UNIVERSITY OF OSLO
Department of Informatics

Juristopia

Semantic Wiki for Legal Information

Master thesis

Ole Christian Rynning

1st November 2007

Preface

Juristopia: Semantic Wiki for Legal Information

Keywords: Wiki, Topic Maps, Semantic Web, Semantic Wiki, Legal Information System, Legal Knowledge-Based System, Legal Knowledge Management

Abstract

This thesis presents the role of Semantic Web technology and in particular Topic Maps in the legal domain. It also presents a prototype and concerns for a system utilising these technologies and use of this particular Information Technology in the legal domain. The thesis also discusses the theoretical background for such systems, and the research explores technology legal professionals can benefit combining social text writing and Semantic Web technology.

The thesis is supported by a Semantic Wiki, Juristopia, which has been my prototype for testing ideas, and used in discussion and reasoning on how the Information System Design components can be used together to the benefit of legal professionals.

Acknowledgements

What has brought me to the final idea of this thesis is a series of coincidences and factors hard to describe. Initially I wished to focus on Topic Map Technology and Lovdata, the main source of Norwegian Legal Information. The aim was to aid text searching and improve “searchability” and “findability” in their vast bases. This proved to be a bigger task than anticipated, and was very hard to constrain myself into a Master Thesis.

This thesis is written at Department of Informatics, University of Oslo. During the last year of my thesis I was employed as a Research Assistant at the Norwegian Research Centre for Computers & Law (NRCCL), Department of Private Law.

I wish to thank all the people at NRCCL for giving me the chance to work there as a Research Assistant, Jon Bing in particular for the inspiration and advice during writing this thesis. Also thanks to Gisle Hannemyr at Department of Informatics.

Special thanks to my predecessor at NRCCL, Kirill Miazine, for being my sparring-partner and helping me to refine all my ideas. His invaluable input, feedback and shared vision on this technology have helped a lot.

Table of Contents

Table of Figures

Part I

Introduction

1. Introduction

1.1 Motivation

The Semantic Web is an exciting set of technologies, and a natural successor of the current web. The Semantic Web is both a vision, and a set of technologies enabling this vision. The Semantic Web intends to knit information together, by improving "searchability" and enable us to navigate information across different web information systems. A goal of the Semantic Web is to use elements of Artificial Intelligence to make computers capable to assist us in finding the relevant information. This is to be done by attaching semantic value; to describe the relationships between information and information resources. By this one intend to add meaning to the information, so that computers can assist us in processing the vast amounts of data from multiple sources, and transform it into chunks of consolidated information, which is relevant to us.

1.2 Problem definition

There is a considerable effort in Legal Information Systems research in regards to the Semantic Web, but the main focus lies on the W3C stack of suggested technologies, as opposed to Topic Maps technology. Most papers discuss how rules and ontologies can be used for creating different models of how computers can find and process information. I aim to let the users of the system design the ontology, based on the content they author, and how a Semantic Wiki can be used in the process of documenting Legal knowledge.

1.3 Research Goal

The main goal of this thesis is to explore Semantic Web technologies and Topic Map technologies can assist in administering legal knowledge, and legal method - or legal process knowledge and experience can be supported by a simple Legal Information System in the form of a Wiki.

In the thesis I will explore the possibilities of creating a LIS/KBS, codenamed "Juristopia". Juristopia is a Semantic Wiki system, meaning it combines Wiki-system features such as

collaborative text writing and revision control, with Semantic Web features in the form of a knowledge representation layer.

I also aim to explore features from other web-systems such as users and groups/spaces, access control, comments, page ranking, groups/spaces, and most important a visualisation application for the knowledge layer.

For exploring these tasks, I will use a high-level programming language, Ruby with an agile web-application framework called Rails, or Ruby on Rails (RoR). I will delve into several technologies in the process. However; one of the main technologies I will apply in creating the knowledge representation layer is Topic Maps.

1.4 Context

Ever since Intel-founder Gordon Moore in his 1965 paper coined what is now referred to as Moore's Law; that CPU power, memory, storage and network capacity will increase exponentially, almost doubling every two years, and will continue for at least another half decade. This leap in technology has together with the rise of the Internet (Hanseth, 2002) and the hyperbolic adoption-rate-success of the World Wide Web (Hannemyr, 2003) has created a new era in Information Science.

There has been an explosion in the availability of digital information, creating new giant information bases like online encyclopaedia Wikipedia, video-sharing site YouTube, social networks like LinkedIn and Facebook and generating a multi-billion dollar enterprise of search-engine companies like Google.

The number of Information Systems has also increased exponentially, also following Moore's Law. Nearly every business or organization uses at least one information system. Some information systems are tailored to support their users, while others are adapted into the user's environment. All in all, just about every human use an information system, to serve their need, whether it is for business, learning or amusement. At the same time all these information systems profoundly affects the manner we do business, communicate, perform our duties and use information. A scary side-effect is that these systems also to a certain degree control which information we may find, alas; which we do not.

Legal practitioners also make use of information systems as part of their daily routines. Today, law firms and legal practises of all sizes at the minimum enrol a client journaling system. Many subscribe to online resources and databases such as Lovdata, a Norwegian legal information retrieval system that contain several legal databases, from legislature to legal literature. A few law firms even have more specialised systems such as knowledge bases and standard document collections.

1.4.1 Computers & Law

There are several different studies and research fields within the area of Computers & Law, and two broad groups. The first group is that of producing and analysing laws and regulations pertaining to the regulations of computers and computer use. For instance, regulation on the telecommunications sector or on privacy protection matters. The other group is the one I address, the study on how to make use of computer and information systems with the purpose to aid and support legal practitioners in general. This area of research on Information Systems is called Legal Information Systems (LIS) and has several sub systems including Decision-Support Systems, Legal Information Retrieval Systems and Knowledge-Based Systems (KBS) (Valente, 1995).

1.4.2 Artificial Intelligence & Law

The study on Legal Knowledge-Based Systems also falls under the research area referred in legal literature as the field of Artificial Intelligence & Law. The Artificial Intelligence (AI) study's core goal is to make computers inhibit intelligent behaviour (Valente, 2005). For the moment the main focus is on providing computers with sets of logics and rules and base the intelligence on solve tasks using these.

The common denominator of these systems is that they make use of a knowledge representation layer in order to solve the problems the systems are designed for; typically storing an legal information in the form of legal texts and resources.

The knowledge representation layer of Knowledge-Based System is often hidden for the users of the Information System. It is often authored by experts in knowledge management, the knowledge domain, and the computer system. Thus it is very difficult to change and understand how this knowledge representation layer works, or even add to it for a layperson. Researchers

focus on how one can embed such a feature into the information system, without being intrusive and require a lot of effort from the authors and contributors.

1.4.3 Hyperstructures, Ontologies, and the Semantic Web

Legal texts contain references to other legal texts, laws and legislation. These relations between each other can be used to create hyperstructures describing the relations between information resources. Hyperstructure is a fairly new term within the realm of Legal Information Systems. It stems from hyperlinks, the traditional means of navigation between pages on the Web. Hyperstructures are very basic forms, and can be used to form the basis of ontologies.

During the last decade, ontologies have gained increased attention, especially in the fields of Computer Sciences, Knowledge Engineering and Artificial Intelligence. Ontologies are used to express declarative knowledge, and relationships between knowledge and information. With the introduction of the Semantic Web the focus on ontologies was renewed.

1.4.4 Legal Method

Now remember that the lawyers' first purpose is to resolve legal disputes and problems. A lawyer will analyze and reason on the facts of the problem, and apply these into a legal context. In this process the lawyer will use her own legal knowledge, some which she may recall the outline or basis of, some of which will be needed to refresh on, and some she may need to research thoroughly. If the problem is intricate, the lawyer will usually make use of Legal Method. Legal Method is the methodology and practice on how legal practitioners go about solving legal problems. A common flow of events is that the lawyer looks at the legislature and regulations to the problem, and then reads the preparatory work. They then check for decisions pertaining to the problem, and at last any other legal resource necessary to solve the problem. The lawyer obtain all of these resources in books and papers, by exchanging knowledge with other law practitioners who have had similar cases, or by searching online bases like Lovdata. Besides the information found through Legal Method, the lawyer will use experience, and at last a good bunch of creativity to solve the problem (Bing, 1982).

This manner of solving a legal problem includes many knowledge processes that can be assisted and supported by information systems. Deducing the legal problem from the facts is a knowledge process, retrieving the relevant legal resources is another process, knowing where to

look is another process. Common to all is that experience is a key to solve the problem in a timely fashion.

1.5 Readers Guide

- Chapter 1 is a brief introduction to this thesis. The goal is to set the context of this thesis, the research question and motivation for this thesis.

- Chapter 2 describes the research method and methodology which has been used. It also covers the information system design and development principles I have undertaken.

- Chapter 3 focuses on central theory and definitions for (legal) information systems, Wikis and the data vs. information vs. knowledge paradigm.

- Chapter 4 focus on knowledge representation, classification schemas and ontology.

- Chapter 5 is about the Semantic Web vision and discuss technologies such as HTML, XML, W3C's Semantic Web stack and Topic Maps.

- Chapter 6 is about Juristopia, and covers user concerns, architecture and notes on implementation of the system. Including features of the system and the reasoning behind these. It goes into small detail on the central decisions and frameworks used for developing in a bird's eye perspective. This chapter has a strong technical focus on programming and includes some code fragments.

- Chapter 7 is the final chapter and includes discussion and conclusion. It also concludes on the main discoveries and problems, and suggests future improvements.

2. Method

2.1 Research Approach

This thesis is a product of documenting the reasons and decisions taken during designing and constructing a LIS. It also covers the technology covered during this process. I have not undertaken a quantitative exploration of other systems and their features; however my decisions and discussions are supported by observations and interactions with such systems.

My decisions are furthermore supported by qualitative research in the form of interviews with Knowledge Managers in three of Norway's largest Law Firms, during these long and open interviews, I have also done some observations on and interactions with their current systems. A large portion of document analysis has also been a big part of the information gathering in this thesis.

I have used a variation of Action Research, in the form that my own agenda to produce the prototype LIS, Juristopia, has been the primary motivation behind writing this thesis (Dick, 1993; Dick, 1997).

2.2 Project Management and Method

Formal requirement analyses are often undertaken prior to the other Software Development Life-Cycle[1] (SDLC) phases in projects concerning information systems of larger scale. "Big design up front" (BDUF) methods like the much criticised Waterfall-model[2], but also more modern and heavily used models like the iterative model[3], IBM Rational Unified Process[4], tend to implement

[1] Software Development is considered to have a Life-Cycle in that they all commonly contain separate paradigms, such as planning, implementation, testing, and so on.

[2] The Waterfall Model is an old and heavily discouraged development model. Tasks like design, analysis, implementation and testing are separated, and all focus is set on one task at a time. The model is often attributed to Winston Royce's 1970 paper, which is a very ironic fact as he in this very paper criticised the model, and actually advocated another development model: the iterative model.

[3] An Iterative or Incremental development model is a model where the basic idea is that the project goes through cyclic stages including planning, design, implementation, and testing. First after several cycles leads to a deliverable.

[4] RUP is a framework for an Iterative development model, where projects select the parts of the framework needed in a given project.

full analysis and design phases. A large study and design up front often leads to more problems once the implementation phase of the project begins is a learning that was concluded back in 1994 by the Standish Group in their CHAOS report[5], but the industry has continued these erroneous practises.

I have approached the problem in a more modern and flexible manner using agile methods for both project management and development. In this project I have used elements from agile methods like Scrum, Lean and Extreme Programming (XP). Most of the agile methods work best for smaller teams, with 4 to 10 members, so by being alone I have not been able to reap the benefits to a full extent. A common feature between the different types of agile methods is that the initial requirements and design of the system is "lighter", and that in depth analysis and requirements gathering is more integrated and incorporated with the development process. Agile Methods are usually very business oriented, and employ heavy customer interaction with the developers. As there is no customer in this thesis, I have modified the methods and skipped this focus.

2.2.1 Solo Scrum

Scrum is a very general lightweight project management methodology. Scrum is a set of practises that can support most development models, and is especially suited for agile models (Schwaber and Beedle, 2001). The essence of Scrum is that it allows for control over all issues that take part of a project, mainly by enforcing a strong focus on dialog between the project team members. Being alone on this project, the dialog aspect is obviously not as relevant. Scrum is a flexible development model, and takes account of project teams at the size of one member, referred to as Solo Scrum (Kniberg, 2007). By using Solo Scrum, I have adapted some central features of Scrum into my development and research method.

[5] A limited version of the report is available at http://www.standishgroup.com/sample_research/chaos_1994_1.php

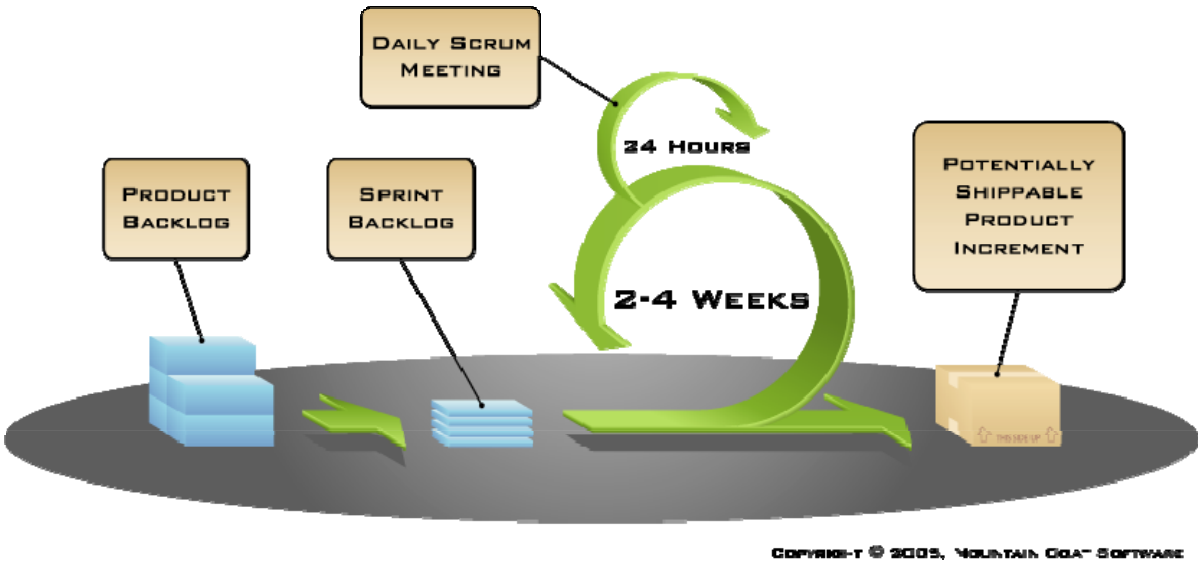


Figure 2-1 The Scrum Process

The essence and definitions of the Scrum method I use is as follows:

1. *Stories*, sometimes called *user stories*, are descriptions of the use cases of a system. These stories are broken down into *features* required in the delivered product. It is common to use a *storyboard* to collect the ideas and features. A storyboard is simply a drawing, a sketch, a board with post-its or similar containing of all the tasks and features needed in the implementation of a deliverable.

2. The *product backlog* is the history - or a set of all features, changes and stories implemented during the entire project's lifetime.

3. A *sprint* or *sprint loop* is a single iteration in the system development. Each sprint lasts for a given period, from one day to several weeks. In regular Scrum practices, these sprints last between two to four weeks. In Solo Scrum the sprints are kept shorter than in larger projects.

4. Prior to each sprint, a sprint planning is initiated. In this process the project leader, or *scrum master*, together with the team decides which features are to be delivered and implemented during the sprint. Also in Solo Scrum it is important to decide which features are to be focused with in the coming sprint. These features are during the sprint stored in the *sprint backlog*.

5. During the sprint, each day contains a brief meeting, called a *scrum*. The method states that these meetings are to be short and concise, for instance held in the walk to- and

during lunch. The purpose is that team members are to keep each other updated in the progress, the current problems and focus through dialog. In Solo Scrum these briefings can be replaced by for instance writing down a few lines about the status on a note, or to annotate your source code commits[6] in a well-formed and descriptive manner. At the next session, start by revising these notes.

6. At the completion phase of a sprint, a *sprint review* and a *sprint retrospective* is held. The purpose is to update the product backlog, and to sum up a review the entire sprint backlog.

7. The sprint leads to a *product increment* or *deliverable*. A product increment is usually a deliverable, a functional version of the final product/system. The increment is usually a working system with all the features from the product backlog implemented. This deliverable is to be reviewed by the *product owner* (the customer) and the team in case some new features or in the case some expectations were not met.

2.2.2 Lean Philosophy

I use Lean more as a philosophy than a development method, as the methods in Lean builds on and refine elements of the other agile development framework I use, XP. The core of the Lean philosophy is that a "lean deliverable" only includes features that provide business value to the customer. Furthermore, the features that give most business value are those that should be implemented first (Poppendieck and Poppendieck, 2003).

In other words, the developer should not deviate from the user stories developed together with the customer. They should avoid implementing "cool" features unless the customer has explicitly asked for these. One main reason behind this thinking is connected to the return of investment for the customer. Hidden or un-needed features may create negative side-effects like more tests, more possibilities for failure, higher maintenance, more documentation, all in all developers spend (waste) time and bill for features the customer did not ask for and has not agreed paying for.

[6] See 2.3.1 Revision Control

2.2.3 Extreme Programming

The third agile methods framework I make use of in this thesis is Extreme Programming, or XP. XP is an agile development framework, made from Kent Beck's personal experiences as a project manager, and first explained in his book, Extreme Programming Explained in 1999. From then as it has been adopted by more and more, it has evolved as a methodology incorporating new ideas and refining original ideas. I use XP as the development method within each Scrum sprint. XP includes numerous practises, and you select the ones that fit your needs rather than employ the entire stack. In this project I have selected the following practises (or principles) from XP:

1. KISS is an acronym for Keep it simple stupid. It is one of XP's main principles and applies to all aspects of the development including design, implementation and testing.

2. Standards and best practises for code, tests and (code design) patterns are important to ensure readability, and making the code revisable when I in the future need to understand how some code or test is supposed to work.

3. *Behaviour Driven Development*[7] (BDD) is an extension to *Test Driven Development* (TDD) in XP. The foundation of BDD and TDD is that *behaviour* or *testing* is the first to be produced, and should be used to control the implementation phase. BDD calls these product *specifications* (on behaviour), while TDD calls them *tests* (of code). None of the code written is valid unless it has *coverage*, meaning there is no un-specified or un-tested behaviour in the code. BDD/TDD is more a design tool than a testing and development tool in that it forces developers to heavily think about the feature and problems it is set to solve, prior to implementing the behaviour of the feature.

4. Continuous Integration (CI) means that the code you write during a sprint is integrated into the "production environment", the final product, continuously as you refactor the deliverable. I use an automatic CI system as part of my development framework so that I can at all times validate that a feature works and code is correct as I commit it to my revision control system.

5. *Small Iterations*, each iteration developed in XP should be as small as possible, and do as little as possible change to the product. This keeps the code easier to test and is easier to

[7] Dave Astels is often cited the person who coined the BDD term in his blog-post available at: http://blog.daveastels.com/2005/07/05/a-new-look-at-test-driven-development

spot errors and causes immediately as they are tested. This practise is combined with BDD/TDD by running tests and CI with each iteration of the code base.

6. *Continuous Re-factoring* of the design, the specifications and the code. While KISS and BDD/TDD together advocate a simple strategy for solving your tasks, you may not always get there at the first attempt, or the code may be of dubious quality. XP coins the term refactoring as the practise of revisiting code, and rewriting it.

2.3 Software Tools

2.3.1 Web Application Framework

I have used the Ruby on Rails[8] (RoR) framework as the main web application framework. Web application frameworks are code that integrates typical database driven web applications, and simplifies the creation of server side computing relating to web applications and services. RoR is based on the Model View Controller[9] (MVC) pattern and was developed by Daniel Heinemeier Hansson and released to the public in 2004, which has gained an ever growing installed base, and is now among the most popular web application development frameworks. RoR is developed in-, and use the Ruby[10] programming language, a high-level interpreted scripting language, and makes use of the Convention over Configuration[11] principle which enables very rapid development of web applications.

The Ruby programming language has an excellent library support in RubyForge[12] RubyGems[13], which are packages of programs, plug-ins and libraries that can be used by RoR. I use a number of such gem packages, more on RoR and this in the Juristopia Implementation Chapter.

[8] http://www.rubyonrails.org/

[9] Model View Controller is a software architectural pattern first introduced by Trygve Reenskaug in 1979. MVC is often used in user interface programming, as it creates a simple to read and maintain pluggable interface by separating backend (model), from the business logics layer (controller) and the users screen (view).

[10] http://www.ruby-lang.org

[11] By default the software will configure itself for the most “logic” tasks. Code in the system will for instance map a model User to the database Users, and provide magic methods and hooks for common tasks in web development.

[12] http://www.rubyforge.org/

[13] http://www.rubygems.org/

2.3.2 Integrated Development Environment (IDE)

TextMate[14] for Mac OSX provides a very simple, but excellent IDE for developing RoR applications. TextMate is an advanced text editor that provides syntax highlighting, and bundles for Ruby and RoR development. The bundles contain macros, and enable a developer to code very quickly, and at the same time provide possibilities to run and debug fragments of code in real-time.

2.3.3 Software Modelling

I use OmniGraffle[15] for Mac OSX to draw figures and UML[16] models. It is a very simple to use all around tool that creates clean and good-looking models.

2.3.4 Revision Control

I have used the open source software system, Subversion[17] (SVN) for source code revision control[18]. Subversion is a two-tier system, where you have a centralized code repository server. This enables developers to connect from any location, and update code. Revision control means that all changes to code are stored, so one can revert to any previous edition of the code.

The process of saving data to the code repository is called a “commit”. With each commit the developer can add a comment or annotation describing what changes or additions were included in the commit.

[14] http://macromates.com/

[15] http://www.omnigroup.com/applications/omnigraffle/

[16] Unified Modelling Language is a standardised specification language for object modelling, and is used to create abstract models for software systems.

[17] http://subversion.tigris.org/

[18] Revision control, version control, software configuration management (SCM), and source-code management are synonymous terms.

Part II

Theory, Definitions and Historical background

3. Theory and Definitions

This chapter is an exploration on the surrounding and central topics within legal information systems and web application development. It is meant to provide a reader with an overview on the concepts that are relevant in the scope of this thesis.

3.1 WikiWikiWeb

WikiWikiWeb or in the short form: Wiki [wi: kee:] is a very simple form of information system first created by Ward Cunningham when he in the 1995 decided to create a new, really simple and quick CMS[19] (Content Management System) for his website[20]. He called the new system the WikiWikiWeb, after the "Wiki Wiki" airport express shuttle buses at Honolulu (Cunningham, 2003). Wiki is Hawaiian, and means *quick* or *swift*.

Cunningham's WikiWikiWeb was inspired[21] by Apple's HyperCard system which was a simple and scriptable system that held information and notes in simple "cards", and allowed people to create cards containing text, music, photos and even small videos, and link them to another through simple links. HyperCard was in turn is quoted by its inventor, Bill Atkinson to be inspired by Vannevar Bush's memex system[22] (see 5.2).

Since the early conception of Wikis they have evolved to power huge information bases such as Wikipedia, which is the most commonly known and successful Wiki system of today. As of October 14th 2007 Wikipedia contained over 2,047,000 articles (Metawiki, 2007).

The definition of Wikis has thus evolved from being very simple systems for creating textual pages using hyperlinks to navigate - to complex systems supporting different domains of information, metadata, relational data, categorisation to mention a few.

[19] In this thesis I will not use the term CMS, as they are a subtype of Information Systems focused on providing access to information, and in general the same principles apply.

[20] http://c2.com/cgi/wiki?FrontPage

[21] http://c2.com/cgi/wiki?WikiWikiHyperCard

[22] http://c2.com/cgi/wiki?HyperCard

3.1.1 Wiki Principles

Common to all wiki systems is that they provide easy access to loosely structured information. Wiki is a constantly evolving system, and there is no single formal definition of which features and principles are required to be implemented to call a system a wiki.

Ward Cunningham has collected some of the principles used in different wiki implementations and created a general list of features common in wikis[23], I have revised this list to summarise typical modern usage of wikis.

Social or Collaborative system

Another central feature of Wikis is that it is based on not enforcing any unified formal text (design) pattern. Each page may vary vastly in style depending on the writer, but recommendations or guidelines for how to format texts are common. Wikis are most successful as a form of a social or collaborative system, where people together author the content (also called social or participatory writing). Where there is a collective incentive to create good information and more authors participate, the more successful a Wiki system is.

Simple Access

Another core principle of Wikis is that access to editing pages is simple. Early wikis allowed content to be editable by anyone, in such that anonymous visitors were allowed to edit pages just like registered users, and there were no restrictions as to who were allowed to edit information. With the increase of popularity it turned evident that simple access control was needed for most wiki portals. Wikis are now used for tasks like product documentation besides encyclopaedic use like Wikipedia, and thus the principle of anyone editing is not as relevant and applicable as its earliest forms.

Simple Links Between Content

Wikis commonly employ *simple* or *automatic hyperlinks* between wiki pages through the use of WikiWords (or CamelCase notation). WikiWords are concatenated words that use uppercase to distinguish the different words. WikiWords have special meaning in wikis in that they refer to another page or concept. Most Wikis use WikiWords as *one* means to create links between the pages, but also allow authors to manually create links with simple syntax. For instance

[23] http://c2.com/cgi/wiki?WikiPrinciples

"TargetPage" would automatically link to the page named TargetPage while [[TargetPage]] or even a form to describe the link with another text: [[TargetPage:"Click on this link to see the Target Page"]] is allowed. On Wikipedia it is common to use the latter two forms for creating hyperlinks, mainly because Wikipedia does not only use WikiWords to form Wiki pages, but also more descriptive page names like "Target_page_(scope)" are used to separate scope to certain words are allowed.

Simple Syntax

Easy and simple syntax for editing is central to the success of Wikis. Authors should not focus on learning and knowing gazillions of mark-up codes in order to describe information - which is the reason Wikis exist in the first place. The lower the threshold to create and update information is, the more likely it is to get users participating in creating the content. In Cunningham's first wiki implementation, he excessively used syntax he called "double syntax". Double syntax was based on, well, doubles. For instance did two newlines translate into a new paragraph, concatenated words with uppercase (WikiWords) turned into links,

Version Management

Version management is important in Wikis. Sometimes in the case that authors disagree with another or a Wiki page is attacked by trolls (people deliberately writing false or inaccurate information) the possibility to revert these changes is important in maintaining a consistent and truthful information source. Sometimes authors and contributors with to see the difference between updates of a page to identify what has been changed, and Wikis typically include support for displaying the difference between versions. Version Management is important for another aspect too, which is consistency in information relationships or associations: sometimes as information is edited other information objects may contain links to information that is being changed, and as such version control can be used to preserve the integrity of such links.

Searching and Navigation

Wikis normally employ features like back-links or list-pages that list all other pages that link to a particular page. This makes creation of category and collections of pages simple and ease the effort needed to update other pages when changing information in a page. Wikis also contain special pages that list all pages in the system, list recently edited or new pages and pages that list the pages that do not contain links to other pages (orphans). Simple search algorithms to find pages that contain search words in the page name and simple search query support is common.

Some Wikis have even attempted creating special navigation pages that visualise the hyperstructures between pages using simple nodes and directed arrows.

3.1.2 Wiki Criticism

Wikis have received criticism for many reasons. The main reasons of criticism are in fact the same reasons that make wikis such a successful technology. In respect to the most cited, biggest and most active online wiki of today, Wikipedia, with over 33,000 editors, there exist some more profound aspects of concern.

One of the first problems evident on wikis in general were people maliciously spamming or deliberately abusing their editorial rights by adding wrongful or highly biased information, also called the act of *trolling*. If such people (or spam bots), *trolls*, are allowed to edit pages they may be disruptive to the quality of the information (Sanger, 2004). Early spam bots were handled by adding "Turing-tests", tests that asserted that the editor was a human and could interpret a question or an image analysis, such as a CAPTCHA-test (Completely Automated Public Turing test to tell Computers and Humans Apart), which were popularised through the use of obfuscated images with text bits by (now) professor Luis von Ahn[24] at Carnegie Mellon University.

Another main reason Wikipedia is criticised as a less-credible source is that there is no editorial revision or formal page review of the available information. Anyone can in fact edit or review pages as they please, and many of the editors are non-specialists in the field they write about. Co-founder of Wikipedia, Larry Sanger, criticised Wikipedia for being *non-elitist*, and even bordering to *anti-elitist*. Elitist in this context means specialists, that authors are professionals or academia and have above average knowledge on a topic they write about (Sanger, 2004). When Sanger uses the very strong wording of anti-elitist he uses the example that when he was working with Wikipedia, he tried to introduce a policy that editors had to defer to expert editors' opinions. This was hard to enforce, as co-founder, and Wikipedia boss, Jimmy Wales was decidedly against such a policy. In this concern, Sanger also brings up the fact that the users of wikis call it unreasoned censorship when such policies are enacted in the wiki world. Jimmy Wales is an avid advocate of self-regulation over review[25]. G.E. Gorman goes even one step further than Sanger and criticise wikis for being realms of small kings: "*'head cases' both within and outside of academe*", "*vehicles for one-upmanship among competing academics"*, and

[24] http://www.cs.cmu.edu/~biglou/

[25] http://en.wikipedia.org/wiki/User_talk:Jimbo_Wales/Archive_27

forums where "*unformed and juvenile views are aired"* when the wiki "collaborators" battle for their own views (Gorman, 2005). Both Sanger and Gorman state that many of the most active wiki community members exert elements of the anti-elitist stance.

Lack of diversity and overly biased views are also common on wikis like Wikipedia, whether this is due to trolling, one-upmanship or non-specialist knowledge. American actor and comedian coined *truthiness* to describe this kind of practise. Truthiness was awarded Word of the year for both 2005 (American Dialect Society) and 2006 (Miriam-Webster dictionary) (Meyer, 2006). Truthiness essentially is a satirical word to describe that "the truth is what you agree it is". Several pages on wikis are apparent truthiness or heavily biased, this ranges from hidden marketing for products, and even more serious skewing of information for political purposes. The latter evident by much reported incident where staff members of U.S. Senator Norm Coleman edited his presentation in favour of the senator. This sprung into a broader investigation where it was found that more that 1,000 edits were performed by staff members from the U.S. House of Representatives and the U.S. Senate (Anderson, 2006). Not all of these edits were illegitimate, but as an objective source of information, this has damaged Wikipedia's reputation as a proponent of Information Democracy.

The above reasons are foremost in why wikis do not hold a great reputation with experts, librarians and academia. A few U.S. Colleges have even banned Wikipedia as a citation source, including prominent UCLA (University of California at Los Angeles) (Chen, 2007). It is in other words difficult to determine the quality of information available on a wiki such as Wikipedia. Not necessarily because the information in general is of poor quality, but rather the opposite - the available information is pretty good and accurate. The inaccuracies are first shown when digging deeper on the detail level. Information may be skewed or imbalanced by the lack of editorial review, and this is tightly coupled with the "anyone can edit" principle. This is argued to lull a reader into a false perception, where the reader assumes and expects fairly accurate information for any given topic, while that is not the case.

Larry Sanger quit Wikipedia in 2002, and has since launched a new wiki, citizendum.org, which tries to solve some of the weaknesses of Wikipedia. The near anarchistic principle of allowing anyone to edit and collaborate on pages, even anonymously have been revised, and all users must now register with their full name to be able to edit. Furthermore, the idea behind Citizendum is still that user collaboration will still be the main source of information, but the editors will censor, neutralise and fact-check information through editorial review of all pages, performed by experts.

3.2 Information Systems Definitions

As mentioned in the introduction, this thesis is about a Legal Information System (LIS) subsystem, a Knowledge Based System (KBS). There is a need to define what constitutes this particular kind of LIS. The definition of the central terms *data*, *knowledge* and *information* have meaning both in the theory and appliance of information systems as well as in the philosophical sphere. The terms overlap each other, and are often used among another, so I need to start off defining these terms in the context.

What is legal information? It's simple to answer: All information that a lawyer or legal practitioner use to produce legal advice. Some examples include laws, statutes, regulations, preparatory work, legal commentary, decisions, agreements, standard agreements, notes on legal procedures and legal theory.

At the same time this is a very complex type of information, with many relations to another, cross-references, relations to legal principles, and both domain specific and context specific terms. Legal information usually contains several inter-dependencies – is information built on other information. Legal information is in never *stable* information; it is always changing, and is constantly evolving with society. Legal practitioners and scholars may be in need for "the current legal information for a specific problem", which may not be the same as the "legal information a year ago for a specific problem".

LIS systems typically refer to the same principles as other information systems, where information is tightly connected with the scenario, such as in Medical information systems (MIS). LIS research is however maybe most connected to current research on Library Information Systems, and Information Management. The whole area including both Information and Knowledge; Systems and Management builds on each other and may be difficult to separate. In this thesis I will not separate much between information systems and knowledge based systems.

3.2.1 Data

Data is the basic building blocks of a digital representation. Data is formed from raw data into collections of symbols, characters and numbers. According to Luciano Floridi's proposed *General Definition of Information* (GDI) (Floridi, 2005b) data by itself has no meaning or

context. Data is simple building blocks constructed of bits and bytes that can be stored, retrieved or processed.

According to Floridi's theory, there are four main characteristics, or types, of data (Floridi, 2005a): Primary data, Metadata, Operational data and Derivative data.

Primary data

Is the data we normally refer to as data; typically this is the data that forms information - the core data of an information system. Primary data is typically stored in databases or files in an information system. Instances of primary data are commonly persisted (stored) data structures, formatted texts and documents, etc.

Metadata

Metadata is a form of data that describes other data with the purpose of providing semantic content. In a database centric view common metadata can be fields telling when primary data content was created, updated, by whom, etc.

Operational data

These are data that describe the usage of data within a system, instances of operational data are found in database transaction logs, web controller logs, web access logs, etc. Operational data is usually generated by the information system to tell how the system is used, identifying faults, security and general performance (benchmarking).

Derivative data

These are data that can be processed from the other three forms of data, for example in the process of data mining, auditing, search optimization and other ideometric analyses (usage, click-flows, user interface optimisation, etc).

Data processing

Data processing consists of retrieving (primary) data or streams of data, feed it through an algorithm and produce output. The output can be new data, or it can be data with a context and meaning - semantic content.

The difference between data and semantic content can be illustrated by an algorithm summing up the first 10 Fibonacci numbers (1, 1, 2, 3, 5, 8, 13, 21, 34 and 55).

If the algorithm outputs the number: 143 no context or meaning has been added and the output is simply new data. On the other hand, if the algorithm outputs "The sum of the first 10 Fibonacci numbers is 143" the algorithm has added semantic content to the data.

3.2.2 Information

The difference between data and information is often referred to as a merely philosophical topic, as there is no clear difference between data and information to a user of a system.

Dictionaries typically define information in terms of communication, reception and understanding of knowledge, intelligence, facts, news, advice or data[26]. According to Floridi's *General Definition of Information* (GDI) (Floridi, 2005b) in regards to data information is:

> *"σ is an instance of information, understood as DOS, if and only if:*
>
> *1. σ consists of n data (d), for n ≥ 1;*
>
> *2. the data in σ are well-formed (wfd);*
>
> *3. the data in σ are meaningful (mwfd = σ)*

DOS (declarative, objective and semantic) is what constitutes semantic content. In other words, information is semantic content which is built of data (and metadata) and has meaning.

Information Object

An *information object* is any *medium* or *thing* that holds information such as documents, pages, images, pod casts, video clips and so on. I will also refer to information objects as simply *objects*.

According to Ted Nelson, *"Information as a commodity is a myth. Information always comes in packages"* (Nelson, 1999) information is always presented in an information object, so when I refer to information, I imply its information (or knowledge) learned or attained from an information object.

[26] http://www.dict.org/bin/Dict

Resource

I use the term *resource* as the further abstraction of information objects: a resource is any information object.

Metainformation

Metainformation is information about the nature or functions of information. Metainformation is not widely discussed in research, but plays an essential part in knowledge-based systems in that *knowledge rules* are metainformation - they declare the nature or function of other information.

Semantic Information

According to Floridi, another subtype of information is semantic information. This is also a philosophical topic; what is the difference between information and semantic information?

There is no consensus on what constitutes semantic information. Some tried to define a Standard Definition for semantic Information (SDI), based on the GDI. Simply by using the concept of DOS *for information*, the outcome was the initial SDI. The author of the GDI, Luciano Floridi, contested this definition of semantic information very profusely, and stated that it has a major flaw: what separates semantic information from regular information, is not only that it is *meaningful*, but also that it is *truthful* (Floridi, 2005a). This paper describes nine reasons the truth aspect is needed. He thus revised his own GDI (see previous page), and added a fourth rule, to form his own (proper) SDI for semantic information:

> *4. the σ are truthful."*

Truth is indeed a very important factor in describing semantic information, especially in a knowledge-based (semantic) system. Unless a truth aspect is added, false or misleading information may also be defined semantic information. Information builds on other information, and if false information is allowed into the system, it may in turn corrupt the other information per se. More on this is discussed under Wikis in the next chapter.

3.2.3 Knowledge

Knowledge is an elusive term to define. Knowledge is typically defined in dictionaries as both an element of information, a scope of information, as well as the cognitive acquaintance, interpretation, enlightenment, or appliance of information. In information systems theory, knowledge differs from information and data in that new knowledge may be inferred from

existing knowledge. Some argue that knowledge is information with semantic truth, or validated "*justifiable* or *explainable semantic information"*. In other words, if information is data with meaning, then knowledge is the cognitive acknowledgement, or the interpretation of processed semantic information (with implied truth). In computer science and in particular in the field of AI the tradition has been that knowledge is defined in a functional manner in order that computers can process it. The

Semantic Information + Interpretation ➔ Knowledge

Interpretation is typically described by well-defined *axioms*, or *knowledge rules*, in other words:

Semantic Information + Axioms ➔ Knowledge

A central concept with knowledge is that *new* knowledge can be *inferred* from information and *existing* knowledge meaning that one can process information through a set of axioms, and thus produce new knowledge, or interpretations:

Semantic Information + Existing Knowledge ➔ New Knowledge; or

Semantic Information + Existing Axioms ➔ New Axioms

Knowledge inference in computer centric context is an axiomatic process, based on well described knowledge rules, or axioms. Axioms are simple rules that have a logic value that are used in describing semantics in information.

3.2.4 Logics for Information Systems

Research on logics has been popular in field of AI & Law in the last decade (Valente, 1995). Knowledge Based Systems are often *axiomatic systems*: systems that use *axioms*, or well-defined (logical) rules to describe knowledge. Much knowledge representation research is based on using *descriptive logics* (*first-order predicate logic*, *modal logic* or *higher-order logic* on sets of *axioms*). As I've earlier pointed out, new knowledge can be extracted from semantic information together with existing knowledge. To find and derive these new axioms one typically uses *inference* or *deduction* techniques. It is difficult to separate the meaning of inference and deduction without going into a more in-depth introduction on logics. This thesis

will not document the principles and quirks of different forms of logics, but I feel the need to describe a few basic terms and methods before going onto ontology and semantic web themes.

Axioms and knowledge-rules in information science

An (non-logic) *axiom* is a short sentence or statement that describes a single rule that is assumed to be true. In information systems context of Semantic Web and Ontology an axiom is also a well-defined fact or assertion about a *thing*. Axioms are considered formally stated facts and assertions that together with other axioms form well-defined *rules*. Examples of simple axioms:

1. 'A' is a 'Legal Text'.
2. 'B' is a 'Legal Text'.
3. 'B' is type 'Law'.
4. 'A' is same type as 'B'.
5. 'A' is not the same type as 'C'.
6. 'C' is a 'Legal Text'.

Axioms are considered true by default, so if an erroneous or incorrect axiom is specified, it may "corrupt" the (natural) logic in that it is always treated as a truth, and produce false results. A lot of research is focused on reasoning to deduce false axioms.

Deductive reasoning

Deduction or deductive reasoning is in the traditional perspective (Greek Classic philosophers like Aristotle, Thales and Pythagoras) the "art" or applying *general facts and principles* to reach *conclusions* to more specific facts and principles. A classical example of deduction is[27]:

1. All apples are fruit.
2. All fruits grow on trees.
3. Therefore all apples must grow on trees.

[27] http://en.wikipedia.org/wiki/Deductive_reasoning

Inference

Inference is to derive conclusions solely from already known facts. One of the definitions of knowledge is that new knowledge can be *inferred* from information plus existing knowledge. Inference is much like deduction, but it is applicable not only to general facts and principles, but for any observations, data, or axioms. One use deductive reasoning when *inferring*, an example of inference on the above example is:

1. All apples are red (observed).

2. Therefore all apples are red and grow on trees.

4) is not a general *principle*, but a knowledge-rule based on all the observations of apples in this system, which were all red. Even though 4) is not *truthful*, inference based on it is allowed and valid and 5) is a valid new rule based on the existing data. This is a general problem in knowledge-based systems with limited data and false assertions (axioms).

Inductive reasoning

On the other hand *induction* or *inductive reasoning* is to *induce* or *derive* general principle(s) *from* specific facts. Induction is not specifically used in this thesis, but elements of inference are used in inductive reasoning. When Isaac Newton observed the specific cases of apples and other things falling to earth – he *induced* the general theory of gravitation.

Contstraints Logic Programming

Research on logics became part of the field of AI in the late 1970s. During the 1980s the Prolog programming language became popular within the field of AI, and along with that spawned a focus on constraint (logic) programming languages (Park and Hunting, 2003). Whereas logic programming is based on inferring and deducing new knowledge, constraint logic programming is based on limiting the number of hits or reducing based on the knowledge the system contains. Constraint logics programming also uses inference and deduction to form new axioms that are used in the reductions. Logic programming makes use of different forms of logic to deduce and constraint results by using axioms:

Descriptive Logic

Descriptive Logic (DL) is a rather broad set comprised of different sets of logics. In AI and IS this form of logic has gained a lot of attention since the discovery of modalities in graph-based

semantics in the early 1960s. Current Descriptive Logic research is focused on how modal logics can be used to model semantic relationships between knowledge, and uses elements from classic logics theory like Quantification (comparisons), First Order Predicate Logic (for describing resources), Modal Logic with Kripke Semantics (for describing relationships and structure between all resources) and Higher-Order Logic or Type Theory (for describing inheritance and equalities between resources).

Modal logic

Modal logic is based on inferring (verbal) modalities, for some set of relevant axioms. Modal verbs such as 'can', 'would' and 'might' are used in forming outcomes that can be seen as necessities, possibilities and contingent conditions. This means that by using modal logic one can also compute new axioms *for certain* contexts *from certain* rules, and these axioms need not be valid in all contexts. Axioms can thus be inferred validly for certain contexts, but the same axioms are false in other segments (Blackburn et al., 2004).

Kripke semantics are the basis of Descriptive Logic.

Modal logic separates the inferred results into three main modalities:

Necessity – it is not necessarily false (thus must necessarily be *true or false*).

Possibility – it is *not possibly false* (thus it is possibly true).

Contingent – it is not necessarily false but not necessarily true either.

For example if the system contains three axioms that state:

1. "A is a Law text"
2. "Law texts are Formal Laws"
3. "B is the same as A"

With applying modal logic the system can infer that "B is possibly a Law text" as well as the contingency if B is law: "B is necessarily a Formal Law" and so on.

Another subset of modal logic is Kripke Semantics, named from its discoverer, Saul Kripke. Kripke Semantics are based on modal quantifications from tuples (Kripke frames) or triples (Kripke models). Kripke's modal logic is the basis of Descriptive Logic languages like OWL,

Web Ontology Language and its predecessor OIL (Ontology Inference Language). In Kripke semantics *satisfactions* are used to determine the validity of rules. Then by using simple Kripke semantics could say in regards to the above example (in respect to the axioms) that "1) is satisfied by 3)", "2) forces 1)" and "3 satisfies 1".

First-order Logic

First-order (Predicate) Logic (FOL) is often used as the basis to form both simple axioms and *suggest* possible outcomes for individual knowledge objects. FOL is in general used to describe single information objects, and even though quantification can be used in order to compare knowledge with other knowledge, it is generally only used for generic cases in Descriptive Logic. For instance if "A is a Law text" and "B is the same as A" then "B is a Law text" is a classic example of First-order logic. Resource Description Framework (RDF) uses FOL to define properties of a resource.

Higher-order logic

Higher-order Logic (HOL) is also used in respect to type theory and category grammar. In Description Logic elements from HOL are for instance used to specify relationships between types of information and knowledge. For instance "Formal Laws are Law texts" and "Law texts are Legal texts" then one can infer that "Formal Laws are Legal texts". RDF uses HOL to define inheritance and type-specific properties a resource can have as a consequence of inheritance.

3.3 (Legal) Information System Challenges

The main problem with information is that there is too much of it. A big challenge in information science today is to make information manoeuvrable (accessible, searchable and findable); to enable humans finding the *right* information.

3.3.1 Information Overload

Technostress (Rosen and Weil, 1997), *data smog* (Shenk, 1998) and *information pollution* (Nielsen, 2003) are other definitions of the modern day phenomenon of information overload. The phenomenon is connected to the ever growing and vast quantities of available information. When too much information is available, *"findability*" and *manoeuvrability* are challenged. *Information quality* is also closely connected to the information overload problem.

3.3.2 Information Quality

It is hard to define what constitutes good quality of information. In the context of Semantic Web information should at the least be well-formatted, and well-defined. Quality is also determined by accessibility and findability; the *right* information is the information a reader sought for. Quality information is also (re-) usable, readable and accessible. Usability and presentation, accessibility and information plays an important role in deciding what good information quality is.

Information should be simple and consistent

Jakob Nielsen's "less is more" mantra is connected to his information pollution definition (Nielsen, 2003) states:

> *"Excessive word count and worthless details are making it harder for people to extract useful information. The more you say, the more people tune out your message."*

Nielsen' bases this mantra on extensive analysis of how people read websites. Contrary to reading books reading from computer screens is tiring, and pages are often competing for attention with several hundreds or thousand of other pages. This leads to web readers often scanning pages instead of reading them back to back like with books (Nielsen, 1997). The competition between pages is often rooted in that web readers are not always sure that the information they've found is the one they sought for. With more complicated information one have to read a lot more to be able to understand the information and only then be able to separate relevant information. This principle of less is more is important to of when authoring information and is therefore important to the information system to provide guidelines for how to author information, especially in user-driven content sites like Wikis. Nielsen provides six guideline principles that web sites should implement (Nielsen, 1997):

- *highlighted **keywords** (hypertext links serve as one form of highlighting; typeface variations and color are others)*
- *meaningful **sub-headings** (not "clever" ones)*
- *bulleted **lists***
- ***one idea** per paragraph (users will skip over any additional ideas if they are not caught by the first few words in the paragraph)*
- *the inverted pyramid style, starting with the conclusion*

- ***half the word count*** *(or less) than conventional writing*

Information should be well-presented

Nielsen also state that information quality is connected to the presentation of the information other than the words themselves. Inconsistent pages with different colour-schemes, text styles, document headings, navigation orders and layouts will require users to use more time on identifying elements of the presentation design and is considered less readable and accessible. Bad designs results in that visitors are left with the impression that the site is badly organised, and the information is often less trusted. Good design is design that makes information easier to read and knowledge easier to attain.

Information relationships should be exposed

Information connection is the most important factor for good quality of information on the web. Like in research, good research builds on other good research, and the clearer these bonds are, the more trustworthy the research will be. Digital information is no different. To keep information as short as possible, while at the same time *complete* is a daunting task. By focusing on making the little bits of information accessible (exposed) and associated (in respect to relationships) to other bits of information, the trustworthiness and readability of the information is strengthened.

Information relationships should be consistent and preserved

Another aspect of information relationships is that information systems are too often built and designed for a single purpose, such as to make a product catalogue available for web users through a browser. Often the uses of websites are not well thought through; and at later revisions pages may change addresses, products may appear, change or be deleted. Hard to read and non-descriptive URLs are reasons for broken links and erroneous links (links to wrong information). This proves it difficult to integrate and preserve information relationships with other web-services and related sites that may wish to use or link to this information.

Information system designers should take the future into account when designing access to information, and try their best to not alter the reverse relationships that may link to *their* information as a core principle of the systems design (see 5.2.0 No Dead Links).

3.3.3 Information Manoeuvrability

Managing information can be very difficult. Data comes in all kinds of formats, text, images, video, sound and other digital media are all part of our information intake. Many of these media exist as loosely unstructured data, while others are moderately structured. There is no such thing as unstructured information; all information comes with a certain structure, whether it comes through dialog, an essay or a Law text. Data and information can be stored in all the above media, accessible as files, or they can be embedded in (relational and object) databases, etc. Some sites have navigation trees and site maps; some rely solely on search engines while other sites combine these methods. A common factor is that to make the information available, a website needs to rely on some form of indexing, as indexes are the keys to find and manoeuvre the information. Manoeuvrability is thus very dependent on the quality of the information index, at the same time.

Information Indexing

The purpose of an information index is to find information in a system. Information indexing is the process to analyse information objects and create a registry of what information can be found where. They are used for searching, organisation, and navigation of information such as search engines, site maps and site navigation. Indexes are typically implemented as trees, nested sets or graph-based data structures that point indexed terms to information objects. Indexes can both hold complete copies of all available information objects or simpler structures that map certain terms to particular information objects. It is common to separate between two distinct forms of indexes: *subject-based indexes* and *keyword-based indexes*.

Subject-based Indexing

Subject-based indexing is a form of indexing using subjects relevant to the information object as the indexing words or phrases, and is characterised and indexed by subjects created and based from the *understanding* of an information object (Bing, 1984). As indexing algorithms are not advanced enough to understand information or knowledge, only particular elements of these, such indexing still requires a vast editorial effort by humans to achieve.

Keyword-based Indexing

Keyword-based (also: key-term-based or token-based) indexing is the most common type of indexing today, mainly because it is simpler to implement, understand, and can be used to achieve very good results almost automatically. This form of index is typically built by parsing

information objects, tokenizing the words and phrases found, and finally to create the *mappings* between these tokens (keywords) and the information objects in the index.

Uncontrolled vs. Controlled Indexing

Another important aspect of indexing is separating between *free or defined* indexes (Bing, 1984), also called *uncontrolled* vs. *controlled* indexing. An *uncontrolled* or *free index* is an index where keywords are not processed through defined rules or processes set to limit the scope of the word, but rather selected by the indexing algorithm solely from the tokens found in the information object. A *controlled* or *defined index* uses mechanisms to *control* which terms are used for indexing, for instance by replacing synonymous words with broader definitions such as through thesauri or ontology as I will later get back to.

Automatic Indexing

Automatic indexing is usually performed by running an indexing algorithm over information objects. The algorithm differ from solution to solution, but in modern indexers these algorithms typically work by tokenizing all the words and phrases, to apply processing rules to the tokens (defined indexes) and then to analyse the relations between tokens before inserting these entries into the index.

The set of control mechanisms for identifying and separating documents and document elements used by indexing algorithms vary much between implementations; some use elements of artificial intelligence (AI), they often use classification scheme-based lookups such as thesauri, taxonomies and ontology, as I will also later discuss more in depth (see chapter 4.2).

Search engines use “web-crawlers” or “web-spiders” that are automated processes that *crawl* through websites and index all information objects they find. Web search engines will crawl through any site they can find on the Internet, as long as the site hasn’t opted out from being indexed (with robots.txt[28] or by asking the search engine directly), where as website search engines usually process the information objects internally and not through the web.

Manual Indexing

Manual indexing is usually superior to automated indexing in quality, but does require an enormous effort in time and manpower as the quantities of information grow. Humans are still

[28] http://www.robotstxt.org/

able to understand information and create qualified subjects and indexes to information far superior to what computers are able to.

The process of creating an index can therefore be performed *manually*, *automatically*, and is often a combination of both (Sullivan, 2007). Combination of both automatic indexing (automated assistance) and manual indexing is currently considered the best option. With defined control schemes such as AI, NLP, thesauri and ontology, this assistance can be substantial in achieving subject-based indexes.

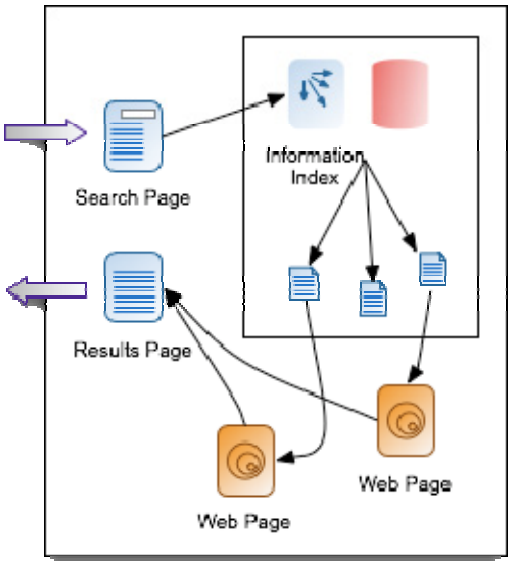
Text-based Searching

Text-based searching has been the basis for legal information retrieval for the last decade. Text-based searches normally use some form of Boolean logic to narrow down search queries (more in chapter 3.2.4). Boolean logic is a set of simple operands and functions that enable fast traversing and drill-down functionality especially on keyword-based indexes.

Search queries can use simple operands like "AND", "OR", "NOT", often abbreviated to '+', '?' and '–'. The queries consist of select keywords and phrases that should be found in the result. Internally in search engines these queries are translated to very fast machine code that is used to limit the hits found in the index (and report results). Boolean based searches are very fast, as computer can almost natively translate this type of search queries and pull directly from indexes (which are already optimised by design). This is also the biggest problem of Boolean-based searching – it only finds exactly what users ask for.

While being very simple to get up and running and produce good results, it was early stated that Boolean searching is not at all optimal for information systems retrieval (Verhoeff et al., 1961). This problem with searching is twofold:

1. The lookup algorithm only reports on exactly what users "ask" for.
2. The indexing algorithm only index the tokens and phrases found in a given information object.



The simple explanation to why Boolean logic is not an optimal solution is connected to 1) in

that users do not always know exactly what they are searching for, and 2) in that different authors does not use the same tokens or phrases.

There are many ways to improve the index in respect to 2) for instance by applying controlled vocabularies for indexing, filters and substitution lists and several other tricks such as AI, counting cross-references, and so on.

Figure 3-1 A Typical Web Search

Findability

Findability is a rather new term used in the Semantic Web context. It relates not only to information being findable but also that the *right* information being found. Whereas most systems try to optimise *searchability*, or methods to index all available information, the findability paradigm is focused on creating well-defined, and intelligent collections of high quality information, usually manually indexed and editorially processed information (into knowledge) thus making it accessible and understandable for computers.

Understanding Information

Two important elements in *understanding information* are:

1. Syntax: Grammar and Natural Language Processing (NLP) techniques such as occurrence and frequency of words, with occurrence NLP can infer that densities of for instance present tense verbs may indicate an introduction and so on. A computer can interpret information by checking the syntax through parsing the object and check syntactical rules both mark-up (XML, HTML, SGML) and other well-defined languages (such as C, Ruby, English, and Norwegian).

2. Semantics (meaning): semantic data and semantic information is however much more difficult for computers to understand. As I will later elaborate on; well-defined data, syntax and rules can assist in this process to understand elements of the information, but no computer system is yet “smart” (educated) enough to be able to understand the pure meaning of any given information object.

The main problem with (automatic) indexing is that computers do not *understand* the objects they are processing; the quality of the index is only as good as the indexing algorithm. A computer has a hard time separating a single word in different contexts, and even though advanced control mechanisms based on meta- data and information together with ontologies,

inference engines and NLP-techniques have created a noteworthy increase in the ability to perform automatic indexing, it is not yet feasible as subject-based indexing (see chapter 4.2).

Making computers understand information

In order to let the computer understand information, it must firstly be well-defined and formatted with an accessible and parseable language a computer can read. Syntactically this can be done by using a descriptive mark-up language such as XML. This is a form of making information accessible and parseable.

Enabling computers to understand the semantic meaning of these syntactically well-defined information objects is more advanced. Adding semantic information and data (metadata, knowledge rules and axioms, description logics and relations, create ontologies) are necessary to make computers understand information. The more they know, the more they can “understand”.

4. Knowledge Representation and Aquisition

Knowledge or knowledge representation layers is an abstract layer used to describe semantic information as well as semantic data into knowledge. Knowledge Acquisition is the process of retrieving knowledge from the system. At the same time this layer is often a conceptualization comprised by the knowledge about the resources of a system. Knowledge layers can be compared with subject-based indexes, as most technologies focus on describing the contexts and information for a specific domain. Knowledge representation layers are used to describe the information and relations between resources. Depending on the implementation and technologies used this layer can exist as annotations to an index, used by indexes, as annotations in form of meta-data or even as an external and entirely separate system. Regardless the implementation, there are a few central concepts that most knowledge technologies employ in knowledge based and information systems research.

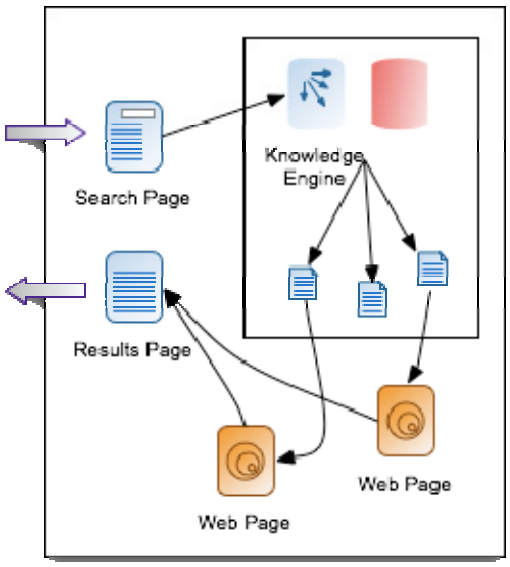


Figure 4-1 Knowledge Engine Query

Knowledge and Inference Engines

The knowledge layer is typically accessed through a knowledge or inference engine. When looking up and comparing with existing knowledge it is typically called a *knowledge engine.* There are many different types of knowledge engines, I won't go in breadth or depth about all of these, but two essentials are "RDF triple stores", and "Topic Map Engines". Both of these are essentially knowledge engines that take in data, and merge these with existing to create unison in the inputted information. These stores are by themselves indexes that can explain an entire domain, but there is no understanding in these.

When dealing with a system that should determine or *infer* if some action is valid or some bit of information is true based on knowledge in the knowledge layer it is commonly called an *inference engine*. I will in this thesis refer to both as knowledge engines - a service that responds to certain semantic query and responds with the proper semantic information.

In Information Systems science especially in the field of Knowledge-Based Systems *inference engines* are used to find or infer well-defined rules and concepts based on already known axioms and rules.

Semantic Searches

Searching and looking up information from the knowledge layer is called *semantic searching*. There exist many ways of performing semantic searches, and many models of doing so. Semantic searching makes use of a knowledge layer in some manner to structure and aid the search results (see Figure 4-1 Knowledge Engine Query).

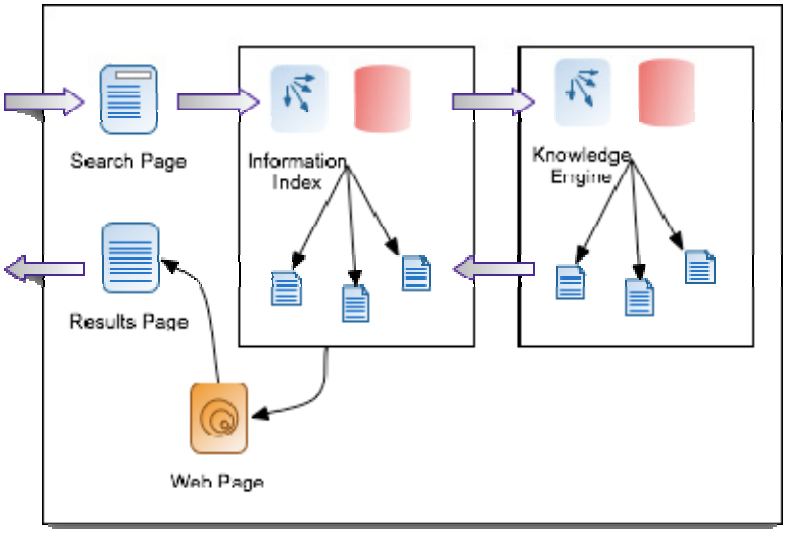


Figure 4-2 Hybrid Semantic Search System

Typically semantic searches provide options to reason with the knowledge layer: to be able to ask and narrow down which information is wished for with for instance drill-down on available subjects or concepts or specific scopes of information (for instance Norwegian texts and Computer Science related).

Semantic search implementations are often accomplished by hybrid systems using both knowledge engines and typical text-based search engines in regular information indexes. For instance by using a text-based searching algorithm that internally uses an inference or knowledge layer engine to “refine” the search based on the structured relationships from the information it found from the text-based search, or the other way around, where the query is first processed by the knowledge engine, and the (qualified) results are then used by the text-based search, or even both directions.

Semantic searches usually add functionality that enables users to limit or *filter* searches. If one define limits prior to the search, it is called *pre-filtering*, and if it's done after a search, it's called *post-filtering*. Other features common in semantic search results are *groupings* of similar or related results, and *ordering* options that allow one to refine the results ranking by specifying certain metadata or specific terms (subjects).

4.2 Categorising and Classification Schemes

Ever since the dawn of time humans have been categorising. Aristotle's ten *praedicamenta* in his work *Organon* were definitions on how to create an order for all *things* into *subjects* and *predicates*, and is probably the one model of categorisation that has survived the longest. In the medieval ages categorisation models were common and most built on Aristotle's model. As the 1700s brought forth systematisation, Swedish doctor of medicine Carl von Linné was central in the paradigm shift from categorisation to classification, in particular through his work, *Systema Naturae*, a work that classified over 4400 animals and 7700 plants[29]. How we categorise and classify things has since been a central topic within anthropology and especially the science of cognitive anthropology (Bowker and Star, 1999). Many new classification schemes have been conceived also in our century; some successful and others temporary. Two examples of revolutionary categorisation schemes are the library scheme of Melvil Dewey and Paul Otlet's knowledge organisation models (Bing, 2006).

4.2.1 Metadata and Simple Classification Schemes

Information and knowledge representation classifications have been a major research area the last decade. The patterns between information and relations between data and information can be formalised into subject-oriented classification schemes (Garshol, 2004).

Metadata is any data about data; data about information. Metadata form vocabularies Typical instances of metadata found in most applications include are creation, modification and access dates, information about authorship, ownership, document title, and so on. These metadata are pretty much "default" and "implicit", and have their practical use, mostly within applications. As the research focus on information grew it became apparent that metadata had practical uses for indexing, user studies have shown that people often remember the "default" elements of

[29] http://en.wikipedia.org/wiki/Systema_Naturae

information to a given source, such as approximate publication date and authorship (Hert, 2000), legal practitioners often recall the date of a decision, the judge(s) or even the parties when looking up. The process of defining which metadata to capture as well as to create and administer metadata is an expensive and time-consuming continuous process. A huge pitfall is that not all metadata is useful, and thus capturing the wrong metadata leads to useless work. The Dublin Core Metadata Initiative (DCMI) has introduced standards[30] and guidelines[31] for metadata. While the DCMI standards are flexible enough to support custom needs, they are most commonly used for annotating sources with the "default" metadata. DCMI defines the following fifteen elements as core metadata: contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title and type (DCMI, 2006).

4.2.2 Keywords and Tags

Keywords are one of the simplest forms of metadata scheme. While being simple, it has also proven very efficient in certain settings. Keywords are descriptive words that capture the main theme or essence of a source.

In the early days of the web, keywords were heavily used by search engine indexers, which read the "meta keywords"[32] in HTML pages, and used these in the weighting algorithms[33]. A problem with this approach early search engines painfully experienced was that spam, porn and various sites employed vast quantities of generic keywords to gain more hits for (often completely unrelated) search queries. This quickly led to search engines discarding the "meta keywords". Keywords however lived on and are still widely used internally in information systems, often hidden from the users, and used by the indexers and database queries.

Tagging is another implementation of keywords metadata. To *tag* means to annotate a *thing* with descriptive tags (keywords) as a *tagger* see fit. Tags are normally transparent and viewable to anyone, and often used as index words for finding other information objects with the same tag.

[30] ISO/TC46/SC4 (2003) *ISO Standard 15836-2003,* [2007-10-11], ISO<http://www.niso.org/international/SC4/n515.pdf>, NISO (2007) *NISO Standard Z39.85-2007,* [2007-10-11], NISO<NISO Standard Z39.85-2007>.

[31] http://dublincore.org/documents/

[32] Meta keywords were an early metadata effort where documents were described with keywords in the HTML tag, <meta>. For instance <meta name="keyword" value="example keyword use">.

[33] When indexers like search engines "weigh" information, they typically run through weight-algorithms that decide which information is more relevant.

Tags have really gained momentum the last few years and are now considered a standard feature in Web 2.0[34] systems.

A typical Web 2.0 use of tags is the pattern used in the blogosphere[35]. Bloggers annotate their blog posts with tags of themes they commonly blog about. The collection of tags used in a blog is called a *tag cloud.* Tag clouds illustrate the tags used in a blog, and the more often a tag is used, the bigger its font size. Tag clouds are great for giving a quick overview on what information is available and discussed, as well for finding pages for a given term.

Figure 4-3: Tag cloud

The major problem with keywords and tags-based metadata is the simple fact that different writers use different terms for the same themes and even the same documents. Without an editorial *control* over the available keywords, these systems tend to turn into metadata quagmires with terms pointing all over, and not separating the information. To solve this problem, *controlled vocabularies* such as thesauri, taxonomies and ontologies are used.

[34] Web 2.0 is a term that describes the recent focus on social web systems, information systems that focus on connecting people closer and sharing of information, opinions and knowledge. Web 2.0 systems usually contain a set of features like user comments, permalinks (permanent and non-changing links), link backs (reversed links (links from other sites to this), tags and are usually implemented using modern Web technology standards.

[35] Blogosphere, derived from Latin *sphere*, meaning world, and is now commonly used as a term for describing the blogging community, or blogs as a phenomenon.

4.2.3 Folksonomies

As tag based schemes evolved, some sites started to let the users collaborate in tagging information objects. When the users, and not only the authors, are allowed to tag it is called *collaborative* or *social tagging*. The tags from different folk (users) are collected, and form a base of tags called a *folksonomy*.

Figure 4-4 Screenshot of a del.icio.us view

This form of collaborative tagging can also be seen as a form of tag editorial (Christiaens, 2006). When several taggers tag a source, the "correct" tags can be found by analysing the frequency of a given tag, and a *consensus* can be found. This process can help to harmonise the tag base for a given source, as if it was edited by an expert group. A good example of a traditional folksonomy is the social bookmarking site del.icio.us[36]. Del.icio.us is an online bookmarking site where anybody can store their own bookmarks, annotate them with own tags, find other bookmarks with the same tags and access them from anywhere. If other users have bookmarked a particular

[36] http://del.icio.us/

page the system analyse their tags, and provide suggestions as to which tags other people have used (the most). The more taggers for a given source, the more precise the suggestions you receive when bookmarking a page.

4.2.4 Controlled Vocabularies and Taxonomies

Simple metadata vocabularies are not sufficient in describing information in domain specific contexts (Christiaens, 2006). A controlled vocabulary is in its simplest form is a list of carefully selected words that defines categories or concepts used in an information system. Taxonomy and thesaurus are different types of controlled vocabulary. The term taxonomy has been rather widely used and applied for more advanced structures schemes, but I use the simple classical definition of taxonomy (Garshol, 2004):

> *"a subject-based classification that arranges the terms in the controlled vocabulary into a hierarchy without doing anything further".*

Taxonomy can be illustrated as an indented list of words where each indentation is a simple relationship between these words in the form of *super-type* and *sub-type* (or *parent* and *child*).

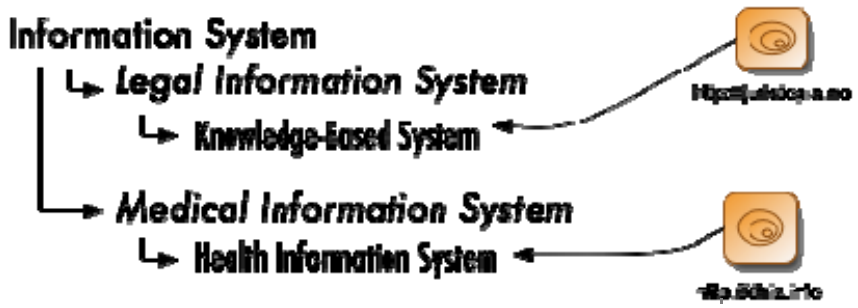


Figure 4-5: Example simple taxonomy

A super-type is the broader definition, and can have several sub-type synonymous subjects with more specific and/or narrower contextual meaning. Super-type and sub-type terms are therefore commonly also called *broader term* (BT) and *narrower term* (NT).

Fig 4-5: The instance http://juristopia.no can thus with taxonomy be classified:

1. Is a Knowledge-Based System.
2. Is a (sub-type of) Legal Information System.
3. Is an (a sub-sub-type of) Information System.

4.2.5 Thesauri

A thesaurus extends taxonomy even further. Like taxonomy, a thesaurus is a controlled vocabulary for holding hierarchical lists of important subject terms. Thesauri however have support for more *semantic controls* to define the relationships between the subjects in the list, beside the BT/NT scheme of taxonomy.

Thesauri are mainly used as a lookup feature when indexing information objects and have been used in information systems for a long time. Lovdata implemented a thesaurus for their indexing algorithm already in 1974 (Bing, 1984), and are still using various implementations of thesauri in both their indexing algorithms as well as in the optimisation of search queries. Where as thesauri have been proven very efficient for indexing, its use for optimising search queries is not as significant in measure (Bing, 1984).

The semantic controls of thesauri vary with implementations, but typical implementations use the controls as defined in ISO 2788 (Garshol, 2004):

- All terms used for indexing are called “Indexing Terms”. Like taxonomy, the main structure of a thesaurus is classified with BT/NT for synonymous terms.

- “Related Term” (RT) is used for things related to, but not synonymous. For instance Laws are *related to* preparatory work.

- Preferred and non-preferred terms are typically used for creating skip-lists, substitution lists and to preserve control of the available terms when indexing. These relationships are usually modelled by using the “USE” or “SEE” controls, with the option to model the inverse relationship “Used For” (UF). For instance: for the subject: *legal principle SEE precedent*. And the inverse: *precedent UF legal principle*.

- Scope Note (SN) can be used to annotate an ambiguous or unclear term with a descriptive string for the applied scope.

There are two basic approaches for constructing a thesaurus: “top-down” and “bottom-up”.

In top-down approach a committee or group of experts in a field are set do decide the scope and breadth of categories and terms that are to be included in the system. The committee first create an initial thesaurus, and are responsible for all later revisions.

The other approach is bottom-up: the thesaurus is built not by experts, rather by the users of a system and by analysing use cases and information patterns that appear as the installed base (number of users, and usage) for the system increase. In strict bottom-up there is no initial predefined scope, breadth of categories or terms, these "grow" as the users of the system require them. In real world applications the bottom-up approach is usually used with a hybrid top-down, where some experts approve or decline suggestions, but it is a good way of bootstrapping a thesaurus without a BDUF (Big Design Up Front).

4.2.6 Faceted Classification

Faceted analysis is another form of categorising commonly combined with thesauri implementations. A facet is simply put a generic category that can be ordered hierarchically like taxonomy. Facet classification, sprung out from Colon classification - the first faceted classification scheme which was developed by Hindu mathematician S. R. Ranganathan as he was working as a librarian in the 1930s[37] (Garshol, 2004; Gruenberg, 1992).

The purpose of the Colon classification scheme was to be able to *describe life, the entire universe of ideas, concepts and things* with a starting point in Ranganathan's *order* which consisted of five "top-level" facets that any information object could be deduced into:

- Personality (who?) – The persona or core subject discussed in the object.
- Matter (what?) – The theme or material being described in the object.
- Energy (how?) – The process described in the object.
- Space (where?) – The setting of the object.
- Time (when?) – The time frame of the setting.

An example of using Ranganathan's colon classification for instance on a commentary article on the "EU Commision Decision on Microsoft's antitrust in 'EU COMP/C-3/37.792 Microsoft'[38]":

- Personality: Microsoft
- Matter: antitrust (EC Treaty Article 82)

[37] Ranganathan published

- Energy: abuse of dominant market position
- Space: EU
- Time: August 2000 - March 2004

Ranganathan's colon classification scheme was since revised 8 times and led to a final generic facet classification scheme consisting of 46 canons (criteria for judgement), 13 postulates (core rules; or truths) and 22 principles (for how to solve deduce actions) (Gruenberg, 1992). This scheme is later considered too generic, and practise in the last decade has been to create schemes applicant to the target domain of information bases. Ranganathan's methodology for how to create a faceted classification scheme is still pertinent (Kwasnick, 1999; Gruenberg, 1992):

1. First decide on the scope and breadth of information objects that the information system will eventually index.
2. Analyse each (type) of information object, and identify facets. Create an initial scheme of broader facets.
3. Group *isolates* – simple subjects and concepts - into the facet scheme.
4. Re-order the isolates within the facets (place them into broader synonymous groups).
5. Establish a *citation order* for the facets. A citation order is the generic categorisation scheme (like Ranganathan's personality, matter, energy, space and time) for any information object.
6. Establish a *schedule order* for the facets. A schedule order is an internal weighting of facets, and can be compared to the preferred/non-preferred controls of thesauri.
7. Apply the notational system – the classification implementation, such as XML-based XFML (eXchangeable Faceted Metadata Language).
8. Process all the information objects and (re-)build the index.

The similarities to top-down thesauri are evident, and in practise implementations of thesauri often include principles from faceted classification theory.

[38] http://ec.europa.eu/comm/competition/antitrust/cases/decisions/37792/en.pdf

4.2.7 Graph Theory

In computer science and mathematics Graph Theory is the study on *graphs* - structures created by objects and the relationships between them. I will not go into the mathematics of graph theory, but a general overview of some interesting concepts is useful to mention. In graphs any object can be interconnected, have relations to any other object, which differs from the parent/child relationships of trees and nested sets. In graphs an object is called a *node*, and a relationship between two nodes is called an *edge* or *arc*. Information relationships are normally not simple parent/child relationships; in real-world information is interconnected and cross-referenced, information is usually connected on several levels, and has several relationships to one another. Figure 3-3 illustrates how some subjects are ordered in tree-based or hierarchical schemes:

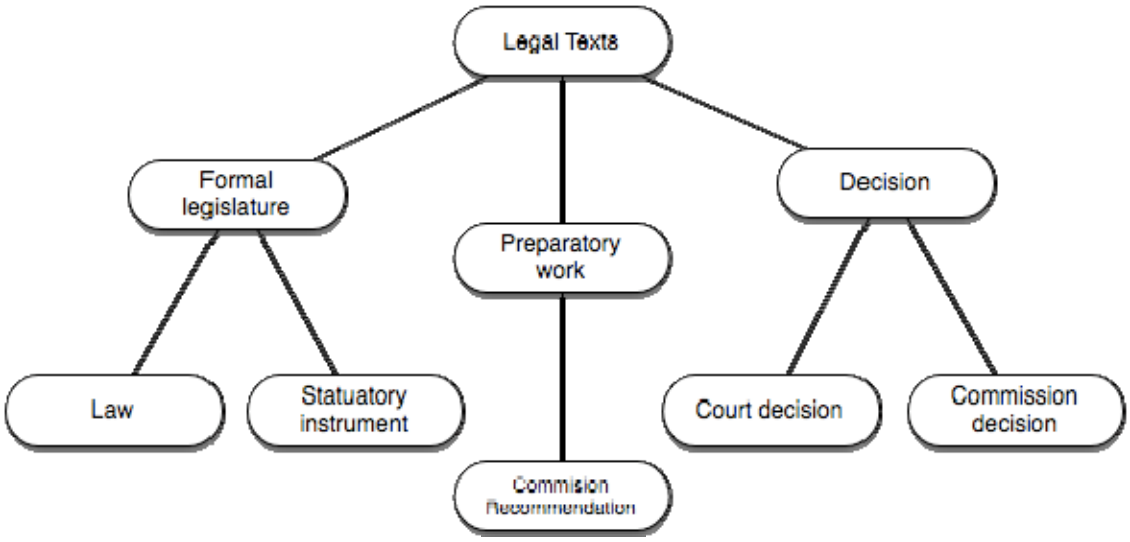


Figure 4-6 A tree structure of some example legal texts

Figure 3-4 illustrates the same subjects in a graph scheme, with some additional links between the different concepts. For instance a Law may be changed based on the outcome (precedent) of court decisions, and are usually heavily influenced by preparatory work, etc.

Basic graph theory also state that graphs can be either *directed* or *undirected*. In an *undirected graph* the edge has a single meaning while in a *directed graph* the meaning differs from which direction you read the edge. For example the edge in a directed graph between the nodes “Law” and “Preparatory Work” could be read “a Law implements a preparatory work” from the

perspective of the Law-node, and “a preparatory work is implemented by” from the perspective of the Preparatory work-node.

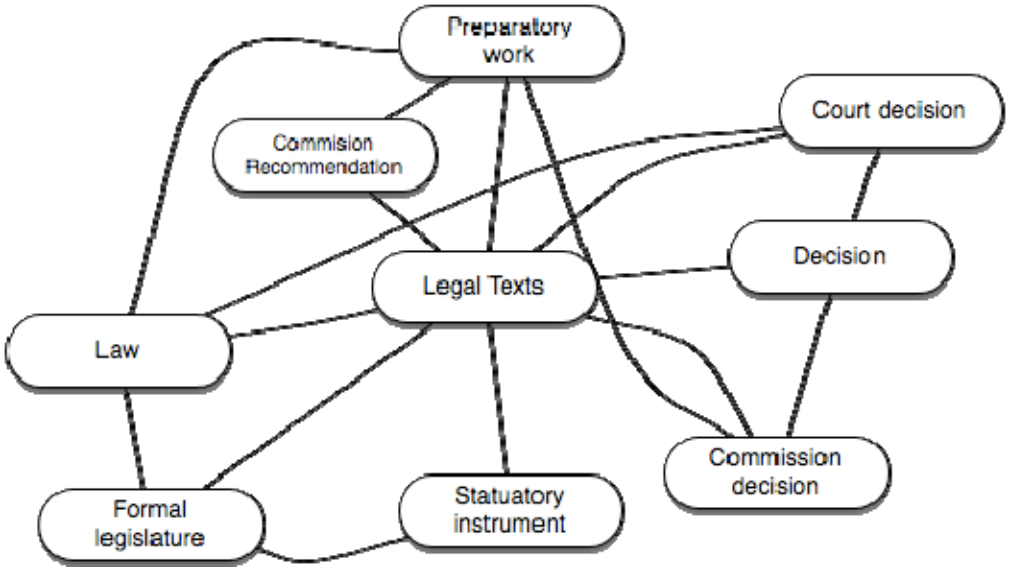


Figure 4-7 A graph illustration of the same legal texts

Another common feature of graphs is *weighting*. A weighted graph means that the edges can be annotated with a defined weight, which means something to the relationship. The meaning can vary from each implementation, but is commonly numeric; for instance the heavier the weight, the more important the relationship is than other relationships of a node; or for instance the lighter the weight, the closer in information value (similarity) the two sources are. Such weighting can be used for indexing algorithms as well as for processing logics. Nodes and edges in graphs can be computed in incidence (connectivity for relationships) and adjacency (same level) matrices which are lists or matrices that respectively specify the relationship (and weights) of nodes and edges, and the adjacency of nodes meaning that they exist on the same graph level (or sub-graph level). For instance synonymous subjects could be placed adjacently in a matrix, and defined in an adjacency-list.

4.2.8 Existential Graphs and Conceptual Graph

An existential graph is a type of digraph (sub-graph with two nodes) created to express the logics (for instance with Kripke frames) (for the edge) between two nodes. Existential graphs are used to describe the (one-to-one) relations and logics. Each node in an existential graph has a set of

properties – rules that describe the node. Conceptual Graphs (CG) and its Conceptual Graph Interchange Format (CGIF) notation is the most used existential graph format as of late. Existential graphs separates between three types of logic for digraphs (describing the quirks of the different forms of logic is out of scope in this thesis):

1. Alpha graphs use Boolean logic axioms and are commonly used on singleton data or metadata elements such as titles, document type, date, creator, etc. "This node AND that node are equal".

2. Beta graphs use first-order logic to describe the relationships through inference on simple axioms.

3. Gamma graphs use (normal) modal logic to describe the relationships through modal inference on the simple axioms.

CG and CGIF notation has been popular in the field of AI & Law research the last few years as an alternative, and as a staging kit to create an ontology because of the simplicity (Valente, 1995) for instance in creation of automated case reasoning tools for common-sense axioms.

The last and most important classification scheme we use today is ontology.

4.2.9 Ontology

Ontologies are widely used in Artificial Intelligence and Knowledge Based Systems, in applications relevant for knowledge and information management.

Classical Ontology (with capital O) has its roots in the philosophy, and the philosophy of being. The word stems from ***ontos***: *being* or *thing*, and ***logos:*** *meaning* or *reason*. More specifically it is the part of metaphysics that take interest in all characteristics of (human) nature and its relationships.

Ontology has several different definitions, seven of which were analysed by (Guarino and Giaretta, 1995) [39]:

[39] Original source not found, citation is from page 7 of Gómez-Pérez, A., Fernández-López, M. and Corcho, O. (2004) *Ontological Engineering,* Springer-Verlag London Ltd, Madrid, Spain.

1. *Ontology as a philosophical discipline.*
2. *Ontology as an informal conceptual system.*
3. *Ontology as a formal semantic account.*
4. *Ontology as a specification of a conceptualization.*
5. *Ontology as a representation of a conceptual system via a logical theory.*
    - *5.1 Characterized by specific formal properties.*
    - *5.2 Characterized only by its specific purposes.*
6. *Ontology as the vocabulary used by a logical theory.*
7. *Ontology as a (meta-level) specification of a logical theory.*

As seen, there are many definitions of ontology, and the term is used in several contexts, and for several different systems. In information science ontology (lower case o), a common feature of ontology is that it represents a structure - often domain specific - consisting of concepts, properties and the relationships between these.

The context of ontology in respect to Semantic Web, Topic Maps and knowledge representation, my preferred definition is the one of Mike Uschold and Robert Jasper, which defines ontology as a generic structure without being tightly connected to a single form of technology choice in terms of logics or other contstraints (Uschold and Jasper, 1999) page 11.2:

> *"An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms."*

While ontology is most often used in terms of being descriptive logics technology, it is also applicable to constraint logics theory and topic maps technology. Ontology in this form can also be determined more as a scheme than as a logic conceptualisation - it provides explicit information of parts of the conceptualisation - or sets of axioms designed to express the (constraint) logic meaning of vocabularies.

Ontology as Scheme

Ontology is also used in the knowledge representation layers for simple structures like metadata, thesauri, and other meta-information like a subject-oriented index for knowledge engines. Ontology is used to express and validate the relationships and associations between information, and as such can be used to validate that information exist and links are consistent. This is more of a schematic approach than an ontological approach based on (constraint) logics. Topic Maps and RDF are examples of technologies that can be used to create such schemes, through simple mapping or definitions of resources. Ontologies as scheme can be compared to a helper for

interpreting information, as a subject-oriented (concept-oriented) index, and an aid for extracting knowledge.

Ontology as Constraint Logic

The other part of ontology in information science is the concept of ontology as logic, as a technology to extract knowledge through reason or constraints. At the beginnings of the 1990s ontologies were mainly built using AI language and modelling techniques that leveraged ontology from being schemes holding meta-information, to well-defined schemes holding well-formed and defined (semantic) meta-information that could be used with descriptive logics or constraint logics. The main reasons for this change is tied to AI research, and the focus that knowledge, and axioms should be *highly formal* in order to make computers able to understand and infer knowledge as previously defined (Gómez-Pérez et al., 2004).

Technologies like DARPA Agent Markup Language (DAML) and Ontology Inference Language (OIL) were combined (DAML+OIL) to create a basis of logics that computers could understand. DAML+OIL were the first description logic standard to gain a lot of momentum, and the ideas has since been refined and revised into what is now the general Semantic Web research technology, in the area of description logics: Web Ontology Language (OWL). Combinations of RDF (to build the scheme) and OWL (to build the ontology), RDF+OWL, are now the most popular means of building logics based ontologies.

I find it important to stress that there are other knowledge representation technologies both for schema modelling (meta-information, class, inheritance, type, concepts) and for knowledge extraction (inference, reason, deduction, constraining) that are of focus and can be used for ontology based knowledge management than RDF+OWL. Such examples are Topic Maps for modelling and Topic Maps Constraint Language for extraction as I will later get more in depth on.

Ontology: Static or Dynamic?

Many ontology based systems focus on using a static ontology (logics) for use to extract knowledge on the knowledge (meta-information) model. In this approach, a fixed set of relations are mapped, that are not likely to change, to describe higher level types, classes and concepts used in the domain. These fixed ontologies are often called CORE ontologies. I have created a table of a few ontologies used in current LIS research:

ONTOLOGY	TYPE	TECHNOLOGY
Lehmann, Breuker and Brower's *CAUSATI$_o^{nt}$* (Lehmann et al., 2005)	General expressive ontology for reasoning about common-sense causations for legal information.	Web Ontology Language
Valente and Breuker's *FOLaw* (Functional Ontology of Law) (Valente, 1995)	Core ontology for modelling legal knowledge, reflecting on knowledge types and dependencies in legal reasoning.	Highly structured Ontolingua[40], expressed with Knowledge Interchange Format[41].
Estrella Projects'[42] *LKIF-Core* ontology (Breuker et al., 2007)	Core ontology for modelling legal knowledge, focus on common-sense basic legal concepts and successor to FOLaw.	Web Ontology Language
Benjamins et.al.'s ontology for judges. (Benjamins et al., 2004)	Ontology for legal information retrieval system for judges.	Web Ontology Language and Protége[43].

The other approach is that the domain model itself controls the ontology, and information is inferred by constraining views on the ever evolving ontology (schema). This is the *dynamic* perspective of ontology. I still haven't found a single research source for this perspective, but I will discuss Lars Marius Garshol's view on this later (see 5.4).

[40] http://ontolingua.stanford.edu/

[41] Knowledge Interchange Format (KIF) is a format to express logic axioms: http://www-ksl.stanford.edu/knowledge-sharing/kif

[42] Estrella Project is an European project for Standardisation of Legal Accessibility to develop and validate an open standards-based platform for legal knowledge management solutions: http://www.estrellaproject.org/

[43] Protégé is an ontology editor and knowledge acquisition system to create and experiment with ontologies: http://protege.stanford.edu/

4.2.10 Ontology Engineering

Authoring topic maps and creating or modelling OWL/N3 ontologies are forms of ontological engineering, a form of research that are usually performed by experts in a field or domain.

I could have undertaken a thorough exploration of what legal information is out there, and how to map this information could be organised into a core ontology, and thus expand the built-in topic types I already have, but I chose not to because my thesis is already too wide. This is the main reason I have not tried to, or undertaken a thorough exploration of creating ontologies. I have experimented with creating ontologies for Law texts, preparatory work, and mapping Lovdata's structure of databases onto a topic map.

As with folksonomy as previously mentioned, the collaborative perspective of social web applications like Wikis can assist in ontology creation, tuning and enable "growing" ontology. An ontology that grows and is changed as editors and the information base grows, preferably by elitists (specialists) such as knowledge managers or legal scholars. Another reason for not getting into this paradigm is that companies of such size that will be able to successfully employ Wikis for internal purposes will usually have knowledge managers that are able to organise the Wiki according to their own organisation of documents, and their own organisational structure of teams, groups, and specialists on fields of law much better than me. This was also a wish expressed during the interviews with knowledge managers, but more on that in the next section.

Part III

Semantic Web

5. The Semantic Web

When, where and to whom the vision of a universal system to bridge all knowledge belong to is impossible to tell. The web as we know it today had many starts and influences, but it is commonly agreed upon that Tim-Berners Lee is the father of the World Wide Web (Schwartz, 1997) , and invented it in 1989 when he was working at CERN.

Berners-Lee is also considered one of the fathers of the "Semantic Web" vision that came forth the last five years of the previous millennium. When I say vision, I do so intently, as there *is no single definition* on what the Semantic Web is. The Semantic Web is the architecture for the next generation of the WWW - a Universal Web, a web of interoperability, where all information is made accessible, findable, permanently, and to allow understanding[44] of data and its associations (semantic links).

Lately the use of the term *Semantic Web* has been wrongly associated to the W3C's set of suggested standards and practise for the next generation web. I will in this chapter first quickly describe W3Cs stance, then the vision, then the technology advance and finally go onto topic maps as a means of creating a Semantic Web system.

5.1 W3Cs Goals of the Semantic Web

W3C does not provide a definition as to what the Semantic Web is or which technologies that it consists of, but rather suggestions on how they would like it implemented. Their rationale for the Semantic Web is to create an interoperable web of information, where data is available and compatible across applications and organisations, through a common vocabulary for information exchange, based on the principle that URIs (Uniform Resource Identfiers) are used for navigation and identification of concepts and things, and that these URIs are exposed and available between Semantic Web applications.

W3C defines two main goals for the Semantic Web (Herman, 2007):

> *The Semantic Web allows two things.*

[44] Understanding for computers means it is well defined and parseable to computers.

1. *It allows data to be surfaced in the form of real data, so that a program doesn't have to strip the formatting and pictures and ads off a Web page and guess where the data on it is.*

2. *it allows people to write (or generate) files which explain—to a machine—the relationship between different sets of data. For example, one is able to make a "semantic link" between a database with a "zip-code" column and a form with a "zip" field that they actually mean the same – they are the same abstract concept. This allows machines to follow links and hence automatically integrate data from many different sources.*

URI Identity and No Dead Links

The third goal of the Semantic Web can be discussed to be that all resources are identifiable through URIs, and that these URIs can be controlled for consistency, that they exist. Links to URIs should not only be Links, but also validated, and possibly annotated (Berners-Lee, 2002).

In the Semantic Web, the *cement* or *glue* is the hyperstructures and hyperlinks between identifiers (URIs). As information objects usually carry some semantic bits and pieces (see further below) these semantic rules can be attached to the URI, but should only the weaker relationships should be attached as information resources change (Berners-Lee, 2002):

> *"... but the philosophy is that an HTTP URI may identify something with a vagueness as to the dimensions above* [edit: changes in information objects]*, but it still must be used to refer to a unique conceptual object whose various representations have a very large a mount in common. Formally, it is the publisher which defines the what an HTTP URI identifies, and so one should look to the publisher for a commitment as to the exact nature of the identity along these axes."*

In order to meet these goals, there are some underlying features that need to be in place. The first goal states that data (and information) needs to be well-defined so that machines can understand the data in them. Secondly the data needs to be so well-defined that the computer is able to understand and infer knowledge from the data. One important thing to note is that W3C does not specify that the Semantic Web is Artificial Intelligence, but *suggests* AI as *one* possible technology that can be used in Semantic Web applications for solving interoperability and semantic knowledge inference.

5.2 The Vision of a Semantic Web

To repeat: The core principle in the vision of the Semantic Web is a web where information objects can be described, accessed and identified through the use of unique URIs. It also enables and exposes the associations between these objects, so that information becomes more available to the users.

The Memex

A potential father to the vision is Dr. Vannevar Bush an American researcher who excelled in all fields he was involved in, whether it was nuclear research, innovation, engineering, or research methods. Vannevar Bush is perhaps most known for two of his achievements in particular: for writing the memo that launched the Manhattan Project, and secondly for his "memex" (Memory Extender) system envisioned in his article "As we may think" published in The Atlantic Monthly, July 1945 (Bush, 1945). The latter is cited in pretty much all serious theses relating to hypertext or information management. The memex was portrayed as a mechanical private file and library, with a monitor, a keyboard and storage capacity for hundreds of years. The memex would store all books, records and communications, and index them accordingly so that whenever you wanted a document, you could simply ask for it on the keyboard. More important, was that he envisioned the possibilities to interconnect all the information, based on its contextual associations through associative (subject-oriented) indexing (Bush, 1945):

> *"…an immediate step, however, to associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the memex. The process of tying two items together is the important thing."*

He envisioned that researchers could connect information in their own memex, and build and store "trails" of selected information on any given topic, composed from all of the available information in the endless maze of information. These trails could be shared with other friends and researchers, so that they could also retrace the entire path to enlightenment, create their own tangents or offspring trails to build on the same research on their own memex as they wished. Sharing information was a big part of Bush's vision, after the war he recognised that research had stretched across borders, and was now a global science. Research groups were no longer located together at the loft of a University or corporation, but spread hundreds of miles apart. To share knowledge and information new systems for communication were needed. Bush also realised that technology is both changing society, at the same time technology is influenced and

formed by the ideas from society. New technology enables us to perform tasks we earlier just dream about; while technology can also constrain the tasks we are to perform.

The memex is a revolutionary vision that still is relevant in the development of the Internet, the World Wide Web, associative indexing (subject-oriented), and knowledge-based systems of today.

Project Xanadu

Theodor Holm Nelson, born 1937 was an American sociologist and philosopher who in 1963 introduced the term hypertext at a conference. Project Xanadu was an ambitious decentralised information systems project which was conceived in 1960. Its aim was like Bush's memex to share information, preserve associations while at the same time accounting for copyright and version management.

In "Literary Machines", Nelson describes the future (Nelson, 1981):

> *"Forty years from now (if the human species survives), there will be hundreds of thousands of files servers, and there will be hundreds of millions of simultaneous users." ... "All this is manifest destiny. There is no point in arguing it, either you see it or you don't"*

Project Xanadu was aimed to provide hypertext from millions of file servers or Literary Machines that held version managed information that could be accessed from anywhere (decentralised) while at the same time keeping account on who access the material, as well as preserving all hyperlinks through version management. Like the memex, Project Xanadu focused on the associations between information. Nelson envisioned a universe or *docuverse* of related documents, where the *"everything is deeply intertwingled"* (Nelson, 1987). With "intertwingled" he meant that all information and knowledge was both intertwined and intermingled. Unlike Bush, Nelson meant that there were no core subjects, and subject-oriented indexing were only a compromise, knowledge doesn't conform to certain subjects, or belong in hierarchical or sequential categorisation or classification.

Nelson did not believe in structured mark-up systems like HTML, but rather non-linear structured system where texts and other information media were related and accessed through the associations, not their machine readability. Nelson also later cursed the WWW for developing into everything the Xanadu tried to prevent (Nelson, 1999):

> *"The Xanadu® project did not "fail to invent HTML". HTML is precisely what we were trying to PREVENT-- ever-breaking links, links going outward only, quotes you can't follow to their origins, no version management, no rights management."*

Xanadu has also often been ascribed and criticised for being a royalty accounting system: a system mainly for preserving copyright, and accounting for royalty payments between authors and users. Nelson did indeed focus a lot of his research on this aspect, but that was not a *primary goal,* but rather an essential feature for creating a system that could work in real life, and pertain to all information available. This is an interesting aspect to the debates around intellectual property projects on the Internet, such as Google Books, Youtube and other criticised (and litigated) systems, but that is another discussion.

The vision of Project Xanadu is getting more and more relevant in this day and age, and I find the most central principles we can learn from it in respect to the Semantic Web are: focus on associative and multidirectional links, version management (to preserve link breakage) and that there is no *one* classification scheme that can encompass a particular information object in a subject-oriented index.

A Perspective from Legal Systems Theory

Francis Hagerup, Norwegian Prime Minister between 1895-1898 and 1903-1905 as well as one of Norway's foremost legal scholars, professor at the Christiania University (University of Oslo), parliament member and later ambassador of Norway wrote an article published in the first edition of Tidsskrift for Rettsvitenskap (Journal for Legal science) in 1888 (Hagerup, 1888). In this article he described the "new" legal method, jurisprudence and also delved onto the semantics of legal texts. He described a new form of legal science and in terms of a subject-oriented focus on the legal concepts and more important that all the associations of legal texts and legal norms could be considered as a fully connected system where you could from any subject trace the theory to any other subject.

A few decades later Hans Kelsen published his legal theory. In this theory I focus on parts of his legal system that builds on the same concepts as Hagerup: that the legal system is interconnected. Kelsen illustrated the legal system as a complete system of ideas and norms in which all ideas and norms together can be represented as a tree which leafs can be traced and back-traced to disambiguate the connections and associations and determine the causations and validity of any norm or legal concept (Raz, 1980b). Kelsen's theory was since criticised by John Austin in that Kelsen assumed a "Grundnorm" (base norm) from which all other norms and

concepts descended from, however no inflictions were added to criticise his illustrations of the legal system it self (Raz, 1980a).

5.3 Web Technology

5.3.1 Hypertext

The memex is remarkably similar to what the World Wide Web has enabled today with hyperlinks. You can follow trails of hyperlinks as your friends publish web pages; however the WWW is not yet capable of following these trails in both directions, there are no guarantees that they lead somewhere (404: non existing documents), nor are hyperlinks capable to understand or validate what they are connecting. Hyperlinks form hyper structures, but these hyper structures are not capable of describing the information structure. This is where the Semantic Web vision comes in: to provide a consistent trail of knowledge, that machines can understand as well as assist in creating and validating information and structure.

Machines have since early been able to read and write different formats, from ASCII text-files to binary formats. Berners-Lee created the mark-up language known as HTML (HyperText Markup Language) to use for the World Wide Web. HTML is a mark-up language that can be read and written by both humans and computers. A mark-up language is a format that is able to describe information by using annotations; directives that tell machines what kind of information is following. Once machines know what kind of information they read, they can handle, format and treat the information found in the marked-up document. HTML is a descendant of SGML (Standardized General Markup Language), but is not as rigid as SGML on what requires mark-up. *Element tags* are the core of both SGML and HTML, and are in HTML used as directives that specify mainly how machines are to render the text (draw it on a screen), and format the output.

Tags are enclosed by the characters ‘<’ and ‘>’, for example: headings: *<h1>, <h2>, <h3>*, paragraphs: * Lorem Ipsum dot silit. * and anchors *<a>*. There are two main types of tag enclosing: opening tags ( ) are used to declare the start of a directive, and closing tags ( ) are used to declare the end of a direction (in this case a paragraph). In HTML (except HTML v4.01 Strict), tags are not required to be “closed”, but were rather decided by a simple parsing-tree: tags were considered open as long as the following tags were considered descendants of the current open tag. This design led to confusion, and is one of the reasons that different web-

browsers did not display the same texts in the same manner. HTML has since been revised to *recommend* that all tags are to be closed, including single-case tags (like images <*img*>, line breaks <*br*>, horizontal lines <*hr*>, et cetera), however they can be now be opened and closed within one single tag by adding a '/' next to the end i.e.: "<*br/*>".

The HTML tag that has been the most important is the anchor tag "<*a*>". Anchors are used to:

- Create in-page navigation:
  **,

- Hyperlink to in-page anchors:
  Go to Anchor,

- Hyperlink to other web pages on the same site:
  Link to Page,

- Hyperlink to other web sites:
  About.

Anchors are unidirectional associations to identifiers called URIs[45] (Uniform Resource Identifier). URL (Uniform Resource Locator) and URN (Uniform Resource Name) are more restrictive subsets of URI. URI is now the preferred term for URL and URN. A URI is a string that identifies an information resource. Resources include web pages, anchors, web sites, web servers, document elements, et cetera. For instance: "*http://juristopia.no/-/*", "*svn://juristopia.no/jt/trunk*", "*#name*", "*/*", are all valid URIs. The first two are fully qualified URIs with protocol (http and svn (subversion)), the third is an internal URI to the #name anchor () on the same document, and the last is a link to the root/index of the current web server.

Tags can also have attributes (sometimes called properties or parameters). Attributes are found within the opening tags, for instance a tag showing an image without a border: <img id="thePageLogo" src="logo.png" border="0"></img> has the attributes *id*, *src* and *border* with the respective values "thePageLogo", "logo.png" and "0". Special attributes are *id* and *class* (*name* is deprecated) were introduced with HTML v3[46]. These are identifiers that can be used to

[45] http://gbiv.com/protocols/uri/rfc/rfc3986.html

[46] http://www.w3.org/MarkUp/html3/html3.txt

identify certain element tags in an HTML page, through the DOM[47] (Document Object Model) which is a neutral interface to access, navigate and update raw documents.

HTML has been revised a lot since its conception, and is fairly stable now at version 4.01[48]. As web usage has grown HTML's limitations have become more and more visible. CSS[49] (Cascading Style Sheets) is a standard that now handles most of the representation formatting of web-pages, and other mark-up languages like XML (Extensible Markup Language) are gaining ground to replace HTML.

5.3.2 From Machine-readable to Machine-understandable Data

The reasons for XML-based mark-up is getting more popular is that HTML is not able to tell anything *about* the information or the *semantic meaning* of the data by itself, other than providing directives that tell where different elements of a page is and how to display these elements. HTML is machine-readable, but the information read is not *understandable* to the computer.

Extensible Markup Language is like HTML derived from SGML. XML is in essence a simplified version of SGML that focuses on SGML's "notorious" emphasis on describing all elements through mark-up. XML is an open standard that is designed for interoperability, and *extensibility*. XML is in other words a very flexible and generic language, in fact so generic that users have to define their own tag names for the information they are marking up. This does however open up for problems, as this implies that these elements must also be described what are, and how to render for machines to understand. XML has solved this with other XML-based technologies like XML Schema (XSD), Relax NG, Extensible Stylesheet Language (XSL) including XSL Transformations (XSLT), XSL Formatting Objects (XSL-FO), and several other technologies and standards.

The main problem with XML-based technologies like HTML is that it requires *a lot* of configuration to make it able to understand data, but the XML-technologies are more generic and thus is easier to read and re-use than the *HTML–way* using *scripts* and *programs* to achieve understanding. XML Schema and Relax NG are technologies for describing the syntax, like

[47] http://www.w3.org/DOM/

[48] http://www.w3.org/TR/html4/

[49] http://www.w3.org/Style/CSS/

SGML's DTD (Document Type Definition). XML can use schemas (as well as DTDs) with a more obvious and XML-like format to specify the *data types*, *simple constraints* on for example *what* elements can be nested within *other* elements, *how many*, which data are *required* etc. Extensible Stylesheet Language (XSL) is used to transform XML documents between forms, for example from XML into XHTML, or from *my* schema to *your* schema. To provide an example as to how XML can be understood by a computer, take a simple XML fragment.

From this loosely described text below a machine is already capable to easily read all the articles, or the first member of chapter 2 article 12, but by itself it isn't told anything about the data, and in many ways not different from regular HTML.

```
<?xml version="1.0" encoding="utf-8"?>
<law id="lov-2000-04-14-31" lang="no">
   <effectiveDate>2001-01-01</effectiveDate>
      <name>Lov om behandling av personopplysninger</name>
      . . .
      <chapter cid="2">
         <title>Alminnelige regler for behandling av Personopplysninger</title>
         <article aid="12">
            <title>Bruk av fødselsnummer m.v.</title>
            <member>
               Fødselsnummer og andre entydige identifikasjonsmidler kan
               bare nyttes i behandlingen når det er saklig behov for sikker
               identifisering og metoden er nødvendig for å oppnå slik
               identifisering.
            </member>
            . . .
         </article>
      </chapter>
      . . .
</law>
```

Figure 5-1 Example XML Law Document

It is *well-formed* XML (apart from the abbreviation "…"'s) as the syntax of the XML document is correct, but it is not *valid* XML, as the tags and elements have not been defined or declared in a schema.

To become valid XML descriptions of where in the document a node is, and of what type it is are needed. Such descriptions of the different elements and types can be defined by using XML Schema (XSD), Relax NG or even Document Type Definitions (DTD). Relax NG is a flexible and simple to understand schema language for XML documents. It supports two different formats: XML-based and Compact. For instance if we wish to be able to parse the effective date

field of the document, and make the computer understand that this field is of type *date*, we can define this with the simple Relax NG (in XML format) document below (this example is abbreviated to only define the *law* node in respect to the *effectiveDate* node of the above XML example):

```
<?xml version="1.0" encoding="utf-8"?>
<grammar
   xmlns="http://relaxng.org/ns/structure/1.0"
   xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
   datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
   <define name="law">
      <element name="law">
         <a:documentation>A law (root node).</a:documentation>
         . . .
         <ref name="effectiveDate"/>
         . . .
      </element>
   </define>
   . . .
   <define name="effectiveDate">
      <element name="effectiveDate">
         <a:documentation>The effective date of the Law</a:documentation>
         <data type="date"></data>
      </element>
   </define>
   . . .
   <start>
      <choice>
         <ref name="law"/>
      </choice>
   </start>
</grammar>
```

Figure 5-2 Example Relax NG to parse date

Now the machine is able to parse the document and validate both that the *effectiveDate* node is placed inside the *law* node, and that the contents is of type *date*, thus making it *valid* XML.

5.3.3 Semantic Bits and Pieces

One of the strengths of Wikis is however that the usability is principled on simple and few syntactical options. Editing is not in general performed by specialists in mark-up languages, and the editors just want to edit texts, and not care about the layouts. Principally I do not wish to implement a full XML syntax to pages, and impose strict forms of formatting of Wiki pages.

HTML has support for identification of certain elements through *id* and *class*; it is still dependent on more complex operations on DOM to understand the meaning and types of elements. Considering the *well-formed* HTML code:

```
<span id=”today”> 2007-10-15</span>
```

One can easily output that date to the screen in any web browser, and one can even use its *id* identifier to find the element in the DOM with for instance JavaScript[50]:

```
todayNode = document.getElementById(‘today’)
```

Even though this is possible, HTML code still isn’t *well-defined*; the date *2007-10-15* is simply a text-node without any form of type or constraints on it and the computer cannot without being explicitly told, or the use of comparisons or regular expressions know if it is a string, three numbers separated by hyphens or an ISO 8601[51] conforming date. In order to make the semantics of the information and data understandable, it needs to be *well-defined*. This is where XML usually comes into the picture, XHTML is essentially HTML written in XML and conforms to (most of) HTML 4, as well as XML DTDs. In XHTML it is trivial to add an XML namespace holding the definitions for the data types, one can achieve the same meaning for certain elements using XHTML one can annotate certain data elements:

```
<html   xmlns=”http://www.w3.org/1999/xhtml”
xmlns:xsd=”http://www.w3.org/2001/XMLSchema-datatypes.xsd”>
          . . .
<span id=”today” datatype=”xsd:date”>2007-10-15</span>
. . .
<span id=”yesterday”><xsd:date>2007-10-14</xsd:date></span>
</html>
```

[50] JavaScript is now standardised through ECMAScript-262 specification available at: http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf

[51] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874

In other words XHTML makes it possible to create well-defined data of certain bits and pieces and even use other XML schemas as mark-up in the same document without the need to fully define the document.

Jon Bing proposed this approach as a simple means of annotating semantic "copymarks" for copyrighted material for web pages in 2003 (Bing, 2003). For Wiki technology the copyright discussion itself is interesting. Where a lot of authors contribute, write and revise on each others information such an approach could be used for defining different scopes of copyright within a certain document, or even annotating cites from other sources with the appropriate "copymarks". These could also be used as control mechanisms to hinder editing of certain quotes and cites of for instance Law text, and so forth. But that is another discussion. Bing proposed a combination of XrML[52] (Extensible rights Markup Language), RDF (Resource Description Framework) and RDF-based DCMI (Dublin Core Metadata Initiative) standards to be used. In addition to the *datatype* attribute the *property* attribute is interesting to mention. The property attribute is an addition proposed with XHTML 1.1 to add further data awareness to data in web pages. It can be used for instance with metadata standards like DCMI (Dublin Core Metadata Initiative), XFN (XHTML Friends Network) and XMDP (XHTML Meta Data Profiles). XFN and XMDP are metadata proposals for adding semantic value to social networking bits and pieces such as links to other sites, and really simple metadata bits like DCMI's standards. XMDP uses the HTML *profile* attribute, while XFN uses the *rel* attribute of anchors to add semantic values and rules, while DCMI as a standard is usually defined through the W3C proposed *datatype* and *property* attributes.

When needed this approach is the more pragmatic method of adding *well-defined* data elements in social systems like Wikis.

5.3.4 Loosely Coupled Meta-Information

Another aspect to information association in Wikis is that the information content, and knowledge, and not only semantic bits and pieces are part of the associations that can be extracted from a given source, not only through that the smaller bits can be well-defined, but also the over all information object can be categorised and classified as part of a bigger subject.

[52] http://www.xrml.org/index.asp

Wikipedia employs such an approach through allowing authors to use special category codes to create associations through back-links and other typical Wiki features.

By following the W3C suggested approach to the Semantic Web typically one would try to describe the information object in means of the semantic bits and pieces, by using RDF, OWL description logic and other annotations to the data and information. This imposes a big effort for authors in terms and challenges the Wiki principle of simplicity. One of the most appealing aspects of Topic Maps technology as opposed to the W3C suggested approach is that it separates the meta-information from the information while at the same time describing associations between topics.

5.4 Semantic Web Technology

There are several technologies supporting the Semantic Web vision. This section will rapidly mention some of the most important, but mainly focus on Topic Maps technology.

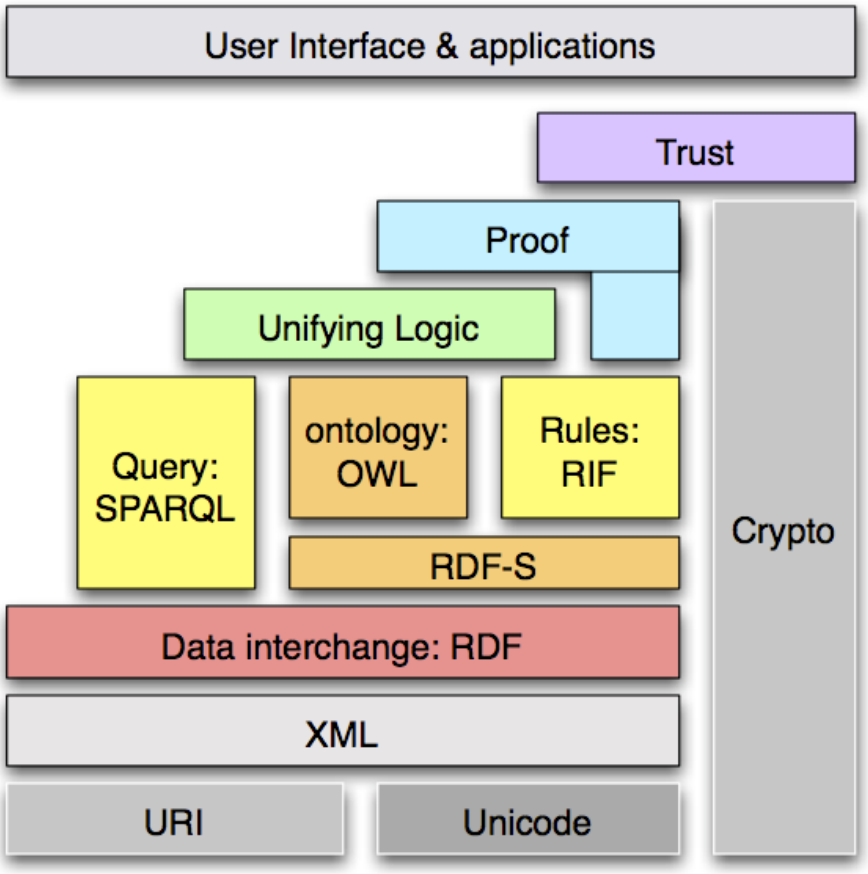


Figure 5-3 Technology suggestions for the Semantic Web (Berners-Lee, 2006)

These are suggested technologies to use in a Semantic Web project. A common false presumption or myth about the Semantic Web is that it should be implemented using the above stack of technologies. The technologies in Figure 4-1 are only *means* of implementing a system where information is exposed and identified across different sites. Description Logic like OWL (Web Ontology Language) is not a Semantic Web requirement, but rather a language that enables us describing information association between URIs, and as such is part of creating the Semantic Web through describing the relationships and associations between information objects found through URIs.

The technologies that the W3C suggest are optimised for computers and Artificial Intelligence to be able to assist the creation of the Semantic Web vision, topic maps technology as I will later discuss is another means of creating a Semantic Web where associations between information objects can be also described.

5.4.1 Resource Description Framework and Web Ontology Language

Figure 5-3 are W3C's suggestions for technologies and web standards that support the Semantic Web, are based on the RDF+OWL family.

RDF is a format for describing metadata for information resources and their properties. RDF was initially introduced as a framework for interchange of simple metadata on top of XML documents, and has grown from this limited use, to be a general and flexible format to break information into semantic pieces that are easily processed by computers.

Resources or Entities

A resource or entity is any information object or thing that can be represented in the RDF model. Instances of resources are any object, such as a person, a law text or an (law) Article. Resources have names, which are URIs that are global identifiers to the resource (for example a document or an image). Resources can appear in RDF as subjects, predicates or objects.

RDF Triples or Statements

RDF triples, also called statements, are small sentences in the form of "subject predicate object" (referring to resources) that describes a rule (axiom) about a resource or a property of the resource (subject) in relations to other resources (predicate, object). The objects can be literal values and be given types, for instance dates or values and specified with XSD data-types. RDF

can be expressed with XML or the simple N3[53] Notation variant like in the example below describing me, where you can find my homepage (through the FOAF[54] RDF Vocabulary (foaf) namespace), etc:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix jt: <http://juristopia.no/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

jt:OleChristianRynning dc:title ”Ole Christian Rynning” .
jt:OleChristianRynning foaf:schoolHomePage <http://folk.uio.no/olecr> .
jt:OleChristianRynning jt:Alias jt:oc .
jt:OleChristian foaf:image jt:images/oc.png .
```

Dublin Core Metadata Initiative (DCMI) as mentioned in chapter 4.2.1 uses the RDF format to express the “standard” metadata. As mentioned earlier in chapter 4, RDF is often consolidated and stored in RDF Triple Stores that collect and merge resources in order to express more information in the knowledge layer.

RDF Schema

RDF Schema is a language to describe and restrict RDF data. With RDF schema one can create logic systems and namespaces that describe the resources of RDF documents. One can also use it to limit and simple validations of literal values (i.e. objects of triples) such as types and create restrictions that for instance say that a #name resource is in the domain of #person.

Web Ontology Language

RDF alone is not enough to create well-formed semantic information. RDF is used along with OWL (Web Ontology Language) to describe relationships and class types through ontology. Like RDF, OWL has several syntaxes, including XML based and simple. Below is an excerpt example for describing a person using the predicate and type descriptive logics of OWL:

[53] N3 or Notation 3 is a very simple format to express RDF triple statements: http://www.w3.org/DesignIssues/Notation3.html

[54] http://xmlns.com/foaf/0.1/

```
Namespace(oc = <http://oc.rynning.no/example/#>)
Namespace(jt  = <http://juristopia.no/example/#>)

Ontology(
 ObjectProperty(oc:author)
 ObjectProperty(oc:creates)
  inverseOf(oc:created_by)
  domain(jt:wiki))
 ObjectProperty(oc:writes
  range(jt:wikipages))

 Class(oc:person partial
   annotation(rdfs:comment "Weird things that are human.")
 Class(jt:wiki partial
   annotation(rdfs:comment "WikiWikiWeb"))

 AnnotationProperty(rdfs:comment)
 AnnotationProperty(rdfs:label)

 Individual(oc:OleChristian annotation(rdfs:comment "Ole Christian")
   type(oc:person))
 Individual(jt:Juristopia annotation(rdfs:comment "Juristopia")
   type(jt:wiki))

 AllDifferent(oc:OleChristian jt:Juristopia)
)
```

One can then infer rules like Ole Christian writes wikipages:

```
Class(oc:author complete intersectionOf(oc:OleChristian
  restriction(oc:creates someValuesFrom (jt:wiki))))

Class(oc:writes partial jt:Juristopia)
```

5.4.2 Topic Maps

Topic maps were designed to handle construction of advanced indexes stretching beyond glossaries, tables of contents and thesauri. Topic Maps technology together with Resource Description Framework (RDF) can be foundations of the future Semantic Web, a web where information is connected, findable and understandable.

Topic Maps (TM) is standardised through ISO/IEC standard 13250, with subchapters 1-7. Topic Maps standards also include: TM Data Model (TMDM), TM XML Syntax, TM HyTime Syntax,

TM Constraint Language (TMCL), TM Query Language (TMQL), TM Reference Model (TMRM) and TM Graphs.

Topic Maps can be represented in several formats[55], but in this chapter examples I will use XML Topic Maps (XTM) (Pepper and Moore, 2001) in examples, but refer to the technology as defined in the TMDM (Garshol and Moore, 2006).

Topic Maps are far more advanced technology than RDF / RDFS, but yet as simple to understand, and simple to implement. Topic Maps builds on ontology thinking and is a simple model for classifying and storing meta-data for complex knowledge and information bases. Topic Maps is suited to capture these complex relationships both on smaller as well as higher levels of cognition and semantic relationships.

Topic Map constructs

Topic Maps consist of several constructs, in this thesis I will only go into basic depth in this thesis. The basics: Topics, Associations, Occurrences, Scope and Subject Identity, should be understood, and I stay clear of other constructs and concepts like reification, locators and only cover basic merging principles. There are two main types of topic map constructs: statements

Topic

Topics are the main construct of a topic map. The word topic stems from the Greek word ***topos***: *subject* and *location*. Anything and everything can be a topic: a topic can for instance be a meta-data representation of a thing, which can be any information object, concept, subject, etc. Topics have a subject which is the primary characteristic of the topic. This subject can for instance be an information object, and can either constitute the subject, or it can indicate the subject (point to) another addressable artefact, for instance through an URI. Topics can also be given an explicit type, or the default type, topic, will be used.

Topics have zero or more names. The most common topic has a single (base) name. Names can be either base names or variants which are variations of names, such as for pluralisation: person-people, and so on. For example the Norwegian Privacy Act law text contains several different (unique) names that all represent the same document: (id) *LOV-2000-04-14-31*, (title) *Lov om*

[55] XML Topic Maps (XTM), Asymptotic Topic Maps (AsTMa=), Linear Topic Maps (LTM) and HyTime are all notations used to create topic maps.

behandling av personopplysninger, (short title) *personopplysningsloven* and (abbreviation) *popplyl*, can be written

```
<topic id="Norwegian-Privacy-Act">
  <instanceOf>
    <topicRef xlink:href="#Norwegian-Law"/>
  </instanceOf>
  <baseName>
    <baseNameString>Personopplysningsloven</baseNameString>
  </baseName>
  <baseName>
    <baseNameString>
      Lov om behandling av personopplysninger
    </baseNameString>
  </baseName>
  <subjectIdentity>
    <subjectIndicatorRef
          xlink:href="http://juristopia.no/Personopplysningsloven"
    />
  </subjectIdentity>
</topic>
```

Through having the possibility of giving a topic several names, and variants helps to make topic maps a very flexible representation of a given information object.

Occurrences

Topics contain zero or more occurrences of information related to and descriptions of the subject represented by a topic. The topic occurrences can be either stored as internal information, normally only used for short texts and strings, or they can be external addressable units of information, identified by URI or HyTime variable links. Occurrences are information objects that are either instances of, indications to the subject matter, or the subject matter itself. An example is a (external) reference to “Privacy-Act”. Here you can see that the occurrence has a type (Norwegian-Law) which for instance is a topic describing what a Norwegian Law is, and a URI to the law text at Lovdata.

```
<occurrence>
   <instanceOf>
      <topicRef xlink:href="#Norwegian-Law"/>
   </instanceOf>
   <resourceRef
         xlink:href="http://www.lovdata.no/all/nl-20000414-031.html"
   />
</occurrence>
```

Associations

Relations between topics are expressed in *associations*. Topic map associations can have types and association roles. In real life information relationships are often both complex and consist of multiple different relations between each other. Associations can express such relationships in regards to both sides, and are always bi-directional. For example a set of Articles comprise a Law, and a Law is comprised by Articles. The relationships between these articles and the law are from the articles is a typical "supertype-subtype"[56] association type. Associations can be given association roles, so this relationship Article should be given a subtype[57] identifier association role, and Law should be the supertype[58]. For example:

[56] http://psi.topicmaps.org/iso13250/model/supertype-subtype is a core standard defined published subject identifier that should be used as the association type ISO13250, chapter 7-3.

[57] i.e. http://psi.topicmaps.org/iso13250/model/subtype

[58] i.e. http://psi.topicmaps.org/iso13250/model/supertype

```
<association>
 <instanceOf>
    <topicRef
     xlink:href="http://psi.topicmaps.org/iso13250/model/supertype-subtype"
    />
  </instanceOf>
 <member>
  <roleSpec>
   <topicRef
    xlink:href="http://psi.topicmaps.org/iso13250/model/supertype"
   />
   </roleSpec>
  <topicRef xlink:href="#Law"/>
 </member>
 <member>
  <roleSpec>
   <topicRef
     xlink:href="http://psi.topicmaps.org/iso13250/model/subtype"
    />
  </roleSpec>
  <topicRef xlink:href="#Article"/>
 </member>
</association>
```

Scope

Topic names, associations and occurrences can also be given a *scope*. A scope can be used to add a specific perspective, view, language or semantic context (etc). Topic map scopes can also be used as access level, for instance can information about who edited a page, when and where be represented as scoped occurrences for a given topic. The system can then determine that only administrative users are allowed to see this scope (or view) of information, or two different topics, with the same name: *Oslo* can either be about *the city* or (Oslo) *the province*.

Scopes can be used to separate meaning of similar subjects and occurrences, for example: legal-terminology and language includes several *homonyms* (or *polysemes*) – words with two- or more meanings, and often context-specific. These are particularly problematic to both laypersons and legal practitioners, especially when used in subject identification, such as through words in a search query. By annotating such words with scope, one can separate these contexts. The classical example of a legal homonym is *aggravation.* In general language it means to worsen, complicate or provoke, but in legal context it means for instance “a high probability to cause death” in the context of robbery, arson and assault, and “a considerably strong culpability” in the context of fraud, fault and insurance. As such one could annotate the subject with scope: “aggravated (scope: arson)” using topic map scopes, or even “aggravated (scope: layperson)”.

Scopes are some times referred to as *namespaces*. A namespace is defined as a set of names that represents an object. For instance if you are in the page Oslo in scope *province,* the field neighbour could refer to Oslo's neighbouring provinces, and not Oslo's neighbouring cities.

Scopes are also used for typical configuration management such as "Localization" (L10N) and "Internationalization" (I18N). In these contexts topic maps are often used to filter information based on the language, number format and currency scopes, and so on. For example the abbreviated base-name in the topic above can be given the scope "Norwegian" which presumably is a topic that describes what Norwegian is, or used to only show to Norwegian visitors.

```
<baseName>
  <scope><topicRef xlink:href="#Norwegian"/></scope>
  <baseNameString>
    Lov om behandling av personopplysninger
  </baseNameString>
</baseName>
```

Topic Maps Merging

Topic Maps technology is designed to be mergeable. To be able to merge smaller topic maps as well as single topics of the same matter (or about the same resource) is a considerable strength. There are several rules for merging topic maps:

Subject Naming constraint

The naming-constraint merge rule states that "topics that share a base name in the same scope are the same". This means that if two maps both hold a topic with the base-name *Oslo* (in the scope *city*) these topics can be merged into one topic. This is not a particularly strong rule, and can easily result in erroneous topics. The safer way is through adding subject identity to a topic.

Subject Identity

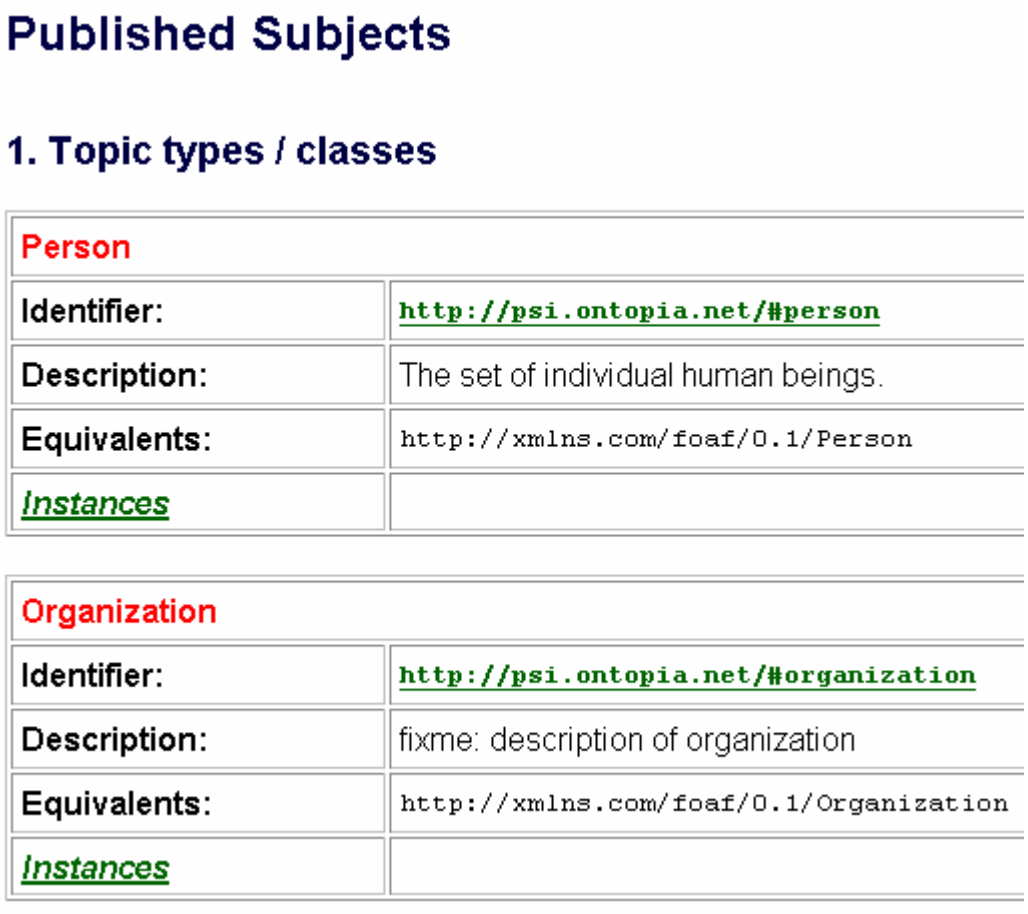
Subject identity is a central part of the topic maps theory. Any topic can be given subject identity. The subject identity merging rule states that "topics that share a subject identity are the same".

In other words, topics that have the same subject identity are of the same matter, and are about the same resource. As an example, subject identity of two topics is set to the URI: "http://juristopia.no/Privacy_act", and two topics, "inspection" and "duty to report" both have this URI as their subject identity they are considered topics of the same matter.

Subject identity does not need to be a URI. It can for instance be another topic, or even a topic in another map, or a PSI (Published Subject Identifier).

Published Subject Identifiers

A PSI is (most often) a URI published along with an explanation of what it is supposed to identify. Published Subject Identifiers are essential in when merging branch, field or industry specific topic maps. Figure 5-4 illustrates two PSIs made available by Ontopia.



Published Subjects

1. Topic types / classes

Person	
Identifier:	http://psi.ontopia.net/#person
Description:	The set of individual human beings.
Equivalents:	http://xmlns.com/foaf/0.1/Person
Instances	

Organization	
Identifier:	http://psi.ontopia.net/#organization
Description:	fixme: description of organization
Equivalents:	http://xmlns.com/foaf/0.1/Organization
Instances	

Figure 5-4 Published Subject Identifiers

Generic subjects can be created and published for the use of entire industries, through common bases as suggested in 2001 by Steve Pepper: "to create a Norwegian, open and democratic

collective knowledge base"[59]. Pepper argued that Topic Maps technology could be used, and any and everyone could write their own information, and link them together in a collective map using a set of centrally Published Subjects.

Figure 5-5 illustrates merging of smaller topic maps (1) that describe several resources (2) that use of the same published subjects (3), then merge in the topic maps engine to produce a big topic map (5) that has links to all three resources (2):

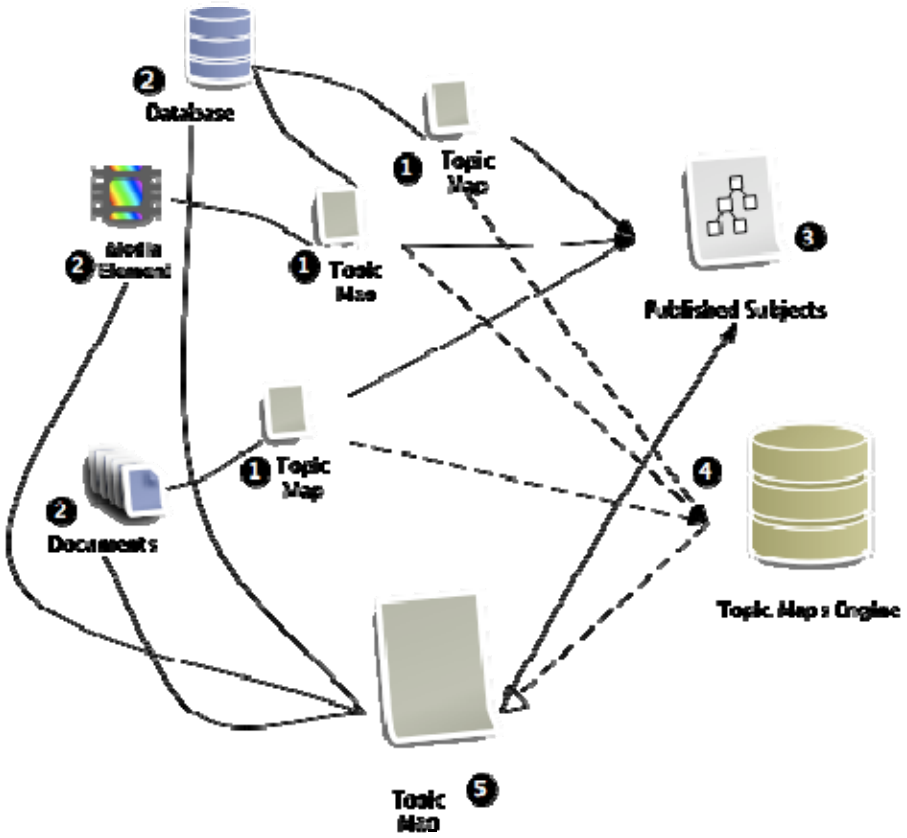


Figure 5-5 Topic Maps Merging

In legal information systems, we have one particular candidate that should consider publishing subjects: Lovdata, the main Norwegian source of legal information. If Lovdata created a base of PSIs for all their laws, these could easily be used as subject identities and enable lawyers to merge information resources about particular laws and paragraphs, such as in legal commentary. Published Subjects could be created for special fields as well, for instance for concepts in fields of law, legal entities, and so on. Anyone can publish their subjects; the main problem is to make people use the same PSIs.

[59] Chronicle, only in Norwegian: http://www.aftenposten.no/meninger/kronikker/article242930.ece

Using URIs as subject identity is the simplest way to achieve compliance with other classification schemes like RDF. It is also in accordance with the third goal of the W3C for the Semantic Web as mentioned before. In a most Wikis this is a feasible alternative to a Published Subject base, as pages are commonly referenced through a unique URI, though not all qualify to do this, for instance JSPWiki use id parameters to view pages i.e. “http://example.com/wiki.jsp?id=Page”.

Topic Maps Constraint and Query Languages

Two standards that are still in progress are Topic Map Constraint Language (TMCL) and Topic Map Query Language (TMQL). These are to be used to constraint topic maps, and to add a simple interface to query topic map engines. Constraining and querying are important features of topic map engines, in order to limit semantic searches and finding only the information wished for.

Topic Map Constraint Language

TMCL is intended to be to TM what OWL and RDFS is to RDF (Garshol, 2003). Unlike RDF, TMCL’s basis is in constraints-based (description) logics instead of descriptive logics, through the use of validations, exceptions and constraints on the entire TM Data Model (TMDM) (except identifiers and topic names) as well as merging of schemes (Moore et al., 2005). This is arguably a better approach to describing vast information resources, as it is based on describing things that can be constrained, and using these in the process to create well defined semantics for a topic map, and at the same time be used as a topic map by itself (useful for core ontologies, and general concepts).

TMCL will introduce a schema format to define class level constraints, map level constraints and enable map validation. Some examples of what these include are: for instance the possibility to demand a name for a topic, or demand at least two members of the “sensor” type for an association of the type “master thesis”. With map validation one can validate if sources in the topic map are invalid in respect to given constraints in a schema.

This is still a work in progress, and there are currently a couple of prototype constraint languages, including AsTMa! for the AsTMa= authoring syntax, and LTM has its own variant as well as experiments with extending the XTM syntax (Garshol, 2003).

Topic Map Query Language

TMQL is unsurprisingly a language in draft to query topic maps engines. It will of course be fully compatible with TMCL, and able to take advantage of the constraints. Like TMCL this is still in draft[60], and has numerous of prototype implementations, including AsTMa?[61] (for the AsTMa* family), tolog[62] which is a prolog-like and SQL influenced query language by Ontopia , and XTMPath for XTM a prototype of an XPath-inspired query style.

[60] http://www.isotopicmaps.org/tmql/

[61] http://astma.it.bond.edu.au/astma%3F-spec.dbk

[62] http://www.ontopia.net/topicmaps/materials/tolog-spec.html

Part IV

Prototype Implementation

6. Juristopia: Semantic Wiki

Along with this thesis I have developed several conceptual systems, ranging from parsing simply formatted Norwegian legal texts to XML, experimented with different topic map engines and tools, such as Ontopia's Omnigator[63], Robert Barta's Perl-based XTM module[64], the open source TM4J[65] (Topic Maps for Java) implementation and Lars Heuer's Python-based Mappa topic map implementation[66].

Along the road I realised that most of my systems were too big for a master thesis, and instead I wanted to do something feasible, yet still interesting enough that I could learn from it. My first meeting with topic maps technology was in the Internet Technology classes at Bond University, Australia, where Professor Robert Barta was lecturing. I grew interested in topic maps, and have since wanted to do a master thesis based on this technology. Topic Maps have evolved a great deal since then, and is now a viable (and possibly the best) technology for solving the needs of subject-oriented indexing.

6.1 Semantic Wiki

Semantic Wikis extend Wiki technology as earlier described in chapter 3. There is no set definition as to what is a Semantic Wiki, but they are all based on the vision of the Semantic Web. The best attempt I've seen so far is SemWiki.Org definition of Semantic Wiki (SemWiki.org, 2006):

> *Semantic Wikis try to combine the strengths of Semantic Web (machine processable, data integration, complex queries) and Wiki (easy to use and contribute, strongly interconnected, collaborativeness) technologies.*

I think this definition is simple enough, and thus in terms with Wiki's inventor Ward Cunningham's views.

[63] http://www.ontopia.net/solutions/omnigator.html

[64] http://search.cpan.org/dist/XTM/

[65] http://www.tm4j.org/

[66] http://code.google.com/p/mappa/

Other definitions are often in terms with technology choices, like “a Wiki using RDF and an ontology written with OWL”, and so on. The basis of a Semantic Wiki in my terms is any Wiki with an underlying knowledge representation layer that describes the information available.

6.2 The User Perspective

During this thesis I have interviewed and had talks with knowledge managers of four of Oslo’s biggest Law firms, and lawyers in a number of smaller practises. With a few exceptions Norwegian law firms do not have a strong focus on knowledge based systems. Why this is – is hard to say, I did sense some organisational disdain about information technology in general, maybe that may be a reason. However that was not my intent to solve, and the interviews I held were focused on extracting as much relevant information and ideas for Juristopia as I possible. So to speak I did have an agenda with the interviews; to form some goals for my implementation.

The four Norwegian law firms I did interview do have good knowledge management strategies; the exemplary being Thommessen Krefting Greve Lund AS (TKGL). They were the first to establish a knowledge management programme, and are still miles ahead of the others. All the knowledge managers interviewed seemed enthusiastic about their challenges, in spite of the (socio-) technological barriers.

The knowledge systems employed were all document management based systems; some had implemented separate knowledge systems that held selected and reviewed resources, while others deemed their entire portfolio of information knowledge. I identified close to Wiki collaborations in some instances, but then with a strong focus on document ownership and review from senior lawyers such as team-leaders of legal fields and partners.

The five most important demands I identified included:

1. Good indexing, being able to find the right information quickly. None of the interviewees were satisfied with the current quality of their searching engines.

2. They all wished to be able to measure the time spent on creating and editing documents, and by whom.

3. Registration of use was very important, preferably along with a quality ranking. The main purpose was to be able to see which documents were more interesting and popular and thus more important for periodical review.

4. Being able to create packages of related documents.

5. Being able to send comments to the author(s) on elements in the information.

I have taken these five demands into consideration for my Juristopia implementation.

6.3 Juristopia Implementation

Juristopia is a *prototype* of a Semantic Wiki system. I stress that it is a prototype, because it is still under development, the quality of the code is dubious, and the design is not stable. I haven't yet decided whether to release it as open source or not, that depends on whether I find it decent enough for release, and the software design stable enough for allowing others to collaborate. The daily build of Juristopia is available at http://juristopia.no and source code available (as of writing this) through Subversion at: svn://juristopia.no/jt.

Juristopia is developed with the Ruby on Rails web-framework, an agile web-framework based on the Ruby programming language. Ruby is a high-level interpreted language known and popular for its elegant syntax and object oriented approach. The Rails framework is based on the MVC[67] (Model View Controller) software design pattern, which allows (intuitive) separation of the presentation of the data, what the user/computer sees (the view), the operation and business logics of the system (the controller), and the data storage (the model). The Ruby programming language boasts a rich set of plug-in libraries called RubyGems; I make use of several of these (including rails) in the project. The Rails framework also supports a number of *plug-ins* and *generators* - plug-ins that assist in creating code.

[67] MVC is a software design pattern which was invented by Tryve Reenskaug in 1979, then working at Xerox Parc, USA.

6.4 Ruby on Rails framework

With Rails creating the skeleton model is a trivial process, you have generators that create the specs, models, database migrations (schemas) and everything else needed for you with simple commands:

```
$ script/generate rspec_model TM/Topic
      create  app/models/tm
      create  spec/models/tm
      create  spec/fixtures/tm
      create  app/models/tm/topic.rb
      create  spec/fixtures/tm/tm_topics.yml
      create  spec/models/tm/topic_spec.rb
      exists  db/migrate
      create  db/migrate/002_create_tm_topics.rb
```

The Rails ActiveRecord gem has a number of interesting relationship mappings between database model and object model allowing one to create dependencies for objects and tables (has_many, belongs_to, has_many_and_belongs_to, etc). I make use of these relationship mappings when creating the object models:

```
class TM::Topic < ActiveRecord::Base
  belongs_to :reifier, :polymorphic => true
  has_many :names, :class_name => 'TopicName',
        :foreign_key => 'parent_id', :dependent => :destroy
  has_many :scoped_names, :through => :names, :conditions =>
        "names.scope != 0"
  has_many :occurrences, :foreign_key => 'parent_id', :dependent =>
         :destroy
  has_many :scoped_occurrences, :through => :occurrences, :conditions =>
        "occurrences.scope != 0"
  has_many :associations  :class_name => 'AssociationRole',
        :foreign_key => 'parent_id', :dependent => :destroy
  . . .
end
```

Another example is for instance the TopicName model, to map that it belongs to the topic and has a number of name variants:

```
class TM::TopicName < ActiveRecord::Base
   belongs_to :topic
   has_many :variants, :foreign_key => 'parent_id', :dependent => :destroy
   . . .
end
```

Juristopia has two controllers determining the flow between the models. First is the User controller that allows people to log in, out, register and change password, then there's the Wiki that controls editing of pages and user access (by filtering if a user is logged in or not) and does all interaction with the pages and the topic map data model (through the TM::TopicMap library).

I have also attempted to extract the topic maps engine out of my code and into a generalised library, such as an "acts_as_topic" plug-in, but I this process is still in working. The rationale behind a plug-in is mainly to introduce others to use my implementation, and hopefully get some involvement into improving it.

6.5 Juristopia Architecture

6.5.1 General Decisions

The primary goal is that the system shall be simple to use, so convenience and simplicity will be weighted heavier than flexibility at any dispute. The second primary goal is that the system should be flexible and simple to work with externally; I have therefore selected to use XHTML for view representation for web browser clients, XML for web services, and XTM for exporting topic maps data.

Juristopia shall be open source software, and make use of available standards where possible. The over all architecture shall be simple and easy to understand by other programmers as well as interoperable with other services and clients on the web.

The Rails framework support Representational State Transfer (REST) natively, and thus, the overall application architecture will be based on REST. REST architecture is a simple (web) application architecture that allows *resources* to be accessed and modified by use of HTTP[68]

[68] HyperText Transfer Protocol is the standard protocol used on the WWW to transfer HyperText.

protocol verbs GET, POST, PUT and DELETE, and does not require state to be preserved or additional parameters to be sent in order to access content through URIs, solving a big challenge on the Web (Fielding, 2000). REST furthermore makes it simple to interoperate with several different forms of clients. To make a method respond differently and serve different content, whether it is a XML web service asking, a web-browser or a topic map browser this can be done trivially with Rails with simple Ruby code:

```
def show
  @wiki = Wiki.find(params[:id])
  respond_to do |format|
    format.html
    format.xml  { render :xml => @wiki.to_xtm }
  end
end
```

When a request comes in for 'http://juristopia.no/Page', this will be served with html format, while if the request is for '/Page.xml', the framework will use corresponding ':xml' format for the data gathered from the page's 'to_xtm' method.

With all agile development you start by giving a loose description of what the system is to entail; the project goal. In my project it is simple: "A Wiki with User authentication and a Topic Maps engine to store metadata and navigation information, as well as to query". In most situations you will typically have a client and you together you start creating user stories (and expected behaviour of the system).

As mentioned in the methods chapter, I use a form of Test-Driven Development, called Behaviour Driven Development (BDD) in the development. In Ruby there is a library called rspec[69] that makes writing specifications (henceforth: spec) simple. Underneath is an example for a spec that sets up the controller, and checks if a user is added (count incremented by one), and also that the user is redirected to his own page upon registration:

[69] http://www.rspec.org/

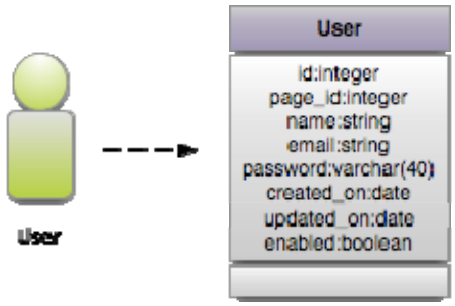
```
context "The Users Controller on Signup" do
  controller_name :users

  specify "should allow to register" do
    lambda do
      create_user :name => “John Doe”
      response.should redirect_to(“http://juristopia.no/wiki/John_Doe”)
    end.should change(User, :count).by(1)
  end
end
```

6.5.2 User System

First off, I’ve implemented a very simple user registration system. I won’t go into detail how I’ve done this, as this is a pretty straight-forward common practise that doesn’t need any particular focus. What is important to mention here is that the users’ model stores simple information about the user and attaches the user to a Wiki page, and thereby includes the user in the knowledge structure.

Figure 6-1 User model



The user system supports registration using a simple Turing-test verification, and features to retrieve lost or forgotten passwords by e-mail. I thought about using OpenID here, and might add support for that at a later time. All users are editors in the system, but in the future I might implement an Access Control List based system to limit and grant different levels of access for pages.

6.5.3 Wiki Pages

The main feature of the system is the Wiki. I have implemented this as a RESTful[70] web service that serves pages as XHTML, XML, and even XTM Topic Map fragments. Most Wikis by default uses a REST-like architecture, and by enforcing a full RESTful implementation, other clients than web-browsers, such as web services and search engine spiders may access the data

[70] A system is considered RESTful when it conforms to the Representational State Transfer (REST) architecture.

in well-defined manners, and as such integrate with the system. This means that pages are accessible through the same URI, only using different HTTP verbs: reading (GET), creating (POST), updating (PUT) or removing the page (DELETE) all use the same URI, for instance:

http://juristopia.no/SomePage

The significance of such a design comes is that the URI is the single parameter needed to manoeuvre the resource, and the state is determined by the HTTP verb.

The pages model consists of simple elements:

- Unique title, which is also used to create the URI, and acts as the primary external identifier of a page. All Rails models use the built-in 'id' attribute to refer to instances internally.
- A 'body' text field, which holds the text inputted by the user - the Wiki structured text.
- In the initial release I have a 'fragment' field which holds the topic map and rules used by the navigation panel. This field will probably be phased out eventually as the Topic Maps engine evolves and is smart enough to determine this view by itself. (This has later been phased somewhat out).
- A 'sidebar' text field is used for holding the text viewed in the sidebar, such as page specific semantic queries.

Version Control

Each page is *version controlled* through the acts as versioned plug-in[71] that creates a simple version controlled schema for a model. This is simply done by adding the definition to the model file like:

```
class Page < ActiveRecord::Base
   acts_as_versioned
   non_versioned_fields.push 'title'
   . . .
end
```

[71] http://wiki.rubyonrails.org/rails/pages/ActsAsVersioned

I have decided to not version control the page title, as this is used both to generate the URI to the page as well as the URI base name of the topic. By doing it this way one can change the base name of a page without needing to re-reference everything in the database and the topic map.

Pages Acts like Topics

The simple structure of the page model makes it easy to integrate it directly into the topic map by making the page itself act like a topic in the system, thereby being directly connected to the knowledge representation layer. Pages are used for all sorts of topics as I will later go into more detail about. To achieve this integration, I have written a simplified Topic Map engine that enables one to specify models to be considered as topics. The topic map functionality is included into the model when the special library is loaded.

6.5.4 Topic Maps Engine

The Topic Maps engine is one of the most difficult efforts of Juristopia. Mostly because of there are so many ways to integrate the TMDM standard, especially in a flexible language like Rails. I opted for a simplistic version, and limited some of the heavier features in the TMDM standard. It should be noted that I experimented a lot with Python-based Mappa and Java-based TM4J and Omnigator topic map engines, and was tempted to use one of these for the integration. The main reason I chose to instead create a minimal Topic Maps engine is that if topic maps as a means to create Semantic Web applications are to be adopted by the Rails movement, they will need a light-weight, minimal-configuration engine.

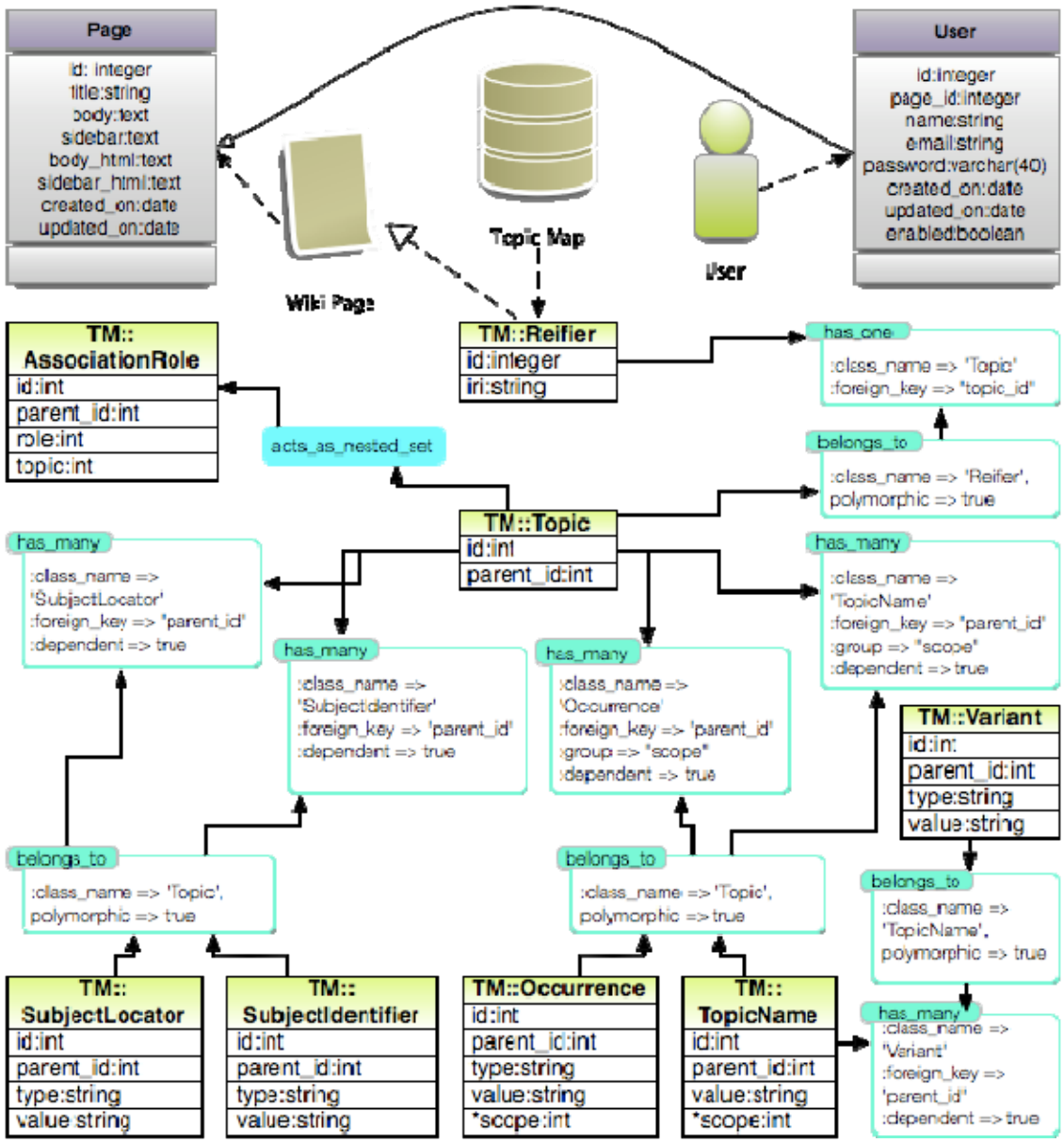


Figure 6-2 The Juristopia Data and Topic Maps Model.

The topic map model as illustrated in Figure 6-2 is a simplified implementation from the Topic Maps Data (Garshol and Moore, 2006). Scope is implemented as optional references to topic (by id). The TM engine is primarily used to assist the pages and controls the entire findability process of the system, as well as creating a meta-structure of the Wiki's content.

Associations

The Association Role uses a nice hack, through using a nested set data structure I am able to map associations and association members of these recursively in an optimised manner. The ambiguous part of it is the field named "topic", this has a dual meaning: In the context of the first

association parent object it can be used to define a scope, whereas in child objects it is used to as the topic reference of an association member.

Scopes

Scopes are implemented as statements (topics) in this model, and all topic references are identified using the internal Rails id field, which is vastly quicker than using (string) name translations. I have created the model on the assumption that (almost) everything is a statement (topic). All though scope support is minimal, it is also flexible. Topic names, associations and occurrences have a default '0' property on scope, and it is hence an optional value, this can be solved easily by adding this to the database migration:

```
$ vi  db/migrate/008_create_tm_base_names.rb
. . .
    create_table :tm_topic_names do |t|
      t.column :parent_id, :integer
      t.column :value, :string
      t.column :scope, :integer, :default => 0
    end
. . .
```

Subject Identity

Subject Identifiers and Subject Locators can be added to any topic, and are easily accessed from the reifier's topic property. Both tables are identical in structure and have options for types and values as strings. Subject indicators like Identifiers and Locators are not considered statements, and thus I have solved their implementation by using two string variables: *type* and *value*. The type field is introduced to be able to specify that the value is of a type such as a 'URI', 'pdf' or 'illustration', etc. If there is no type, the engine will assume it is an IRI[72] (Internationalized Resource Indicator) (URI encoded in UTF-8[73]). The topic maps engine will assume by default that any subject identity is a IRI, hence the default string type of value. Subject Locator support are added and will mainly be used in the system as link redirections to the main subject identifier, which is an IRI for the reified page.

[72] http://www.w3.org/2001/XMLSchema#anyURI

[73] http://www.ietf.org/rfc/rfc3987.txt

Reification

The reification object is in essence the Wiki Page model, and merged into the Wiki pages in the system. The integration of the topic map and the Wiki Page mode have been experimental both as a mix-in[74] and as a superclass of the Page model, in the end a mix-in was the most convenient solution. The reification object also includes an 'iri' string field to hold the main IRI of the system for simple access and redirection purposes in for instance cases when a valid Subject Locator is hit by the Wiki controller.

Names and Variants

The topic maps engine has support for names, there is no implementation of variants of names yet, but the model is created, and the implementation should be feasible. I was unsure if the Topic Map - Data Model is supposed to even add scoping support for name variants, which is the main reason this functionality is missing.

6.5.5 Authoring the Topic Map

The Topic Map library, TM::TopicMap has simple methods for controlling the topic maps model, the API[75] is simple:

[74] Ruby can "mix-in" entire classes, meaning that it merges two objects and inherits all methods and properties of the mixed-in class similar as if it was to inherit the class. Page by default inherits ActiveRecord::Base, and multiple inheritance is only available in Ruby by the use of mix-ins or through class evaluation (not experimented with).

[75] API is Application Programming Interface, a model of which methods and functions a library supports.

```
# Create a new blank topic map with an optional namespace
tm = TM::TopicMap.create( :namespace => 'http://juristopia.no/' )

# Creates a new named topic in the topic map
topic = tm.create_topic()

# Add a name to a topic
topic.add_name(:value => "Law", :scope => @scope)

# Add a Subject identifier to a topic
topic.add_identitfier(:type => "URI", :value => "http://uri/Resource")

# Creates a new association  for a topic with a role and scope
assoc = topic.create_association(:role => @role, :scope => @scope)

# Add a association member to a topic association
assoc.add_role(:role => @role_topic, :topic => @topic)

# Create a new occurrence for a topic
topic.create_occurrence(:type => "URI", :value => 'http://lovdata.no',
            :scope => @scope)
```

A requirement of this topic map implementation is that users shall be able to edit topic maps from inside of Wiki pages, and doing so in a simple and intuitive matter.

I have approached this by creating a small set of in-line codes to author semantic content.

6.5.6 Inline Topic Map Editing

The Wiki engine processes the page content after creation or updates. This process converts the text to HTML, recognises WikiWords, creates hyperlinks and extracts the in-line semantics.

Data about information resources (occurrences), their relations (associations), names and such can easily be mapped into the text with a simple and intuitive syntax.

There is no support for scope in this inline syntax.

Subject Identifiers

The primary organisation feature of Wiki pages is the subject identifiers. Juristopia supports simple means of allowing one to annotate topics with simple subjects such as "is-a <category>" and "instance-of <type>" and "supertype-of <type", etc. This is done in Juristopia with subject identifiers objects. In the view these can simply be annotated by using curly-bracket syntax:

```
Typed: {{TopicTypeOrAssociationType Topic}} or unttyped: {{Topic}}
```

Examples:

{{InstanceOf Law}}

{{FieldOfLaw PrivacyLaw}}

Associations and Hyperlinks

Hyperlinks between Wiki pages are treated as very simple associations in the Wiki. There are three main ways of creating in-line associations, ranging from standard WikiWords, to more advanced relationships.

Simple Format

Whenever the processing finds a WikiWords, which is any word matching a simple regular expression My interpretation of what constitutes Wiki Words is not only limited to CamelCase words, but also words containing digits, and special characters '_' and '-', and can even parenthesis (however only in consecutive words).

```
PageName regular expression:
```

/^([A-Z][0-9A-Za-z_-]+([A-Z][0-9A-Za-z()_-]+)*)$/

Examples:

"Wiki_Word", "WikiWord_(CamelCase)", "Plan9_From_Outer_Space", "Plan9-from-Outer-Space", etc.

Simple Bracketed Format

Word within single brackets: '[' and ']' are deemed a links to Wiki pages:

```
[ PageNames ] regular expression: /^[(\w|[-_])+]$/
```

Examples:

"[topic]", "[field-of-law]", "[is-a]"

Advanced Bracketed Format

Fully fletched associations can also be mapped with inline syntax, these are identified by phrases within double brackets: '[[' and ']]'. These phrases are full topic map associations, and consist of starting with the association type, and then all the members, as pairs separated by ':'.

```
[[association-role member-role:member-topic  member-role:member-topic
      member-role:member-topic  … ]]
```

Examples

"[[author written-by:SomePerson wrote:SomePage]]" and

"[[super-subtype supertype:law subtype:article]]".

Special Relationships

There are some special generic cases of topic map relationships, and I have created parsers for *supertype-subtype* and *related term* (as earlier described with thesauri).

```
Supertype-subtype:    [[> supertype subtype]]
Related term:         [[= firstterm secondterm]]:
```

Examples

"[[> law article]]" and

"[[= car motor-vehicle]]"

The simple formats create associations of type "PageLink", with member association roles "PageLinker" and "PageLinkee" between the two pages. The advanced links allows users to specify the association type and association member types.

I have not added support to associate external entities (through IRI), but that might be a future feature. Currently all external information resources are treated as associations.

Occurrences

There are two main forms of occurrences:

URIs / IRIs

Stand-alone IRI strings starting with "http://" are considered occurrences of type IRI (http://www.w3.org/2001/XMLSchema#anyURI).

Marked-up Occurrences

Elements (on the same line) enclosed with pipes ('|'), including two colon (':') separated strings are considered occurrences.

```
| type of occurrence : value of occurrence |
```

Examples

http://www.example.com/?id=2

http://juristopia.no/uri

|name: Ole Christian Rynning|

|date of birth: 1982-02-08|

|age: 25|

|homepage: http://oc.rynning.no/|

|a string: the quick brown fox jumps over the lazy dog|

The first string i.e. "date of birth" is considered the *type* of an occurrence, while the second string i.e. "1982-02-02" is considered the *value*. All occurrences of the latter format will be converted into definition lists (HTML '<dl>' tag), while URI occurrences will be converted to hyperlinks.

6.5.7 Built in Topic Types

Juristopia contains a few built in Wiki pages that have special meaning, these are: "SupertypeSubtype", "Supertype", "Subtype", "RelatedTerm", "BroaderNarrower", "BroaderThan", "NarrowerThan", "IsA", and "InstanceOf", etc.

These built in types can be used to create basic structure between pages. I do not have any constraints on use yet, and it I assume people will understand when to use related types like

"BroaderNarrower", "BroaderThan" and "NarrowerThan" in associations (roles). I consider the "IsA" and "InstanceOf" topics main building pieces to create structure, as they can easily be used with subject identities.

In-line Syntax Page Types

Pages can be given types either through associations using the instance-of or is-a predefined relationships

6.5.8 Querying the Topic Map

The API defines some simple query (retrieval) mechanisms

```
# Get all topics
topics = tm.topics()

# Get all associations
assocs = tm.associations()

# Get all associated topics to a given topic
topic.associations()

# Get associated topics of a given type (IRI)
topic.subject_identifiers(:filter => { :value => "IRI" })

# Get all topics in scope
topics = tm.topics(:scope => @scope)

# Find a topic by name
topic = tm.find_by_name(@name)

# Find by IRI (URI)
topic = tm.find_by_iri(@iri)
```

In-line Querying Syntax

Juristopia has currently only support for three simple queries:

Queries on Subject Identifiers

To perform simple queries on subject identifiers and subject locators, you use the '?' notation. This performs a query that returns the set of all topics that contain the IRI (in the topics subject

identifers), and lists the results as a bulleted list, by default it uses the built-in "InstanceOf" as the Subject Identifier type, but you can also specify one manually:

```
[[? WikiPage(SubjectIndicatorType) WikiPage ]]  or
[[? WikiPage ]]
```

Example:

[[?PrivacyLaw]] and [[? InstanceOf PrivacyLaw]] are the same.

Output: Bulleted list of all pages with which has defined {{InstanceOf PrivacyLaw}}

Queries on Association (Roles and Member-roles)

The second query type uses '=' notation. This performs a query that lists all the topics that have associations of the specified association type (or role type) and the value of the IRI or the WikiPage as the role type member:

```
[[= WikiPage(SubjectIndicatorType) WikiPage]]
[[= WikiPage(AssociationTypeOrRole) WikiPage]]
```

Queries on Both Subject Identifiers and Associations

The third query type ('&') retrieves all topics that have the type and corresponding value. This type searches both in the subject identifiers as well as in the association roles, on associations it return any association where the Member and the Association type, regardless of their adjacency:

```
[[& WikiPage(AssociationType)  WikiPage]]
```

For instance [[& WrittenBy SomeAuthor]] will match both subject identifiers and associations where SomeAuthor is mentioned in an association where WrittenBy is found (usually as the AssociationRole: Wrote).

With these queries Juristopia implements a very simple interface for information object (or page) linking and categorisation. Each page is treated as a Topic in a topic map, an all WikiWord hyperlinks form topic map associations.

6.6 User Interaction

Visual Navigation Application

The visual navigation application of Juristopia is based on a simple Adobe Flex[76] 2.0 application that reads a simple XML output describing the topics, associations and occurrences collected by a page.

Flex is based on XML and ActionScript 3.0 (a dialect of the ECMAScript standardised scripting language and similar to JavaScript). As my topic maps engine does not yet have a proper de-serialiser (export of topic map into text-format) I implemented a simple fixed format loosely based on the data model to express how a page can be drawn in XML, that is used by the navigation application (see appendix).

Figure 6-3 Navigation Panel for Visualisation of Pages

Once a page is loaded the navigation display should draw the

The navigation panel is still a fixed prototype that can at the moment only access one fixed flex HTTPService (XML document) (http://stage.juristopia.no/nav.xtm). The latest build of this service can be accessed on http://stage.juristopia.no/.

[76] Adobe Flex 2 is a cross-platform development framework for creating rich Internet applications. Flex enables you to create expressive, high-performance applications that run identically on all major browsers and operating systems: http://www.adobe.com/products/flex/

With the visual navigation application I wish to create a simple graphical view for any given page. Initially I have only added support to see the simple subject identifiers a given page has, but I have attempted to draw associations as well, and occurrences.

Editing

Editing a page is done simply by clicking "Edit" on an existing page or navigate to "NewPage" page to create a new page, both gives this interface:

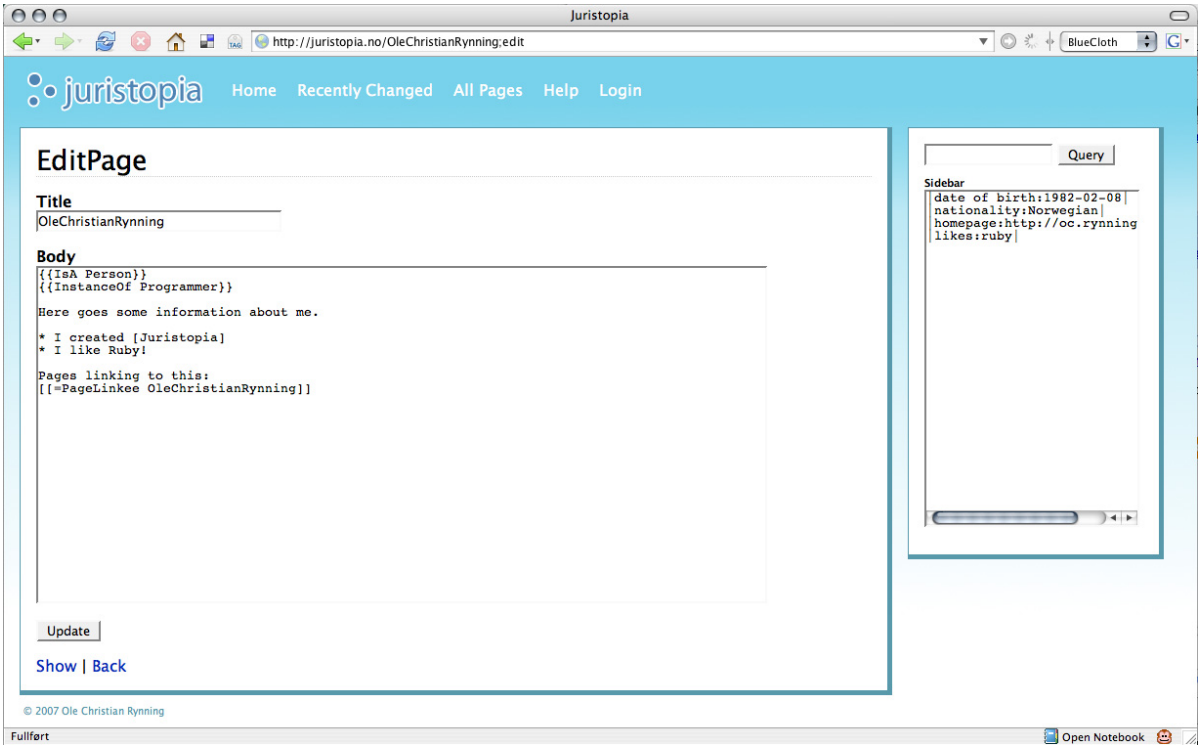


Figure 6-4 Editing a Wiki Page

When clicking update (or create in NewPage), the title is "Wikified" (spaces replaced with underlines '_', special characters escaped so they can be used as IRIs, and both the *body* and the *sidebar* processed. The topic maps *extract* method converts all of the aforementioned in-line syntaxes into topic map items, at the same time, the Wiki controller converts these and other simple markup (like two newlines means a new paragraph, 'h1.' ➔ '<h1>', etc, into HTML, and saves this "HTMLized" data in the *body_html* element.

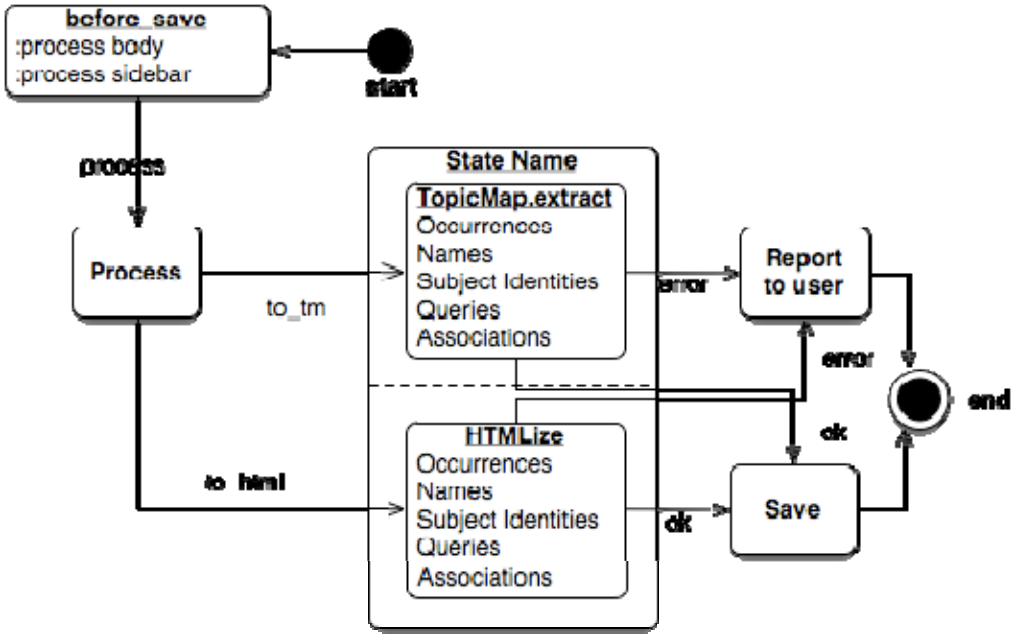


Figure 6-5 A UML state chart for the Wiki page save pre-processing

Viewing Pages

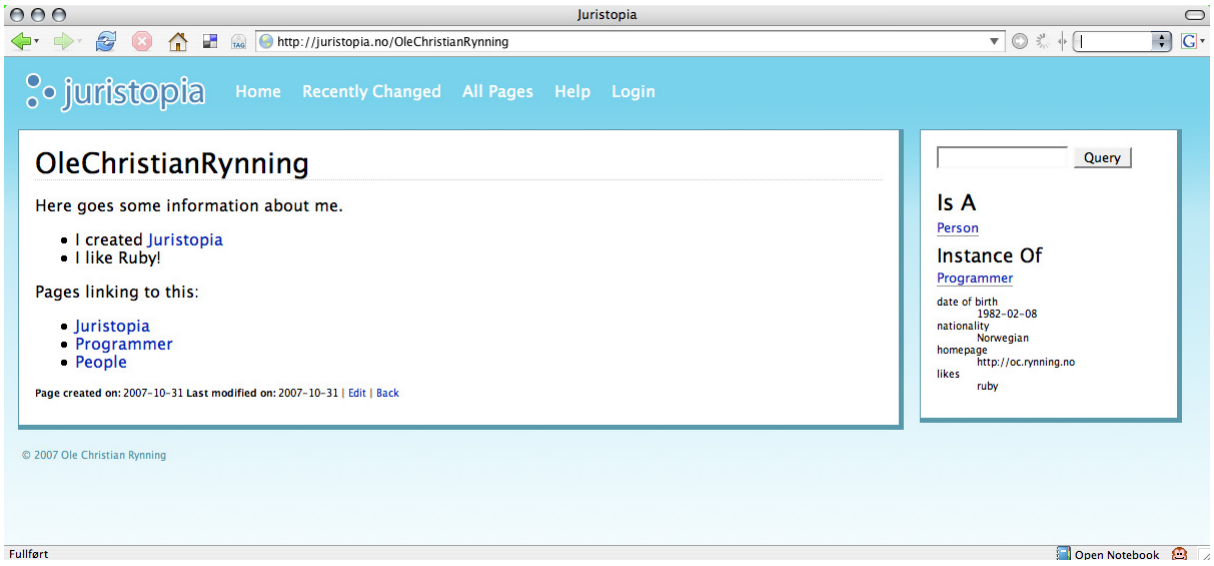


Figure 6-6 Viewing a Wiki Page with the Visual navigation panel hidden.

Editing Patterns

A simple pattern I have identified, is to use the sidebar for occurrence content, as well as queries, such as the [[=PageLinkee OleChristianRynning]] (pages linking to this), are well-placed in the sidebar.

As content grows, the Juristopia ontology will grow, and hopefully a lot of interesting patterns will show up.

Part V

The results and the road ahead

7. About The Contribution

Along the way of exploring the field, writing and implementing my prototype I have had some particular experiences and difficulties I'd like to mention.

7.1 Problems Encountered

The main problem encountered was that there until a couple of weeks ago were no proper lightweight Topic Maps engines available for free, and based on my technology choice I had to implement my own design. Doing this took a lot of effort away from what I really wanted to do initially: to explore different patterns for how to structure topic maps by the use of Wikis. While Ruby on Rails is a very popular framework, and has a great breadth of available libraries, it is still fairly new, and bugs and lack of documentation leads to spending too much time reading code of different standards.

While the Ruby language is very flexible, sometimes the flexibility is too much, and is one of the reasons making it hard to analyse code written in very varying styles. One of the weaknesses of Ruby and Rails is that error messages are not very intuitive and trivial problems and bugs may take long to identify, especially when the more esoteric features of the language and framework are used. Bugs such as a system fully working in my development environment on my laptop, but for some strange reason refuse to even start when deploying it to the server, only to find out that a certain corrupted string encoding library used is the reason.

As with all projects, time has been an important limiting factor. While most of the theoretical information was gathered and processed pretty early, my ambitions have been too big for what is feasible of a part-time student. Not being able to decide on a specific research topic until the very end, and changing my focus several times have also been factors reducing my focus on the particular system I ended up writing about.

7.2 Related Work

7.2.1 Ruby Topic Maps

On October 18th 2007 a new topic map engine, called RTM – Ruby Topic Maps - was released (Bock, 2007). I haven't had time to get really into it, but it seems like a great implementation as I scanned through it. Much of the code is similar to mine, but it is covering a lot more than my implementation. On the other hand its TMDM object model is not so Ruby-ist, and looks slow (a lot of mix-ins and unnecessary inheritance).

7.2.2 Related Work on Semantic Wikis

There are two main approaches to current Semantic Wiki research. The first is to create a Wiki that uses an underlying (internal) knowledge representation layer between all pages, and the other is to analyse existing Wikis, and create an (external) knowledge representation layer describing the resources.

There are already Semantic Wikis available, some are very conceptual, others are ready to use. I haven't done a comprehensive study to document all features or compare available (Semantic) Wiki technology, but I have analysed three of which I find most interesting: The *Semantic MediaWiki*[77] (and RDF) extension to MediaWiki, RDF-based *OntoWiki,* and my personal favourite *Wiksar*.

Semantic MediaWiki Extension

Semantic MediaWiki (SMW) is a project to turn MediaWiki engine into a Semantic Wiki. The MediaWiki engine, maintained by the Wikimedia Foundation, is used by sites including Wikipedia, Wiktionary, Wikiquote, Wikiversity, et.al. SMW introduces structured RDF annotations to existing Wiki pages, so that users can add fact boxes and well-structured RDF information boxes, as well as to query in these resources. (Millard et al., 2006).

Abbreviated Example of an Information box

[77] http://ontoworld.org/wiki/Semantic_MediaWiki

```
{{Extension
. . .
|download  = [http://svn.wikimedia.org/svnroot/mediawiki/trunk/extensions/RDF/
svn] ([http://svn.wikimedia.org/viewvc/mediawiki/trunk/extensions/RDF/ browse])
|readme    =
[http://svn.wikimedia.org/svnroot/mediawiki/trunk/extensions/RDF/README.RDF
.txt?view=markup README]
. . .
|description = flexible framework that goes beyond the ability of the code
described at [[m:RDF metadata|RDF metadata]]
|example   =[[WikiTravel:Wikitravel:RDF]]
}}
```

Extension:RDF is an extension that enables current information boxes to use SMW RDF. Information boxes are simple plug-ins using an own format for representing information in Wiki pages. With these info boxes one can easily create simple representations of information. Extension:RDF allows such information boxes to be export well-formed RDF triples that can be used by further SMW technologies to power the Wiki pages.

Figure 7-1 Example RDF Information Box

MediaWiki has since incorporated Extension:RDF into its code base, and efforts to keep up with SMW are now in planning.

OntoWiki

OntoWiki is a Semantic Wiki that builds on the same principles as SMW. It includes a WYSIWYG editor for inline editing of RDF-triples in Wiki pages, and querying functions inside of these RDF-data along with the regular set of Wiki features. OntoWiki allows sites to install ontologies written in OWL to add meaning to the RDF relationships (Auer et al., 2006). While OntoWiki tries to preserve the Wiki principles of being easy to use, and a portal for collaborative writing, it may be too advanced to qualify as a Wiki, and introduces a very big leap of technology for authors. The prototype is available on http://3ba.se/ but has been offline since June 21st 2007[78].

Wiksar

There is not so much information available about Wiksar, formerly known as SHAWN (Aumüller, 2005). Its Source Forge project page is empty[79], and there is little technical information available. However; there is an excellent prototype available at http://wiki.navigable.info/SomeThing. Wiksar is elegant, fast, easy to understand and easy to use. Wiksar is developed by David Aumüller and written in the Perl programming language, and uses a generic triple store to store relations between information such as usable by RDF, SPARQL and OWL (Aumüller and Auer, 2005). The Wiksar prototype holds information about English authors and strong beers as examples of how to classify and navigate information, and is extremely fun to play with. The syntax of Wiksar is based on in-line codes and WikiWords, making it easy to understand for plain Wiki editors, and at the same time gives great tools for classification. Wiksar also allows in-line queries to the knowledge representation layer, such as listing all pages in a certain category or concept. Navigation items such as breadcrumbs[80] make use of the relations a page has been given in an elegant manner. Aumüller also have a conceptual graph browser, based on the TouchGraph Wiki Visualizer Graph Engine[81] which aids in visualising the structure of the Wiki.

[78] http://web.archive.org/web/20070621141109/http://3ba.se/

[79] http://sourceforge.net/projects/wiksar

[80] Breadcrumbs are also known as navigation trails, and are usually placed at the top of a site, and shows an hierarchical path of the current location, “Home > People > R > Rynning, Ole Christian” is an instance of such a trail, and all of the predecessors of Rynning, Ole Christian (R, People, Home) are links that can be clicked.

[81] http://www.touchgraph.com/

7.3 Future Work and Improvements

7.3.1 Fix known problems

The Juristopia prototype is just that, a prototype. It contains a lot of bugs, and some features do not work at all. It is crucial though to fix some

7.3.2 Improve the Juristopia Framework

The Juristopia Topic Maps "engine" is not really to consider an engine as it lacks numerous important features according to the topic maps TMDM standard (Garshol and Moore, 2006). The Ruby Topic Maps library aforementioned seems like a viable source to analyse to get some improvement ideas.

Navigation Panel Integration

The navigation panel needs heavy work to be usable. Firstly I must figure a way to change its information resource dynamically based on the current page. Secondly the panel needs a mathematic node placement/balancing algorithm that both balances the map, and centres it on the currently chosen node.

Topic Map Serialisation / De-serialisation

A central piece lacking in Juristopia is a serialisation and de-serialisation of the topic maps into an interchange format such as AsTMa=, LTM or XML Topic Maps, as well as import of such is essential for creating a well thought through.

Topic Map Querying

Ideally the query language should still remain as simple in syntax [[?|=|&]], but it should also use a well-formed query-language for the backend, so as more advanced, and combinations of the simple syntax can be queried, i.e. [[? Norwegian & PrivacyLaw & CriminalLaw]]. The Topic Map engine should implement a language i.a. tolog, AsTMa?, etc.

Topic Map Merging

Another feature lacking in Juristopia is Topic Map merging. This is as explained above one of the central concepts of topic maps, and necessary for both integrating with other meta-

information bases, as well as consolidation of internal topics. At the time there are no techniques for optimising the internal structure, and enable

7.3.3 Examples for Use

The Juristopia system is still lacking representable content. Apart from some test pages I have used to test features like querying and in-line syntax, it is still not a good presentation for the system. I hope to be able to add such a content-base to the system in the coming days. The content-base should be a good collection that shows what kind of data should be added.

Topic Map Patterns

It is important for the adoption of such a system to establish an example information base that shows how the semantic bits and pieces, and information structure can benefit from the use of Topic Maps in a real life example. There are many interesting questions that can be researched and analysed here, especially in what are successful page patterns, which queries are popular in use, which topics are used for classifications, and so on.

8. Conclusion

This thesis has hopefully shown that Topic Maps can be used as a means for giving Wikis stronger structure. Essentially in terms of exposing the hidden structures in loosely structured information content.

I have gone through the theory leading to; and the Semantic Web. I hope that this thesis can be useful for people who will later research Topic Maps, and especially Topic Map-driven Wikis, and Wiki hyperstructures. I have introduced the concept of ontology-driven sites and explained what ontology is, but I have gracefully skipped the process of ontology creation

I feel I have given a thorough exploration report on my main goal: to explore how Semantic Web enabling-technologies and the reasons behind them, and how they can assist in creating a simple yet efficient system to inscribe legal information and meta-information. This second aspect also includes creating a well-sized examples base including some simple Topic Map Pattern examples for the users of the system. Juristopia which is the deliverable of this project will live on, and be improved on according to the points mentioned in the previous chapter.

This thesis has hopefully shown that Topic Maps can be used as a means for giving Wikis stronger structure. Essentially in terms of exposing the hidden structures in loosely structured information content.

Through the exploration I feel I have gained a good understanding of what technologies are available, and many problems and solutions using the technologies. While researching and implementing I have also gathered a lot of experience along the way about legal resources through the experiments. Whereas I am not quite satisfied with the this part is the socio-technical analysis of usage and how to create page structure, and information organisation, I think the system is capable of doing so, but I haven't had enough time to explore this aspect.

References

Anderson, N. (2006) *Congressional staffers edit boss's bio on Wikipedia,* [2007-10-25], last updated: 2006-01-30, Ars Technica<http://arstechnica.com/news.ars/post/20060130-6079.html>.

Auer, S., Dietzold, S. and Riechert, T. (2006) OntoWiki – A Tool for Social, Semantic Collaboration, Department of Computer and Information Science, University of Pennsylvania & Institut für Informatik, Universität Leipzig<http://www.informatik.uni-leipzig.de/~auer/publication/ontowiki.pdf>.

Aumüller, D. (2005) SHAWN: Structure Helps a Wiki Navigate, Department of Computer Science, University of Leipzig<http://the.navigable.info/2005/aumueller05shawn.pdf>.

Aumüller, D. and Auer, S. (2005) Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR, Department of Computer Science, University of Leipzig<http://www.semanticdesktop.org/xwiki/bin/download/Wiki/WikSAR/22_aumueller_semanticwikiexperience_final.pdf>.

Benjamins, V. R., et al. (2004) Ontologies of Professional Legal Knowledge as the Basis for Intelligent IT Support for Judges, *Artificial Intelligence and Law,* vol. 12, no. 4, pp. 359-378.

Berners-Lee, T. (2002) *What do HTTP URIs Identify?,* [2007-10-10], last updated: 2007-01-15<http://www.w3.org/DesignIssues/HTTP-URI.html>.

Berners-Lee, T. (2006) *Artificial Intelligence and the Semantic Web,* [2007-10-01]<http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html>.

Bing, J. (1982) *Rettslige kommunikasjonsprosesser,* Universitetsforlaget, Oslo.

Bing, J. (1984) *Handbook of Legal Information Retrieval,* [2006-10-10], Norwegian Research Center for Computers and Law, UiO, (Electronic Book HTML) <http://www.lovdata.no/litt/hand/hand-1991-0.html>.

Bing, J. (2003) *Copymarks: A suggestion for simple management of copyrighted material,* [2007-10-10], NRCCL<http://efn.no/copyright/copymarks.html>.

Bing, J. (2006) Building Cyberspace: A Brief History of Internet, UiO, Oslo, p. 26.

Blackburn, P., Benthem, J. v. and Wolter, F. (2004) Modal Logic Handbook<http://www.csc.liv.ac.uk/~frank/MLHandbook/>.

Bock, B. (2007) *RTM - Ruby Topic Maps,* [2007-10-30], last updated: 2007-10-26<http://rtm.rubyforge.org/>.

Bowker, G. C. and Star, S. L. (1999) *Sorting things out. Classification and its consequences,* MIT Press, Cambridge, Massachusetts.

Breuker, J., et al. (2007) OWL Ontology of Basic Legal Concepts (LKIF-Core), Estrella Project<http://www.estrellaproject.org/doc/D1.4-OWL-Ontology-of-Basic-Legal-Concepts.pdf>.

Brunell, M., et al. (1988) *X.EARN Report,* [2007-07-22]<http://www.nic.funet.fi/index/FUNET/history/heureka/xearn.txt>.

Bush, V. (1945) *As We May Think,* [2007-06-23], The Atlantic<http://www.theatlantic.com/doc/194507/bush>.

Chen, L. (2007) *Several colleges push to ban Wikipedia as resource,* [2007-10-25], last updated: 2007-03-28, Duke Chronicle<http://media.www.dukechronicle.com/media/storage/paper884/news/2007/03/28/News/Several.Colleges.Push.To.Ban.Wikipedia.As.Resource-2809247.shtml>.

Christiaens, S. (2006) Metadata Mechanisms: From Ontology to Folksonomy ... and Back, *Lecture Notes in Computer Science,* vol. 4277, pp. 199-207.

Cunningham, W. (2003) *Correspondence on the Ethymology of Wiki,* [2007-10-12], last updated: November 2003<http://c2.com/doc/etymology.html>.

DCMI (2006) *Dublin Core Metadata Element Set, Version 1.1,* [2007-10-11], Dublin Core Metadata Initiative<http://dublincore.org/documents/dces/>.
Dick, B. (1993) *You want to do an action research thesis?,* [2007-07-11]<http://www.scu.edu.au/schools/gcm/ar/art/arthesis.html>.
Dick, B. (1997) *Approaching an action research thesis: an overview,* [2007-07-11]<http://www.scu.edu.au/schools/gcm/ar/arp/phd.html>.
Fielding, R. T. (2000) Representational State Transfer (REST)<http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.
Floridi, L. (2005a) Is Semantic Information Meaningful Data?, *Philosophy and Phenomenological Research,* vol. LXX, no. 2, pp. 351-371.
Floridi, L. (2005b) *Semantic Conceptions of Information,* [2007-08-11]<http://plato.stanford.edu/entries/information-semantic/>.
Garshol, L. M. (2003) *Living with topic maps and RDF: Topic maps, RDF, DAML, OIL, OWL, TMCL,* [2007-10-15]<http://www.ontopia.net/topicmaps/materials/tmrdf.html>.
Garshol, L. M. (2004) *Metadata? Thesauri? Taxonomies? Topic Maps! Making sense of it all,* [2007-02-10], Ontopia<http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html>.
Garshol, L. M. and Moore, G. (2006) *Topic Maps - Data Model,* [2007-10-17], last updated: 2006-06-18, ISO/IEC JTC1/SC34<http://www.isotopicmaps.org/sam/sam-model/>.
Gómez-Pérez, A., Fernández-López, M. and Corcho, O. (2004) *Ontological Engineering,* Springer-Verlag London Ltd, Madrid, Spain.
Gorman, G. E. (2005) Is the wiki concept really so wonderful?, *Online Information Review*, Vol. 29, Emerald Group, Wellington, NZ, pp. 225-226 <http://www.emeraldinsight.com/Insight/viewPDF.jsp?Filename=html/Output/Published/EmeraldFullTextArticle/Pdf/2640290301.pdf>.
Gruenberg, L. (1992) *Facet Analysis,* [2007-09-11], attbi.com<http://www.asis.org/Conferences/Summit2002/Gruenberg.ppt>.
Guarino, N. and Giaretta, P. (1995) Ontologies and Knowledge Bases: Towards a Terminological Clarification, *Towards Very Large Knowledge Bases: KBKS'95*, no. Mars, pp. 25-32.
Hagerup, F. (1888) Den nye Retsvidenskab, *Tidsskrift for Retsvidenskab,* vol. 1888, no. 1, pp. 1-58.
Hannemyr, G. (2003) The Internet as Hyperbole, *The Information Society,* vol. 19, no. 2, pp. 111-121.
Hanseth, O. (2002) *Gateways - just as important as standards. How the Internet won the "religious war" about standards in Scandinavia.,* [2007-07-27]<http://www.ifi.uio.no/~oleha/Publications/nordunet.pdf>.
Herman, I. (2007) *W3C Semantic Web Frequently Asked Questions,* [2007-10-11], last updated: 2007-10-10, w3.org<http://www.w3.org/2001/sw/SW-FAQ>.
Hert, C. A. (2000) *Studies of Metadata Creation and Usage,* [2007-10-11], School of Information Studies, Syracuse University<http://www.fcsm.gov/01papers/Hert.pdf>.
ISO/TC46/SC4 (2003) *ISO Standard 15836-2003,* [2007-10-11], ISO<http://www.niso.org/international/SC4/n515.pdf>.
ITST (2004) Definition of Open Standards, National IT and Telecom Agency, Copenhagen<http://www.oio.dk/files/040622_Definition_of_open_standards.pdf>.
Kniberg, H. (2007) *Scrum and XP from the Trenches*, InfoQ, (Electronic Book PDF) <http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>.
Kwasnick, B. (1999) The role of classification in knowledge representation and discovery., *Library Trends,* vol. 1999, no. 48, pp. 22-47.
Lehmann, J., Breuker, J. and Brouwer, B. (2005) CAUSATIONT: Modeling Causation in AI&Law, *Lecture Notes in Artificial Intelligence,* vol. 3369, pp. 77-96.

Lehtisalo, K. (2005) *The History of NORDUnet. Twenty-Five Years of Networking Cooperation in the Nordic Countries.,* [2007-09-11]<http://www.nordu.net/history/TheHistoryOfNordunet_simple.pdf>.
Metawiki (2007) *List of Wikipedias,* [2007-10-14], last updated: 2007-10-14<http://meta.wikimedia.org/wiki/List_of_Wikipedias>.
Meyer, D. (2006) *The Truth Of Truthiness,* [2007-10-25], last updated: 2006-12-10, CBS News<http://www.cbsnews.com/stories/2006/12/12/opinion/meyer/main2250923.shtml>.
Millard, I., et al. (2006) Using a Semantic MediaWiki to Interact with a Knowledge Based Infrustructure, School of Electronics and Computer Science, University of Southampton<http://eprints.ecs.soton.ac.uk/12869/01/ekaw06.pdf>.
Moore, G., Bogachev, D. and Nishikawa, M. (2005) *Topic Maps Constraint Language,* [2007-10-12], last updated: 2005-2-12<http://www.isotopicmaps.org/tmcl/tmcl-2005-02-12.html>.
Nelson, T. (1981) *Literary Machines,* Mindful Press.
Nelson, T. (1987) *Computer Lib/Dream Machines,* Tempus Books, Redmond.
Nelson, T. (1999) *Ted Nelson's Computer Paradigm, Expressed as One-Liners* [2007-10-18], last updated: Jan 1999<http://xanadu.com.au/ted/TN/WRITINGS/TCOMPARADIGM/tedCompOneLiners.html>.
Nielsen, J. (1997) *How Users Read on the Web,* [2007-09-14], last updated: 1997-10-01<http://www.useit.com/alertbox/9710a.html>.
Nielsen, J. (2003) *Information Pollution,* [2007-09-25], last updated: 2003-08-11<http://www.useit.com/alertbox/20030811.html>.
NISO (2007) *NISO Standard Z39.85-2007,* [2007-10-11], NISO<NISO Standard Z39.85-2007>.
Park, J. and Hunting, S. (2003) *XML Topic Maps,* Pearson Education, Inc., Boston.
Pepper, S. and Moore, G. (2001) *XML Topic Maps (XTM) 1.0,* [2007-10-28], last updated: 2001-08-06<http://www.topicmaps.org/xtm/>.
Poppendieck, M. and Poppendieck, T. (2003) *Lean Software Development. An Agile Toolkit.,* Addison-Wesley.
Raz, J. (1980a) Austin's Theory of Legal System. In: *The Concept of a Legal System,* Oxford University Press.
Raz, J. (1980b) Kelsen's Theory of Legal System. In: *The Concept of a Legal System,* Oxford University Press.
Rosen, L. and Weil, M. (1997) *Technostress: Coping with Technology @Work, @Home, @Play,* John Wiley & Sons.
Sanger, L. (2004) *Why Wikipedia Must Jettison Its Anti-Elitism,* [2007-10-10]<http://www.kuro5hin.org/story/2004/12/30/142458/25>.
Schwaber, K. and Beedle, M. (2001) *Agile Software Development with Scrum,* Prentice Hall.
Schwartz, E. I. (1997) *The Father of the Web,* [2007-09-27], last updated: March 1997, wired.com<http://www.wired.com/wired/archive/5.03/ff_father.html>.
SemWiki.org (2006) *SemWiki.org - The Semantic Wiki Community,* [2007-10-10]<http://www.semwiki.org/>.
Shenk, D. (1998) *Data Smog: Surviving the Information Glut,* Harper, San Francisco.
Sullivan, D. (2007) *How Search Engines Work,* [2007-10-17], last updated: 2007-03-14, SearchEngineWatch.com<http://searchenginewatch.com/showPage.html?page=2168031>.
Sutor, B. (2006) *Interoperability vs. intraoperability: your open choice,* [2007-08-27]<http://sutor.com/newsite/blog-open/?p=1260>.

Uschold, M. and Jasper, R. (1999) A Framework for Understanding and Classifying Ontology Applications, *CEUR*, Vol. 18, CEUR Publications, Amsterdam, pp. 11.1-11.12 <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf>.

Valente, A. (1995) *Legal Knowledge Engineering: A Modelling Approach,* IOS Press, Amsterdam.

Valente, A. (2005) Types and Roles of Legal Ontologies, *Lecture Notes in Artificial Intelligence,* vol. 3369, no. 1, pp. 65-76.

Verhoeff, J., Goffman, W. and Belzer, J. (1961) Inefficiency of the use of Boolean functions for information retrieval systems, *Communications of the ACM,* vol. 4, no. 12, pp. 557-558.

Appendix A

All Juristopia Code can be found through subversion on svn://juristopia.no/jt between 2007-11-01 and 2007-12-01. Please contact me by email oc@rynning.no for access to the repository.

Page Model

```
class Page < ActiveRecord::Base
  belongs_to :topicmap, :class_name => 'TM::TopicMap'
  has_one :topic, :through => 'TM::Reifier'

  before_save { |p| p.title = escape(p.title) }
  before_save :extract_knowledge

  validates_uniqueness_of :title
  validates_format_of :title,
    :with => /^([0-9A-Za-z_-]+([0-9A-Za-z()_-]+)*)$/
  validates_presence_of :title

  acts_as_versioned
  non_versioned_fields.push 'title'

  def title=(t)
    write_attribute(:title, t ? t.strip.squeeze(' ') : t)
  end

  def iri(uri, local=false)
    return escape(uri) if local
    return ENV[SITE_DOMAIN] + escape(uri) # http://juristopia.no/+uri
  end

  def escape(s)
    s.gsub(/([^ a-zA-Z0-9_.-]+)/n) do |e|
      '%' + e.unpack('H2' * e.size).join('%').upcase
    end.tr(' ', '-')
  end

  private
    def extract_knowledge
      TM::Helpers::import(body)
      TM::Helpers::import(sidebar)

      body_html = TM::Helpers::parse(body).to_html
      sidebar_html = TM::Helpers::parse(sidebar).to_html
    end
  class << self
    def all
      self.find(:all, :conditions => ['builtin=?',false])
    end
  end
end
```

Navigation XML

```
<?xml version="1.0" encoding="utf-8"?>
<page id="http://juristopia.no/PrivacyAct">
  <topic id="PrivacyAct">
    <type>LawText</type>
    <baseName>Privacy Act</baseName>
    <subjectIdentity>
      <type>IsA</type>
      <value>PrivacyLaw</value>
    </subjectIdentity>
    <subjectIdentity>
      <type>InstanceOf</type>
      <value>LawText</value>
    </subjectIdentity>
    <occurrence>
      <type>ExternalLink</type>
      <value>http://www.lovdata.no/all/nl-20000414-031.html</value>
    </occurrence>
  </topic>
  <topic id="PrivacyLaw">
    <type>FieldOfLaw</type>
    <baseName>Privacy Law</baseName>
  </topic>
  <topic id="LawText">
    <type>LegalInformation</type>
    <baseName>Law Text</baseName>
  </topic>
</page>
```

Navigation MXML

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
width="800" height="250" xmlns:oc="components.*">
  <!-- The HTTPService should be generated from ['page'.xtm] -->
  <mx:HTTPService id="jtNav" url="http://stage.jurstopia.no/nav.xtm"
useProxy="false" xmlDecode="topicMapDecoder"/>
  <mx:Script>
    <![CDATA[
      import mx.controls.LinkButton;
      import mx.core.Container;
      import components.topic;

      class Node {
          var name:String;
          var uri:String;
          var x:int;
          var y:int;
          var text:String;

          private function generate():topic
            var g:topic = new topic();
            var l:LinkButton = new LinkButton();
            // Image
            g.id = name; g.x = x; g.y = y;
            // LinkButton
            l.id = 'L'+ name;
            l.label = name;
            l.addEventListener(MouseEvent.CLICK, function fn():void {
              showToolTip(name, text.slice(0,312), uri);
            });
            l.x = 25;
            l.y = 0;
            g.addChild(l);
            return e;
          }
          public function getComponent():topic {
            return generate();
          }
      }

      class Assoc {
          var role:String;
          var from:Topic;
          var to:Topic;
          public function getFX():int { return this.from.x; }
          public function getFY():int { return this.from.y; }
          public function getTX():int { return this.to.x; }
          public function getTY():int { return this.to.y; }
      }

      private function tDecoder(nav:XMLDocument):void {
        var tm:XML = XML(nav);
        var i:Number = 1;

        // Pre-defined positions for topics
        //var topicPositions:Array = [ [10,90], [50, 90], [100,20], [10,
          20], [230, 80] ];
        //XXX: reverse order (.pop())
        var topicPositions:Array = [ [90,10], [90, 50], [20,100], [80,
          230], [50, 10] ];

        var topics:Array = [];
        var assocs:Array = [];
        var occurs:Array = [];
```

```
/*
This is really conceptual and should be replaced with a proper XML
parsing library... There is no error checking or fancy collision
checking or even clever positioning of topics or occurrences on
the canvas.

XXX1: create proper sets of node positions
XXX2: create classes for TAO, parse into objects, draw objects.
      I.e.
      pop a node position, create an object Topic with baseName
      and URI (pagename), add occurrences as children of this, and
      place children in respect to some PI-based algorithm, or
      even fixed positions/offsets around the topic. Draw lines.
XXX3: create proper associations. They should draw lines between
      members, and the main association role (and possibly
      labels on each side of the assoc role).
XXX4: Parse and draw subject identifiers
*/
  for each (var t:XML in tm.children()) {
    if (t.name() == "topic") {
      var node:Node;
      node.name = String(t.baseName);
      var topicPosition:Array = topicPositions.pop();

      node.x = topicPosition.pop();
      node.y = topicPosition.pop();

      /* ******************************************************
      - Find (and draw) all occurrences XML format:
      <occurrence>
        <type>TopicOrPageName</topic>
        <value>IRI or String</value>
      </occurrence>
      ****************************************************** */
      for each (var o:XML in t.occurrence) {
          var type:String = String(o.type)
          var value:String = String(o.value);
          /*switch(type) {
            case "uri": trace("uri occurrence"); break;
            case "document": trace("document occurrence"); break;
            case "image": trace("image occurrence"); break;
          }*/
          node.text += type + ": " + value +"\n";
      }
      topics.add(node);

    }
    /** Associations only support two members currently **/
    if (t.name() == "association") {
      var assoc:Assoc;
      assoc.setRole(String(t.role));
      var cnt:int;

      cnt = 0;

      for each (var m:XML in t.member) {
          var role:String = String(m.role);
          var topicRef:String = String(m.topicRef);

          for each (t:Node in topics) {
            if (t.name == topicRef) {
              if ( cnt == 0 ) {
                assoc.from = t; cnt++;
              } else {
                assoc.to = t;
              }
            }
          }
          // find topicRef node pos x,y
```

```
                associations.push(assoc);
              }
              //draw assoc:
              //
              r = members.pop();
              l = members.pop();
              roleR = r.pop(); xR= r.pop(); yR = r.pop();
              roleL = l.pop(); xL= l.pop(); yL = l.pop();
              a.drawLine(xR,yR,xL,yL);

            }

          }

          // Draw topics
          for each(var n:Node in topics) {
            topicCanvas.addChild(n.getComponent());
          }

          for each(var a:Assoc in assoc) {
            a.drawLine(a.getFX(),a.getFY(),a.getTX(),a.getTY());
            // XXX assoc label
          }
          /*
          var baseName:String = topics.pop();
          var nodeXY:Array = nodePos.pop();

          topicCanvas.addChild(createTopic(baseName,
              Math.round(Math.random() * 500), Math.round(Math.random() *
                    190) ) );

          while (topics.length > 0) {
            var tx:String = topics.pop();
            // position of tx:

          }

        // Insert nifty graph placement algorithm here!
        */
      }

      public function showToolTip(title:String, text:String,
           uri:String):void {
        trace(arg);
        this.currentState = 'Tooltip';
        toolTipTitle.text = title;
        toolTipText.text = text;
        //toolTipLink.href = uri;
      }
    ]]>
  </mx:Script>
  <mx:states>
    <mx:State name="Tooltip">
      <mx:AddChild relativeTo="{navigationPanel}" position="lastChild">
        <mx:Canvas x="580" y="0" width="200" height="210"
backgroundColor="#EEE7CE" color="#000000" id="tooltipCanvas">
          <mx:Label text="Privacy Act" id="toolTipTitle" fontWeight="bold"
fontSize="14" left="10" top="10"/>
          <!-- Maximum viewable text size: 314 characters... -->
          <mx:Text text="First 300 characters of the Zoomed in Privacy Act
page goes here... Suspendisse eu purus. Nullam sollicitudin. Nullam cursus
iaculis neque. Proin nonummy..." width="180" id="toolTipText" height="140"
horizontalCenter="0" verticalCenter="3"/>
          <mx:Button label="Open" right="10" bottom="10"/>
        </mx:Canvas>
      </mx:AddChild>
      <mx:SetEventHandler target="{linkbutton1}" name="click"
           id="toolTipLink" handler="currentState=''"/>
    </mx:State>
```

```
  </mx:states>
  <mx:transitions>
    <mx:Transition id="drawTooltip" fromState="*" toState="Tooltip">
      <mx:Parallel target="{tooltipCanvas}">
        <mx:WipeLeft duration="100"/>
        <mx:Dissolve alphaFrom="0.0" alphaTo="1.0" duration="500"/>
      </mx:Parallel>
    </mx:Transition>
  </mx:transitions>

  <mx:Panel x="0" y="0" title="Navigation" width="800" height="250"
id="navigationPanel" borderColor="#5086B6" layout="absolute">
    <mx:Canvas width="580" height="210" id="topicCanvas" x="0" y="0">
      <mx:LinkButton x="10" y="178" label="Tooltip"
click="currentState='Tooltip'" id="linkbutton1"/>
      <mx:LinkButton x="76" y="178" label="Manipulate topic"
click="jtNav.send()"/>
    </mx:Canvas>
  </mx:Panel>
</mx:Application>
```

Appendix B

This is a chapter cut from the thesis covering standards and interoperability, a related field, but I didn't feel it to be appropriate in the thesis.

Standards and Interoperability

Standards and interoperability has a central position in the evolution of information and knowledge systems. A main challenge for the Semantic Web to become a reality is connected to this paradigm. The evolution of the World Wide Web and Internet has led to an increasing effort on research into these topics, and to understand the Semantic Web one needs to understand these two central terms. This chapter explains the historic relevance and rationale that led to standardisation of the Internet, and explains the differences of three forms of standards that are common on the Internet.

With early internetworking came stronger needs for standardisation between network equipment. Early wide area networks like Arpanet and NORDUnet used their own protocols, and computer manufacturers all had their own standards for how computers were to communicate. NORDUnet was a Inter-Nordic academic and research backbone, when deployed in 1989, the "NORDUnet plug" (Hanseth, 2002; Lehtisalo, 2005), was the term used to describe the multi-protocol service interface for connecting several different networks, using many different network protocols (Brunell et al., 1988; Lehtisalo, 2005).

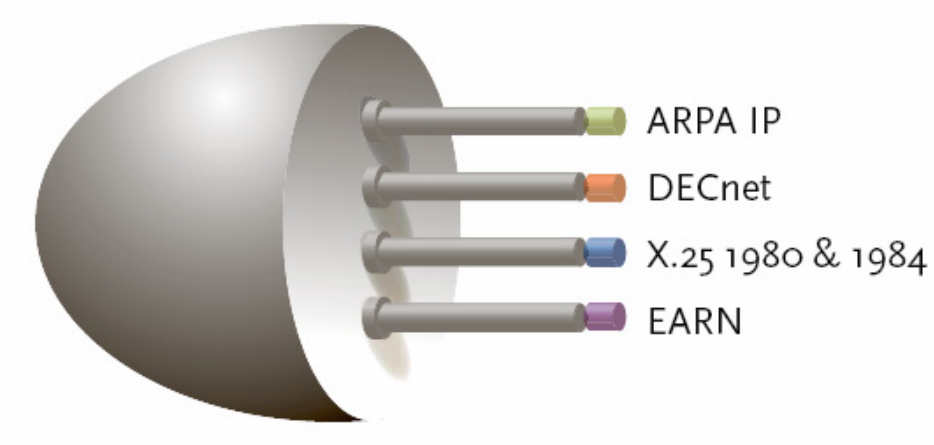


Figure 8-1 The NORDUnet plug (Lehtisalo, 2005)

EARN was a proprietary IBM protocol used on the X.EARN net, which was an initial name for NORDUnet. DECnet was a proprietary protocol as used by research institution CERN. DECnet in turn harmonised three other network protocols: HEPnet, SPAN and Cray's network protocols. ARPAnet and EUnet interconnection was performed with Arpanet Internet Protocol (IP) (and Transmission Control Protocol (TCP)). The International Telecommunication Standardisations Sector CCITT (now: ITU-T) X.25 standard protocol, based on the Open Systems Interconnect (OSI) technology was used by research institutions, mainly in Great Britain.

NORDUnet realised the difficulty of this situation, and aimed to reduce the number of different protocols used. The main discussion was whether to adopt the European developed X.25 standard, or the US developed TCP/IP interim standard. The latter gained the most popularity among computer manufacturers and interface driver system developers, and eventually won the competition, and became the Nordic University standard (Hanseth, 2002).

One of the most important views coming from this war of protocols was that network standards were needed to be interoperable, to be able to speak to one another, and that standards needed to be open as opposed to proprietary to achieve the goals.

Besides the battle of the network protocol, in the last decade, the focus on interoperability has played a strong part of the success of the World Wide Web and Internet technology. Both the commercial sector and academia are heavily into the research on standardisations and interoperability, without these standardisation processes, it is difficult to say if the Internet would be the success it is. Besides the technology, standards play a very important role in shaping the information systems in respect to the Semantic Web. Open, Market and Industry Standards are central in the transformation of the Web into a semantic and interoperable web.

Open standards

There has been a redefinition or change of direction as to what is considered to be "open" in the last five years. Open protocols and standards initially meant that a manufacturer provided the details for how to implement it, and disclosed the important features of the standard. What was considered appropriate use of the term prior to 2004 can no longer be considered the same post-2005 (Sutor, 2006). Sutor states that businesses choose to market their standards as "open" because they can decide whatever they want to call it. Their rationale is that open standards look good to the customers, and helps them gaining market share and popularity.

An example of such dubious use is Microsoft's Open XML (OOXML) standard, mainly developed for use with Microsoft's own Office 2007 Suite, which is now trying to be accepted as an official ISO standard. On the last ISO committee meeting, September 2nd 2007[82], OOXML failed to get the necessary 67% of votes in favour, and failed to achieve less than 25% in disfavour of the standard. The final decision on acceptance in February 2008 will decide if it is to be accepted. Competitors, including IBM and Sun have contested the standard on several points, and there is an online petition[83] to stop the standard process.

Sutor goes further, and states that the market's "open" standards also abuse the interoperability term:

> *"When a single vendor or software provider makes it easier to connect primarily to his or her software, this is more properly called intraoperability."*

This is the essence of the problem accepting OOXML as an ISO standard, and much less an open standard.

The official definitions of what constitutes an open standard also vary, but here are at least three principles that are clear of what makes a standard open (ITST, 2004).

- It is **accessible to anyone free of charge**. No discrimination between users, whether commercial or private.

- It **remains** accessible and free of charge to anyone. No revocation of rights at a later date.

- It is accessible free of charge and documented **in all its details**. There are no parts of the standard that cost, no patents, no hidden costs.

EU's definition[84]

> *The standard is adopted and will be maintained by a not-for-profit organisation, and its ongoing development occurs on the basis of an open decision-making procedure available to all interested parties (consensus or majority decision etc.).*

[82] http://www.iso.org/iso/pressrelease.htm?refid=Ref1070

[83] http://www.noooxml.org/

[84] European Interoperability Framework for Pan-European eGovernment Services, http://ec.europa.eu/idabc/servlets/Doc?id=19528

The standard has been published and the standard specification document is available either freely or at a nominal charge. It must be permissible to all to copy, distribute and use it for no fee or at a nominal fee.

The intellectual property - i.e. patents possibly present - of (parts of) the standard is made irrevocably available on a royaltyfree basis.

There are no constraints on the re-use of the standard.

World Wide Web Consortium's (W3C) definition[85]

transparency *(due process is public, and all technical discussions, meeting minutes, are archived and referencable in decision making)*

relevance *(new standardization is started upon due analysis of the market needs, including requirements phase, e.g. accessibility, multi-linguism)*

openness *(anybody can participate, and everybody does: industry, individual, public, government bodies, academia, on a worldwide scale)*

impartiality and consensus *(guaranteed fairness by the process and the neutral hosting of the W3C organization, with equal weight for each participant)*

availability *(free access to the standard text, both during development and at final stage, translations, and clear IPR rules for implementation, allowing open source development in the case of Internet/Web technologies)*

maintenance *(ongoing process for testing, errata, revision, permanent access)*

Internet Technology open standards include the Internet Societies' Request for Comments[86] (RFC) and Internet Engineering Task Force's[87] (IETF) ratification of these into Internet Standards[88] (STD). For instance the Internet Protocol, RFC 791, is ratified as STD 5. And Transmission Control Protocol, RFC 793 is STD 7.

Another set of open standards are the *web standards*, sometimes called *open formats* since they mainly focus on standardisation and interoperability of the information content, and not the hardware technology. Most of today's relevant Web standards are maintained by the World Wide Web Consortium[89] (W3C). W3Cs open web standards include (X) Hyper Text Markup Language ((X)HTML), Extensible Markup Language (XML), Resource Description Framework

[85] Definition of Open Standards, http://www.w3.org/2005/09/dd-osd.html

[86] http://www.ietf.org/rfc.html

[87] http://www.ietf.org/

[88] http://www.apps.ietf.org/rfc/stdlist.html

[89] http://www.w3.org/

(RDF), Cascading Style Sheets (CSS), and many others. All of W3C's web standards are fully open standards by all above definitions.

Not all open web standards are maintained by W3C, an example is XML Topic Maps (XTM), which is an open standard from TopicMaps.Org that is an implementation of the ISO standard of Topic Maps.

Market "de facto" Standards

As briefly stated, there is a difference between market "de facto" standards and open standards. Market standards normally appear as a proprietary solution, and after gaining a large installed base they are "opened" up (Hanseth, 2002).

An example of a de facto standard turned open is Adobe's Portable Document Format, PDF. Adobe opened up the access to the full specification already in 1993, but was not considered an open standard as Adobe still owned the format, and have not unequivocally provided irrevocable rights for future use in the license. PDF gained a large user base very fast and has been considered the standard for printable paper exchange for several years. PDF and was newly ratified as an industry ISO/IEC standard.

Industry "de jure" Standards

An industry or de jure standard is a standard drawn up and ratified by an official standardisation body. There are several official standardisation bodies; in Norway we have Standard Norge, who ratifies Norsk Standard (NS). Internationally the most relevant standardisation bodies are:

- International Organization for Standardization (ISO), including subcommittees like International Electrotechnical Commission (IEC).
- American National Standards Institute (ANSI) and National Information Standards Organization (NISO), often referred as ANSI/NISO.
- IEEE: Institute of Electrical and Electronics Engineers.

Many market standards become industry standards, a process that further gives value to the standard. Typically these processes are outcomes when competing market actors collaborate in creating a common standard. Sometimes market actors try to get their own "de facto" standards

ratified as industry standards, as in the case of my previous example, with Microsoft's OOXML ISO proposal.

There is however a significant difference between open and industry standards: industry standards are usually only available to obtain for a set fee to the standardisation body.