the corresponding domain. Even if metadata standards are developed to the point of vertical harmonization, i.e. they become fully interoperable , the way that metadata records are created in one collection might differ from how they are created for another collection. Thus, connecting or linking the information might be difficult even if the semantic co-existence level has been reached. The difference between the metadata records of two collections can appear in what metadata elements that are used and also in what values that are used. Thus, how to actually make the connections between metadata records across the web *and* across domains is something that needs to be investigated further.

Within the cultural heritage community no single standard, recommendation or practice for metadata has received such a wide uptake that it has become the de facto standard within this community. The so called memory institutions in this community have often created metadata for the purpose of managing their collections internally. Moreover, they have often made their choice of standard based on this fact that metadata is only used internally, but perhaps not always used it in a correct way. The metadata records were not created to be linkable to other metadata records, and the systems that are used are not designed for "linkability" either. Therefore, the type of research presented above is interesting to pursue within this community. However, initiatives taken

to remedy this situation have resulted in the CIDOC CRM[61], a conceptual reference model aimed at promoting a shared understanding of information within the cultural heritage sector. Our research group is presently involved in the project CultureBroker[62] where a reference model called synergy is being developed to promote better practice for data provision and metadata aggregation processes. These efforts are initiated with the intent to investigate existing practices and develop recommendations for practices when the metadata will be exposed outside of the organization. These efforts are also a part of the work being done within the European network CultureCloud and it also involves the project ResearchSpace[63] at the British Museum financed by the Andrew W Mellow Foundation[64].

A suggestion for how end-user development can be further improved with a graphical user interface has been presented in paper 6. But, what are the actual interests of the metadata record creators in terms of what they want to have control over? Is it to be able to choose what metadata elements that are possible to edit? Or, is it perhaps more interesting to "only" have control of the possible values that can be set for a metadata element? The interest could of course also be different and these kind of aspects call for further research.

In a metadata editor, the set of possible values for certain metadata elements is often fixed and cannot be changed. For example, only certain values from a vocabulary are valid. However, the best way to describe a thing in a collection will change over time, and in order to uphold the quality of the metadata the corresponding thing (=target of description) has to be described in a timely manner (see aspect timeliness in table 1, pp. 26). Can these changes to how things are described be captured in some way, and what is the suitable way to support this in a metadata editor? These are aspects of adaptability in the metadata editing environment that have been recognized as important topics for future research. This kind of research can be carried out with the APM and also include the ideas and issues involved in of end-user development.

61 http://www.cidoc-crm.org/ last visited 20th Oct 2014.
62 http://www.culturebroker.eu/ By the time this thesis was printed this web-site was not yet up and running, but this is the correct address where information about the project will be published.
63 http://www.researchspace.org/ last visited 20t Oct 2014.
64 http://www.mellon.org/ last visited 20th Oct 2014.

7. References

Berners-Lee, T. (2006) *Linked Data*. Available: http://www.w3.org/DesignIssues/LinkedData.html Last accessed 19th Sept 2014.

Berners-Lee, T., Fielding, R., Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax*. Available: http://tools.ietf.org/html/rfc3986 Last accessed 21st Sept 2014

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) '*The Semantic Web*', Scientific American, Available: http://www.scientificamerican.com/article/the-semantic-web/ Last accessed 14th July 2014

Bizer, C., Heath, T. and Berners-Lee, T. (2009) Linked Data - the story so far. *International Journal on Semantic Web and Information Systems*, 5 (3), pp. 1-22.

Bratteteig, T. (2007) Design research in informatics. Scandinavian Journal of Information Systems. 19 (2). p. 66-74

Brickley, D., Guha R.V. (Eds.) (2014). *RDF Schema*. Available: http://www.w3.org/TR/rdf-schema/ last accessed 21st Sept 2014

Bruce, T. R., & Hillman, D. (2004). The Continuum of Metadata Quality: Defining, Expressing, Exploiting. In Hillmann, D. (Ed), *Metadata in Practice*. American Library Association, Chicago, USA. pp. 238-253.

Bödker, S. (1989) A Human Activity Approach to User Interfaces. *Human-computer interaction.* 4 (3). pp. 171-195

Carroll, JJ,. Bizer, C., Hayes, P., Stickler, P. (2005). Named graphs, provenance and trust, Proceedings of the 14th international conference on World Wide Web, Chiba, Japan. pp. 613-622.

Calvary G., Coutaz J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. (2003). A Unifying Reference Framework for multi-target user interfaces, *Interacting with Computers*. 15 (3). pp. 289-308. ISSN 0953-5438.

Costabello, L. (2013) The PRISSMA Vocabulary v2 Namespace Document. Available: http://ns.inria.fr/prissma/v2/prissma_v2.html Last accessed 21st Sept 2014.

Coyle, K., Baker, T. (2009). Guidelines for Dublin Core Application Profiles. Available: http://dublincore.org/documents/profile-guidelines/ Last accessed 21st Sept 2014

Cygniak, R., Wood, D. and Lanthaler, M. (2014) *RDF 1.1 Concepts and Abstract Syntax*. Published as a W3C Recommendation 25 February 2014.Available: http://www.w3.org/TR/rdf11-concepts/ Last visited 21st Sept 2014.

DCMI Usage Board (2012). *DCMI Metadata Terms*. Available: http://dublincore.org/documents/dcmi-terms/ Last visited 21st Sept 2014.

Dodig-Crnkovic, G. (2010). Constructive research and info-computational knowledge generation. In L. Magnanig, W. Carnielli & C. Pizzi (Eds). *Model-Based Reasoning in Science and Technology*. pp. 359-380. Berlin Heidelberg: Springer.

Doerr, M. (2003) The CIDOC Conceptual Reference Module An Ontological Approach to Semantic Interoperability of Metadata. *AI Magazine archive*. 24 (3), pp. 75-92

Daniel, F., Jin Y., Boualem B., Casati, F. Matera, M. and Saint-Paul, R (2007). Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*. 11(3), pp. 59–66. ISSN 1089-7801.

Ebner, H., Manouselis, N., Palmér, M., Enoksson, F., Palavitsinis, N., Kastrantas, K., & Naeve, A. (2009). Learning object annotation for agricultural learning repositories. In *Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies*. IEEE. pp. 438-442.

Ebner, H. and Palmér, M. (2014) An information model for managing resources and their metadata. *Semantic Web*. 5 (3). p. 237-255. Available: http://iospress.metapress.com/content/123u6702x0626896/ Last visited 11th Sept 2014.

Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to developmental research* (Doctoral dissertation). Helsinki: Orienta-Kunsultit

Enoksson, F. (2011) *Flexible Authoring of Metadata for Learning: Assembling forms from a declarative data and view model*. Licentiate thesis. KTH - Royal Institute of Technology, Stockholm, Sweden. ISBN: 978-91-7415-965-3

Enoksson, F (2008). *A MoinMoin Wiki Syntax for Description Set Profiles*. Available: http://dublincore.org/documents/dsp-wiki-syntax/ last visited 21st September 2014

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Diss. University of California, Irvine, USA.

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., & Mehandjiev, N. (2004). Meta-design: a manifesto for end-user development. Communications of the ACM. 47 (9). pp. 33-37.

Gangemi, A., Fisseha, F., Pettman, I. and Keizer, J (2002) Building An Integrated Formal Ontology for Semantic Interoperability in the Fishery Domain., 2002 . In *International Semantic Web Conference (ISWC)*, Sardinia (Italia), June 9-12 2002.

Garrett, J.J. (2005) *Ajax: A New Approach to Web Applications*. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/ Last visited 21st Sept 2014.

Golder, D., Kneebone, L., Phipps, J., Sunter, S. and Sutton, S.A. (2010) A Configurable RDF Editor for Australian Curriculum. In Chowdhury, G., Koo, C. and Hunter, J. (eds) *The Role of Digital Libraries in a Time of Global Change*, Lecture Notes in Computer Science. Springer Berlin Heidelberg. pp. 189-197.

Greenberg, J., Spurgin, K. and Crystal, A. (2005). *Final report for the amega (automatic metadata generation applications) project*. School of Information and Library Science, University of North Carolina at Chapel Hill, USA.

Greenberg, J., Crystal, A., Robertson, D. and Leadem, E. (2003). Iterative Design of Metadata Creation Tools for Resource Authors. International Conference on *Dublin Core and metadata applications*. Seattle, Washington. pp. 49-58.

Harris, S. and Seaborne, A. (2013) *SPARQL 1.1 Query Language*. Available: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ Last visited 21st Sept 2014.

Heery, R. & Patel, M. (2000). Application profiles: mixing and matching metadata schemas, *Ariadne 25*, ISSN 1361-3200.

Hevner A.R., March S.T., Park, J. and Ram, S. (2004) Design science in information systems research. *MIS quarterly*. 28 (1). pp. 75-105.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F. & Rudolph S. (Eds.) (2009). *OWL 2 Web Ontology Language Primer (Second Edition)*. Available: http://www.w3.org/TR/owl2-primer/ Last visited 21st of Sept 2014.

Hopcroft, J. E. and Tarjan, R. E. (1973) A v log v algorithm for isomorphism of triconnected planar graphs. *Journal of Computer and System Sciences*. 7 (3), pp. 323-331.

Hopcroft, J. E. and Wong, J. K. (1974) Linear Time Algorithm for Isomorphism of Planar Graphs (preliminary Report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*. pp. 172-184 New York: ACM.

Hornby, N. (1996). *High Fidelity*. London: Indigo

Hunter, J. (2003) Working towards MetaUtopia-a survey of current metadata research. *Library Trends*. 52 (2). pp. 318-344

IEEE Computer Society (2002) *IEEE Standard for Learning Object Metadata*. IEEE Std 1484.12.1-2002. The Institute of Electrical and Electronics Engineering, Inc. New York, US.

Iivari, J. (2007) A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems*. 19 (2). pp. 39-64

ISO/IEC 15938-2 (2002) *MPEG-7 (Multimedia content description interface), Part 2: Description definition language*, International Organization for Standardization, Geneva, Switzerland

Kaptelinin, V. (2005). The object of activity: Making sense of the sense-maker. *Mind, Culture, and Activity*, 12 (1), pp. 4-18.

Kaptelinin, V., & Nardi, B. A. (2006). *Acting with technology: Activity Theory and Interaction Design*. The MIT Press.

Kaptelinin, V., Nardi B. A. & Macaulay, C. (1999) Methods & tools: The activity checklist: a tool for representing the “space” of context. *Interactions*, 6 (4). pp. 27-39.

Kasanen, E., Lukka K. & Siitonen, A. (1993). The constructive approach in management accounting research. *Journal of Management Accounting Research*, 5, pp 241-264.

Klerkx, J., Duval, E. and Meire, M. (2004) Using information visualization for accessing learning object repositories. In *Proceedings of the eighth International Conference on Information Visualization*. pp. 465-470

Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B. A., Rosson, M. B., Rothermel, G., Shaw, M., Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys*. 43 (3). pp. 1–

44.

Lazar, J., Feng, J.H. and Hochheiser, H. (2010). *Research Methods in Human-Computer Interaction*. Wiley Publishing.

Leont´ev, A. N. (1978). *Activity, consciousness, and personality* . Englewood Cliffs, N.J.: Prentice-Hall

Lukka, K. (2003). The constructive research approach. In L. Ojala & O.P. Hilmola (Eds.) *Case Study Research in Logistics*. pp. 83-101. Turku School of Economics and Business Administration.

Lieberman, H., Paterno, F., Klann, M. and Wulf, V. (2006) End-user development: An emerging paradigm. In: Lieberman, Henry, Paterno, Fabio and Wulf, Volker (eds.). *End User Development (Human-Computer Interaction Series)*. Springer. pp. 1-8

March, S.T. and Smith, G.F. (1995) Design and natural science research on information technology. *Decision Support Systems*. 15 (4). pp. 251-266.

Matsou, Y. and Ishizuka, M. (2004) Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*. 13 (1). pp. 157-169

Mehandjiev, N., Sutcliffe, A., & Lee, D. (2006). Organizational view of end-user development. In Lieberman, H., Paternò, F. and Wulf, V. (Eds.) *End User Development*. Springer Netherlands. pp. 371-399.

Meixner, G., Seissler, M. and Breiner, K. (2011). Model-Driven Useware Engineering. In Hussmann, H., Meixner, G. and Zuehlke, D. (Eds.) *Model-Driven Development of Advanced User Interfaces*. Springer Berlin Heidelberg. pp. 1-26.

Naeve, A. (2005) The human Semantic Web shifting from knowledge push to knowledge pull. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 1 (3), pp. 1-30.

Nardi, B. A. (1993) A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge, MA, USA.

Myers, B.A. and Rosson, M.B. (1992). Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, New York, USA. Pages 195–202. ISBN 0897915135

Nilsson, M. (2010). *From interoperability to harmonization in metadata standardization: Designing an evolvable framework for metadata harmonization*. Diss. KTH - Royal Institute of Technology, Stockholm, Sweden. ISBN: 978-91-7415-800-7

Nilsson, M., Baker, T., Johnston, P. (2008). The Singapore Framework for Dublin Core Application Profiles. Available: http://dublincore.org/documents/singapore-framework/ Last visited 21st September 2014.

Nilsson, M., Miles, A.J., Johnston, P., Enoksson, F. (2007) Formalizing Dublin Core Application Profiles: Description Set Profiles and Graph Constraints. In Sicilia M.A., Lytras M.D. (Eds). *Metadata and Semantics, Post-proceedings of the 2nd International Conference on Metadata and Semantics Research*, Corfu Island in Greece. Springer. pp. 101 – 111

Nilsson, M., Palmér, M., & Brase, J. (2003). The LOM RDF binding: principles and implementation. In *Proceedings of the Third Annual ARIADNE conference*, Leuven Belgium, 2003.

Ossenkoppele, G., Evans-Jones, G., Jaeger, U. and Hellström-Lindberg, E. (2012) Towards a joint definition of European hematology. *Haematologica* 97 (5). pp. 636-637. ISBN 1592-8721.

Palmér, M., Enoksson, F., Nilsson, N. and Naeve, A., (2007) *D3.2: Annotation Profile Specification*, Deliverable within the project LUISA. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-152668 Last visited 30th of September 2014.

Pietriga, E., Bizer, C., Karger, K., Lee, R. (2006), Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In *Proceedings of the 5th International Semantic Web Conference, ISWC 2006*, Athens GA, USA. pp. 158-171.

Piirainen, K.A., and Gonzalez, R.A. (2013) Seeking constructive synergy: Design science and the constructive research approach. In vom Brocke, J., Hekkala, R., Ram, S. and Rossi, M. (eds.). *Design Science at the Intersection of Physical and Virtual Design*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer

Powell, A., Nilsson, M., Naeve, A., Johnston, P., Baker, T. (2007). *DCMI Abstract Model*. Available: http://dublincore.org/documents/abstract-model/ Last visited 21st September 2014.

Powell A. (2005). *People's Network Service: Discovery Service Dublin Core Application Profile (PNDS DCAP)*. Available http://www.ukoln.ac.uk/metadata/pns/pndsdcap/ Last visited 21st Sept 2014

Prud'hommeaux, E., Seaborne, A. (2008). *SPARQL Query Language for RDF*. Available: http://www.w3.org/TR/rdf-sparql-query/ Last visited 21st of Sept 2014.

RDF (2014), *Resource Description Frame work*. Available: http://www.w3.org/RDF/ Last visited 21st Sept 2014.

Rodriguez, M.A., Bollen J., and Van De Sompel, H. (2009) Automatic metadata generation using associative networks. *ACM Transactions on Information Systems (TOIS)* 27(2). pp. 1-20.

Sánchez-Alonso, S., Cáceres, J., Holm, A. S., Lieblein, G., Breland, T.A., Mills, R.A. and Manouselis, N. (2008). Engineering an ontology on organic agriculture and agroecology: the case of the Organic. Edunet project. In *Proc. of the World Conference on Agricultural Information and IT (IAALD AFITA WCCA 2008)*, Tokyo, Japan pp. 24-27.

Studer, Grimm, and Abecker (2007), *Semantic Web services: Concepts, Technologies, and Applications*. Springer.

Sutton, S.A. (2008). Metadata Quality, Utility and the Semantic Web: The Case of Learning Resources and Achievement Standards. *Cataloging & Classification Quarterly*. *46*(1). pp. 81-107.

Vygotsky, L.S. (1962). *Thought and Language*. Cambridge MA: MIT Press.

Waugh, A. (1998), Specifying Metadata Standards for Metadata Tool Configuration, *Computer Networks and ISDN Systems,* 30, pp. 23-32.

8. Summaries of included papers

Summary of Paper 1: An approach for on- demand e-learning to support knowledge work

***Year**: 2007*

***Authors**: Stefan Thalmann and Fredrik Enoksson*

Paper published in Tochtermann, K., Maurer, H. (Eds.): Proceedings of I- KNOW´07. 7th International Conference on Knowledge Management, Graz, Austria, September 5-7, 2007.

The author's main contribution to this paper was the chapter about application profiles

In this paper a need for combining metadata elements is discussed and the idea of application profiles is considered as one solution. This in a case of so called knowledge work. E-learning is described as a continuous lifelong process, in contrast to knowledge work, where learning in a typical work situation is seen as a process that will result in acquiring knowledge to solve a current problem. In such a situation it is optimal to consider the context of the person, here called a knowledge worker. Such information could be retrieved from a profile of the knowledge worker, for example geographical location, operative system used, and the existing knowledge the knowledge worker claim to already have. This information would typically be included in a search for relevant resources to solve a certain problem, which in turn would require that these resources have been annotated with appropriate metadata. Thus, a situation occurs where metadata elements from different standards need to be combined with metadata elements that perhaps only provide meaning inside a specific organization.

In this paper IEEE LOM is presented as an alternative to serve as base schema, to which the other descriptions could be added. Then additional metadata descriptions could be added as extensions. In the conclusion to this paper, the problem of interoperability is brought up, since the proposed solution only creates interoperability within a given set of standards. When a new standard is to be used, a combination of metadata elements cannot be guaranteed to be achievable in a simple manner.

Summary of Paper 2: Annotation profiles: Configuring Forms to Edit RDF

***Year**: 2007*

***Authors**: Matthias Palmér, Fredrik Enoksson, Mikael Nilsson, Ambjörn Naeve*

Paper presented at the International Conference on Dublin Core and Metadata Applications, Singapore 28 - 31 August 2007.

The author's main contribution to this paper are chapters 4 and 5

This paper starts with a discussion about how editors for metadata can be divided into 3 categories *fixed, configurable* or *generic*. It is concluded that fixed annotation tools provide little or no adaptability, whereas a generic annotation tool is very flexible with respect to the metadata it can edit. But, this would more or less mean to edit the syntax that the metadata is expressed, in and this would leave a lot of responsibility to the user when it comes to complying with the syntax of the metadata expressions.

Configurable annotation tools would add a user-friendly interface that can hide the underlying syntax but still be adaptable. From the assumption that the metadata to edit will be represented in RDF, the requirements for a configuration mechanism for annotation tools is discussed and divided into four categories:

- **Completeness** - requires that any correct RDF structure should be possible to edit.
- **Structure** - includes ordering and grouping of the different items of the interface, as well as cardinality constraints.
- **Interaction** - hints on how the value to be edited should be presented to the user, e.g. by drop-down menus, radio-buttons, check-boxes etc.
- **Presentation** - includes possibilities to express labels and descriptions in order to provide the user with hints as to what value to provide. Such hints should also be possible to express in more than one language.

An information model called the *Annotation Profile Model* is introduced as a candidate to meet these requirements in a way that is both language- and platform-independent. The purpose of this model is to provide a configuration mechanism that is specific enough so that a metadata editor can be created from it.

Summary of Paper 3: An RDF modification protocol, based on the needs of editing tools

Year: 2007

Authors: Fredrik Enoksson, Matthias Palmér, Ambjörn Naeve

Paper presented at the 2nd International Conference on Metadata and Semantics Research, MTSR 2007, Corfu Island, Greece, 1-2 October 2007.

The author wrote the major parts of this paper

This paper presents an approach to how RDF metadata can be edited when stored on a computer that can only be accessed on a remote basis. The main problem that occurs in this situation is how to handle the blank nodes in RDF in a correct manner. The design of a protocol that can edit RDF is presented, and the following four categories of requirements that are introduced:

- **Resource centric** - all kinds of subgraphs that are reachable from a named node[65] (possibly via intermediate blank nodes) should be modifiable.
- **Concise modifications** - the modification request should be concise and also efficient, i.e. neither too much nor too little data should be transferred.
- **Without side effects** - The parts of the RDF metadata that cannot not be modified by the metadata editor should be left untouched.
- **Application independent** - There should be no built-in knowledge in the protocol about specific properties or resources in RDF.

From these requirements, possible solutions are discussed and the following two naive solutions are discarded: 1) transferring updates of one triple at a time, since it would yield a "chatty" protocol, 2) transferring the whole RDF graph, which can result in large amounts of unnecessary data being transferred. Other solutions that fall short on how to deal with the blank nodes are discussed, notably SPARUL (an extension to SPARQL), which is seen as a possible solution for editing the remote RDF graph if the structure of the graph is known in advance. This is a requirement that will not always cover the situation where metadata is edited via forms that are adaptable.

65 A resource identified through a URI

Summary of Paper 4: Using a hematology curriculum in a web portfolio environment

***Year**: 2011*

***Authors**: Fredrik Enoksson, Ambjörn Naeve, Eva Hellström-Lindberg*

Paper published in Knowledge Management & E-learning: An international journal, special issue on Advances in Health Education Applying E- learning, Simulation and Distance Technologies

The author wrote the major parts of this paper

This paper describes a use case of annotation profiles in a real setting, where a system of web-portfolios[66] has been customized to a the professional community of European hematology. One of the purposes for creating this system was to provide the possibility to describe learning resources related to hematology, i.e. to express metadata on such resources and to classify them according to different levels of difficulty. Another purpose was to provide the users with the possibility to describe their competence in the field of hematology, expressed as metadata about each user. The metadata editors for both the learning resources and the competence profiles were generated from annotation profiles.

Before the development of the web portfolio system, the European Hematology Association had developed a so called Curriculum Vitae pass port (CV passport) that served as a competence profile for students and professionals of hematology. The CV passport consists of subcategories that contain so called competence-items. For each such item, the hematologist is meant to express her/his level of competence. In this paper, a digital representation of the CV-passport (in the form of an RDF vocabulary) is described and a competence editor is introduced that has been created from an annotation profile. Annotation profiles are also used to provide editors for learning resources where it is possible to express that a particular resource deals with a certain part of the CV-passport. This enables a set of search functionalities which are described further in this paper.

As all medical fields, hematology is evolving, and this requires updates of the CV-passport in order for it to stay up to date. The paper discusses how the use of annotation profiles makes such updates easier. Another aspect highlighted in the paper is that other related medical fields (for example oncology) can reuse parts of the CV-passport, annotation profiles provide an easy way to create the necessary editors.

66 See http://www.ehaweb.org/congress-and-events/online-learning-tools/hematology-confolio (last visited 10th Oct 2014)

Summary of Paper 5: The activity of human metadata creation and the Semantic Web

***Year**: 2014*

***Authors**: Fredrik Enoksson and Olle Bälter*

Paper accepted for publication in the International Journal of Metadata, Semantics and Ontologies (ISSN: 1744-2621)

The author wrote the major parts of this paper

The paper includes an explorative study of the human activity of creating metadata. The study was intended to be broad in order to arrive at an understanding of this activity across domains and to find potential commonalities. When metadata is created, it can potentially become a part of the Semantic Web, where the domain boundaries for the metadata will be removed since metadata records can be linked across different domains. But will metadata records in fact be linked across such boundaries? And what obstacle are there to overcome? The study reported in this paper serve as a starting point for the search of an answer for these question.

A total of 13 participants were included in the study, and each participant was interviewed about the activity of creating metadata. One interview was carried via an exchange of emails, while the rest were performed either in a face-to-face meeting or over the phone. These interviews were semi-structured, and the questions were constructed in order to let the interviewee reflect upon the activity of creating metadata.

The study was inspired by activity theory and one intent was to find the commonalities by mapping them to the activity system. Apart from the human subject an activity system also includes the following: 1) The *object,* which is tightly connected to an *objective* for the activity, 2) the *tools* used, 3) the *rules* involved, 4) the *community* around the activity and 5) the *division of labour* involved in the activity. For 4) and 5) no real common denominators were found. The kind of *tool* that was used for creating metadata was a form-based metadata editor. The study revealed a potential common *objective* in enabling search-possibilities within a collection of things. The *rules* of the activity existed in the form of guidelines on how metadata should be created, which were considered to be important, but not always well documented. The last part of the paper discusses the potential impact of metadata creation on the efforts to establish a Semantic Web, and such guidelines are considered an important factor.

Summary of Paper 6: Towards End User Development for metadata creators

***Year**: 2014*

***Authors**: Fredrik Enoksson and Filip Kis*

Submitted paper

The paper as a whole was a joint effort between the two authors of the paper, where Enoksson wrote the main parts of the paper and performed the interviews. Kis' main contribution to the paper are the parts that deals with end-user development.

This paper describes how the *Annotation Profile Model* (APM) and *RDForms* (an open source licensed code-library that implements the APM) has enabled end-user development (EUD) for metadata record creators. This has the potential to reduce the need for a software developer when form-based metadata editors have to be constructed.

The paper can be divided into two parts. The first part describes how the APM and RDForms can be utilized by software developers to create the metadata editors in an application. Three cases are described where RDForms have been used “in the wild” by software developers. The three cases show that RDForms was considered useful as part of a solution to actual real tasks that the developer had to complete. The second part of the paper discusses how end-user development has been enabled when an application makes use of RDForms to create metadata editors in an application. However, in order to adapt the metadata editor, a fairly complex data structure in JSON needs to be manipulated. An alternative to that solution is presented, which includes a graphical user interface. This user interface is based on the form items of the APM, since each form item often corresponds to a metadata element. The input to each form item is edited via form item editors, and the resulting metadata editor is directly presented in order to provide direct feedback to user.

To reduce the need of a developer will naturally benefit organizations that have limited resources. Therefore, the kind of user that is in focus is not primarily the professional metadata creator, but subject enthusiasts that work in non-profit organizations. Examples given in the paper are (small) local museums. RDForms alone will not make such organization independent of developers, but it has the potential to provide a solution where metadata in RDF can be edited and, if desirable make that metadata a part of the Web of Data. The code library is also independent of the technical solution with respect to how metdata is stored and it does not restrict the set of metadata elements that can be edited.