

**UNIVERSITY OF OSLO**  
Department of informatics

**Semantic Web technology in  
Human Resource  
Management**

**Master thesis**  
60 credits

Fredrik Halvorsen

**01.08 2008**



## Abstract

The aim of this master thesis is to examine how Semantic Web technology can be used in the field of Human Resource Management, more specifically how it can be used to improve areas of online job recruitment, focusing on improving searching, and examining how to match CVs and job ads to find the person best fit for the job.

Today's job search engines are usually based on word matching between the search keywords and the vacancy contents. The results returned are thus often too broad, or has nothing to do with what the searcher was looking for. Introducing Semantic Web technology, organising all competencies in a CV and competency requirements in a vacancy into an ontology, this could be used as a basis of search and comparison that gives much better accuracy than word matching.

In order to gain knowledge about the field and various systems successfully using Semantic Web technologies, a small test system was created. This system is based around a small ontology consisting of a competency branch, a job category branch and other branches with enough features to apply logical reasoning to and ask queries about. The ontology was developed in Protégé, with queries being asked in SPARQL. All is tied together with Jena, an open source Java toolkit, running locally on a Tomcat server. CVs and job ads consists of competencies and competency requirements respectively, and queries and comparison of these competencies are conducted using the ontology relations instead of matching words.

A survey of the technologies used to achieve this will be discussed, as well as some alternatives and competing technologies.

# Table of Contents

Abstract.....	2
1. Introduction.....	6
Human Resource Management on the Web today.....	7
Job description forming and publishing.....	8
Searching.....	8
Job-CV Matching.....	11
Job agents.....	13
Semantic Job Agents.....	13
Thesis outline.....	14
2. What is the Semantic Web?.....	15
Syntax and Semantics.....	16
Semantics.....	16
Ontologies.....	17
The use of ontologies.....	18
The use of ontologies in HR.....	19
Logic.....	19
Use cases.....	21
Creating a vacancy.....	21
Creating a CV.....	21
Saving a CV.....	21
Searching and matching.....	22
Use case scenarios.....	24
Sequence Diagrams .....	24
Benefits over current technologies.....	28
State of the art.....	29
Semantic HR systems.....	29
Existing Ontologies.....	32
Knowledge Bases.....	34
Other Semantic HR tools.....	35
Other competency matching systems.....	36
3. Underlying Technologies.....	37
Technology decisions.....	37
Topic Maps and Semantic Web.....	37
Why Semantic Web is better suited for my project than Topic Maps.....	38
Semantic Web layer cake.....	39
URIs and XMLNS.....	40
XML.....	40
XMLS.....	40
RDF.....	41
Complex RDF statements.....	42
RDFS.....	42
RDFS constructs.....	43

N3.....	43
OWL - Web Ontology Language .....	44
Three flavours of OWL.....	44
OWL = RDFS.....	45
OWL Ontologies.....	45
Open World assumption.....	45
Description Logic.....	46
Letters of DL.....	46
Modelling in Description Logics.....	47
Description Logics syntax.....	48
Semantic reasoner.....	49
Reasoners.....	49
DIG.....	50
Rule Languages.....	50
Querying the Semantic Web.....	51
SQWRL.....	51
nRQL.....	51
SPARQL.....	51
SPARQL syntax.....	52
Migrating from RDBS.....	53
4. Research Method.....	54
System development.....	54
Stages of systems development research.....	55
Background knowledge and motivation.....	56
Finding a core case.....	56
Research resources & data collection.....	57
Gaining knowledge about the field.....	57
Delimiting the field of interest.....	58
Project plan.....	59
System development method.....	60
Audience.....	60
5. Ontology design and project implementation.....	61
Forming an HR Ontology.....	61
What is a competency?.....	61
Competency modelling.....	63
Ontology design.....	67
Classes and instances.....	68
Properties.....	69
UML diagram.....	70
Rules.....	71
StepStone categories.....	71
Protégé.....	71
OWL version control.....	71
Ontology diagrams.....	72

Information Retrieval.....	73
Ontology population.....	74
Ontology Alignment.....	75
Development frameworks.....	77
Eclipse.....	77
Eclipse + IODT.....	77
EODM.....	77
Minerva.....	77
IODT from the Linux terminal.....	78
Jena.....	79
Jena saving owl file in persistent database.....	79
Jena with JSP.....	79
Jena querying an owl file with ARQ.....	79
Jena's Reasoning capabilities.....	80
Sesame.....	80
Web interface.....	81
Version 1 – SPARQL endpoint.....	81
Version 2 – User friendliness and competency relations.....	82
Version 3 - Semantic Job-CV matching.....	86
6. Evaluation & future work.....	88
Results.....	88
Evaluation.....	91
Future work.....	92
Relevance for other domains.....	93
7. Summary and conclusion.....	94
Conclusion.....	94
The future of the Semantic Web.....	95
Bibliography.....	96
Appendix.....	100
Source code for ontology.....	100
Source code for jsp files.....	130

# 1. Introduction

Online job recruitment as of today is to a large extent a one way operation. After having posted the vacancy on various job boards, applicants find it and apply for the job if the requirements matches their competencies. When enough applicants have applied for the position, the best candidate CVs are returned to the customer announcing the position, which filters out promising candidates for interviews. The recruiter then goes through the CVs interpreting the free form text and judging whether this candidate is best suited for the job or not. There's usually no automation in this process, it is all done by hand from when the job ad is designed to the position is occupied.

Imagine a system that automates the process of finding the most suitable candidates, ranking them with a score based on 'semantic closeness' between the CV and the job ad. When a vacancy text is received to be published, it gets tagged with competency requirements, and when users apply for the job, they do the same – choose their competencies from multiple drop downs in different categories, and state how proficient they are in each competency. When enough applicants have applied for the job, the recruiter simply presses a button 'rank applicants', and the list is ranked, with the CV matching the vacancy best getting the highest score.

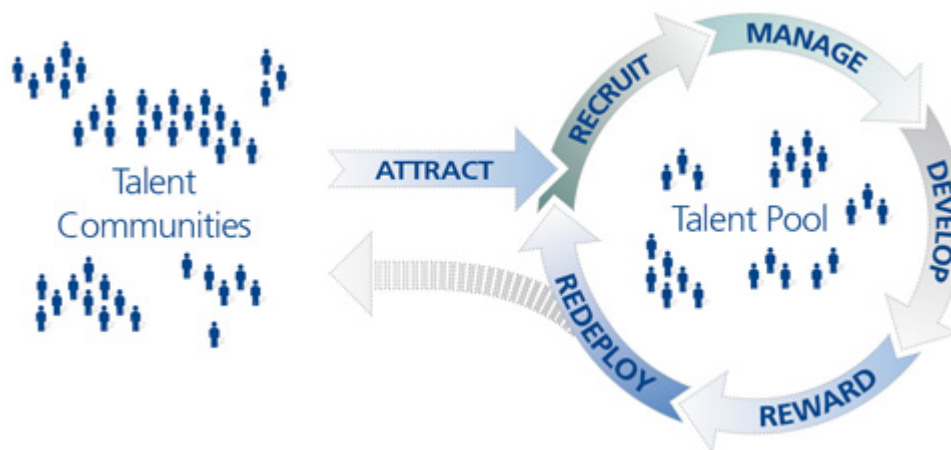
This last scenario could be the result of applying Semantic Web technology in Human Resource Management, and is the main theme in this master thesis. By tagging both CVs and job ads with semantic data organised in a competency ontology, it is possible to compute a match score based on the closeness of the competencies in the ontology tree. I will also look at how Semantic Web technology can improve searching, matching competency constructions against each other instead of free text word-to-word matching.

The next section will describe Human Resource Management on the internet in more detail, as well as how the various areas can be improved with Semantic Web technology.

## Human Resource Management on the Web today

Human Resource Management (HRM) can be a vague concept, meaning different things to different people, organisations and even books on the subject. But taking the concept 'managing human resources' in a broad sense, it covers all activities associated with employment relationships within a company, such as recruiting, hiring, time management, payroll, training and skill development, cost planning, performance monitoring and appraisal, etc. When talking about HRM in this thesis, it is concerning the recruitment process, hiring new staff.

HRM is a large part of the company's responsibilities, and many choose to let Recruitment Agencies (RAs) take care of some of these areas. Attracting the right candidates in today's job market is a challenging task, and for online RA businesses, HRM mainly concerns making sure their customers get the most fitting persons in their free job positions. It concerns everything from publishing the vacancy online in various job ad sites, gathering applications, grading the candidates, scheduling interviews (maybe doing the first one themselves, or conduct video interviews) between the applicants and the companies. It also concerns post-processes like keeping qualified applicants on record for later use, matching candidates with other free vacancies and managing the skills of existing employees.



*Illustration 1: Total talent management (© StepStone)*

To make sure this is possible, the staff in such agencies requires good recruitment tools that must be able to deal with all aspects of the hiring process, be able to manage a large number of both applicant CVs and customers job ads, achieving an effective and professional recruitment process with the best result possible.

## **Job description forming and publishing**

Job descriptions are usually written in a free-form uncontrolled format and sent (usually by e-mail) to publishing sites and recruitment agencies. They will then be published on an online (and perhaps paper-based) job board by a person, which decides what category the job ad fits into.

If a computer is to gather knowledge about these job ads however, the contents needs to be semantically annotated, using concepts from a controlled vocabulary. Controlled vocabularies are often claimed to improve the accuracy of free text searching, such as to reduce irrelevant items in the retrieval list. These irrelevant items, are often caused by the inherent ambiguity of natural language such as synonyms and similar meaning expressions.

Conducting a search in a database that uses controlled vocabulary is potentially more efficient and precise than free form search. The biggest advantage to controlled vocabulary is that once the correct term is found, most of the information you need is grouped together in one place, saving you the time of having to search under all of the other synonyms for that term. On the other hand, creating controlled vocabularies is never an easy task, and with a vast domain like job descriptions and qualifications, getting it right is surly going to be a challenge. The size of the vocabulary needs to be large enough to cover all categories, but not so large it defies its own purpose in allowing too much. Just covering synonyms is a huge task.

I will only touch the surface of this huge and difficult area in my thesis, and have thus limited my competency categories in the scope of computer science, though it's not in any way comprehensive enough to cover a fraction of all the aspects in even this area. I neither have a mechanism for denoting synonyms, trying to avoid them by sticking to categories with specific names, like programming languages etc.

## **Searching**

After the recruitment agency has posted a vacancy, it will then show up if a job seeker looks through the vacancies in that category, or if one of the words in the description matches words in the job seeker's search. The user thus searches for vacancies at job portal websites, matching his/her skills and experiences, applies (usually through a form) and sends in a CV, and wait for the recruitment agency or the job seeker to contact them for further action. As discussed above, free text searching for a vacancy through a web portal creates a great deal of 'noise' in the query result. Here's an example where I tried to search for web designer jobs at various online job boards 13.07.08 :



<b>free text, web designer without quotes, 25 first hits:</b>	
Bemanningspartneren	PDMS Designer
Xtra Personell Norge AS	Piping Designer
Xtra Personell Norge AS	Webdesigner
International Service Check	MYSTERY SHOPPER – SERVICE CHECKER – TEST CLIENT
Xtra Personell Norge AS	Markedsmedarbeider 60% stilling
Rett Bemanning AS	Salgs-/ markedskoordinator
Top Temp Bemanning as	Sales & Marketing consultant
Top Temp Bemanning as	JA-menneske søkes til administrativ deltidsstilling på Østerås!!
Top Temp bemanning Asker as	JA-menneske søkes til administrativ deltidsjobb på Østerås!!
Xtra Personell Norge AS	Piping Stress Engineer Lead
Xtra Personell Norge AS	SI Software - Systemkonsulent .NET
Xtra Personell Norge AS	SI Software - Systemutvikler
Christiania Personell AS	Kundeservice Billingstad
Xtra Personell Norge AS	Junior utvikler
Novo Nordisk A/S	Patent Information Specialist - vikar
Novo Nordisk A/S	Coordination of clinical trial supplies
Top Temp bemanning Bergen as	Lager/produksjonsmedarbeider – Os
Blekinge Tekniska Högskola	1-2 Universitetslektorer i företagsekonomi
Blekinge Tekniska Högskola	Universitetslektor i industriell økonomi
Blekinge Tekniska Högskola	Blekinge Tekniska Högskola söker universitetsadjunkt i medieteknik
Blekinge Tekniska Högskola	Blekinge Tekniska Högskola söker universitetslektor i medieteknik
Lene V Norge AS	Franchisedriver LENE V OS
Lene V Os	Franchisedriver LENE V OS
Rett Bemanning AS	Web-designer
	<b>webdesigner in one word:</b>
Xtra Personell Norge AS	Webdesigner
	<b>'web designer' with quotes:</b>
Rett Bemanning AS	Web-designer
	<b>via category search ('Webdesign og webmaster'):</b>
Xtra Personell Norge AS	Webdesigner
Xtra Personell Norge AS	Prosjektleder
Xtra Personell Norge AS	Senior Prosjektleder
Xtra Personell Norge AS	Support/brukerstøtte
Xtra Personell Norge AS	Nå har du muligheten!
Xtra Personell Norge AS	Selvstendige driftskonsulenter til Vestfold
Xtra Personell Norge AS	Brukerstøtte fra mai 2008?
Xtra Personell Norge AS	SI Software - Systemkonsulent .NET
Xtra Personell Norge AS	Senior Microsoft-Citrix-VMWare driftskonsulent
Xtra Personell Norge AS	Salgskonsulenter til Ementor
Rett Bemanning AS	Web-designer
DnB NOR Personmarked	Rådgiver/Senior Rådgiver webdesign og brukeropplevelse
Xtra Personell Norge AS	IT/Driftskonsulent Drammen / Kongsberg
Xtra Personell Norge AS	Brukerstøtte/1. linje support
mnemonic as	Sikkerhetskonsulent UNIX og nettverk
Inhouse	Three Nordic Site Managers to MTG Online
Xtra Personell Norge AS	Webutvikler
Dfind	Vi søker en WebMaster til kunde
Xtra Personell Norge AS	IT Konsulenter
Xtra Personell Norge AS	Support Ingeniør

As seen from the results there is a lot of noise, and although some of the noise is caused by the fact that job ads are being placed in the wrong categories, the most of the noise occurs when using free text search without choosing a category. There are several reasons for this, but mainly because of the syntactic nature of current database querying; there is not much to go by other than matching the typed words against vacancy texts in the database, perhaps accompanied with location and job sector delimiters.

It should be noted that although I have the most knowledge about StepStone, most of the large Norwegian job boards (Finn<sup>1</sup>, Monster<sup>2</sup> and NAV<sup>3</sup>) states that searching is based on word for word matching, and using quotes further limits the search results.<sup>4</sup>

Traditional databases and inventories of competencies that rely on string search and string matching are thus not sufficient for this task. This approach, not being really effective from a qualitative viewpoint, is a disadvantage both for recruiters and for job seekers. The former have difficulties to choose the best person for their needs in the often massive amounts of CVs and information, the seekers have difficulties to emphasise their competencies through the writing of their CVs and have difficulties finding job offers which correspond to their profiles. Systems that understand the knowledge aspects of competency information are required.

In my thesis, the semantic search functionality tries to improve on syntactic word matching by looking at 'similar' concepts surrounding the concept in question, based upon proximity in the ontology structure.

---

1 <http://www.finn.no/jobb/>

2 <http://monster.no/>

3 <http://www.nav.no/>

4 [www.finn.no/finn/gojsp/daily/misc/help\\_search\\_popup.jsp](http://www.finn.no/finn/gojsp/daily/misc/help_search_popup.jsp)

## Job-CV Matching

One of the most important aspects of recruitment process is getting the right person to the right place. The ability to rank candidates to a specific position is an area where there is considerable gain to be had with Semantic Web technologies. The ability of matching a CV against a job ad to calculate a score based on similarity could be a valuable tool for the recruitment agency, and also a feature to implement in job agents. Although alternatives are emerging (as will be discussed in a later chapter), there are very few systems that has this kind of feature automatically done by computer, so this would prove an important point when looking at what is achievable with Semantic Web technology. Here is a proposal for such a calculation suggested by a program called 'Matching On Competencies (MoC)'<sup>5</sup> :

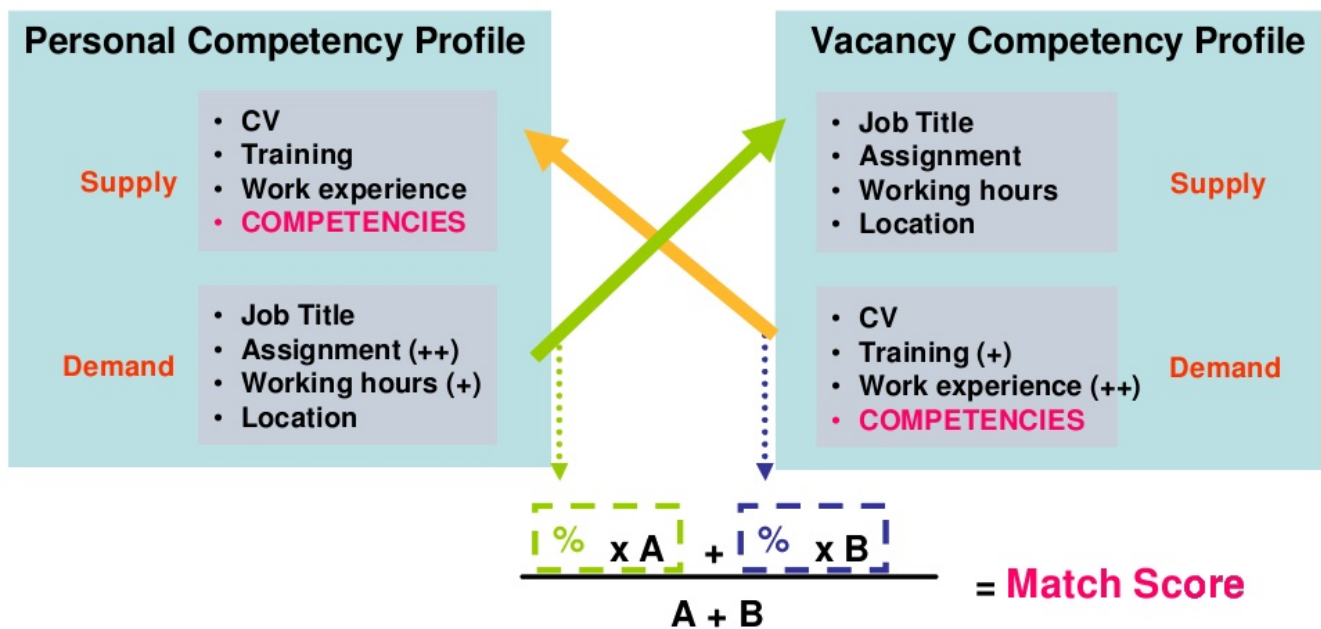


Illustration 2: matching on competencies (© Vrije Universiteit Brussel STAR.Lab)

I have used this figure more like an inspiration rather than a specific implementation, but I've tried to match most of the criteria, with the main focus on competencies, as will be explained later through use cases and the project implementation.

5 <http://eu.hr-xml.org/hr-xml/wms/customers/pdf/Competencies.pdf>

Here is a flowchart of a proposed matching algorithm from (Colucci2003) in Description Logics syntax, described later. The purpose of showing these algorithms is more to show some of the research done in the area, and the complexity of the algorithms, not necessarily use them in my implementation.

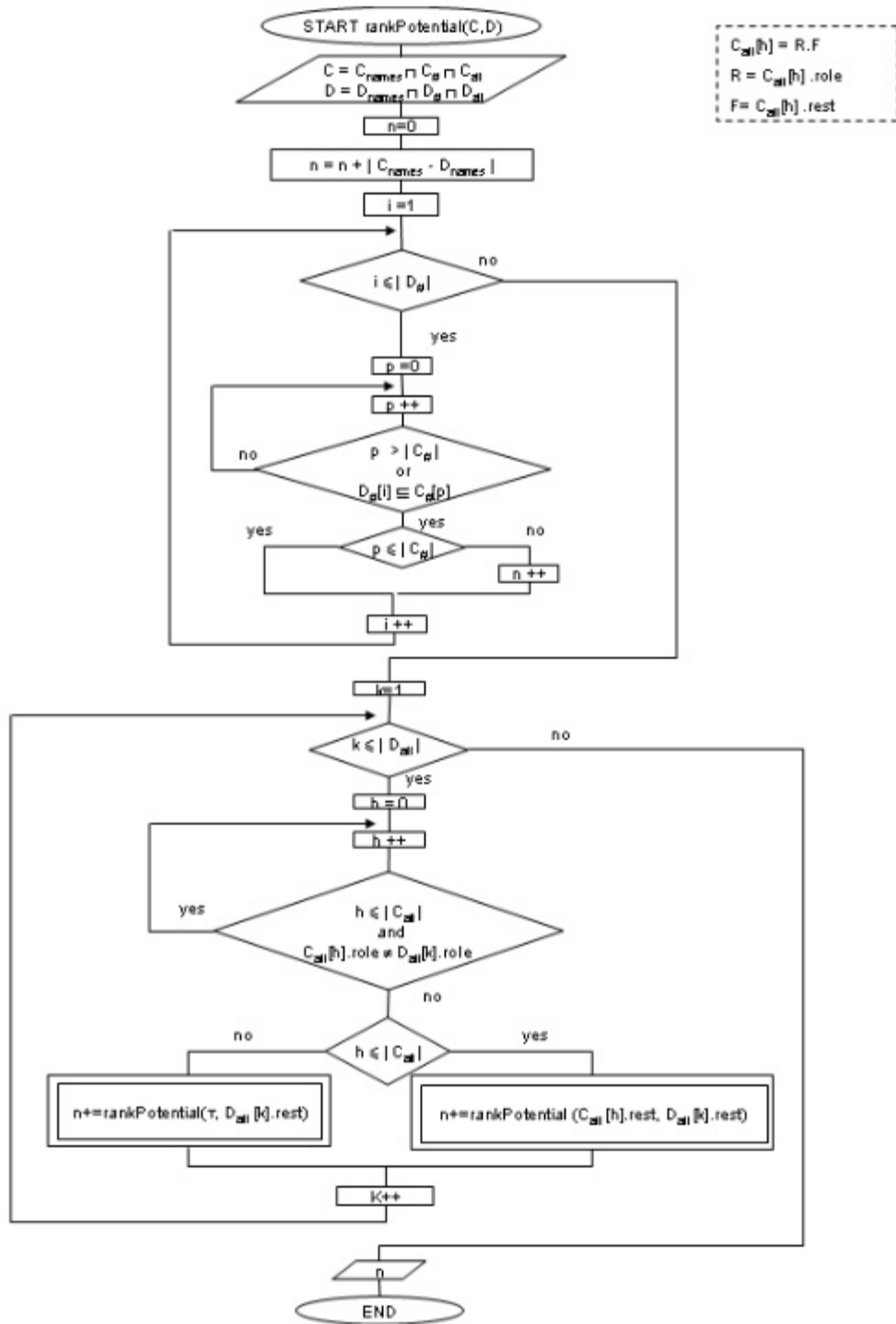


Illustration 3: proposed matching algorithm (© Colucci S., Di Noia T., Di Sciascio E., Donini F.M., Mongiello M., Mottola M.)

## Job agents

Many recruitment agencies offer business intelligence tools such as job agents which are used to find, classify and structure job ads tailored to the user's CV. Using an agent implies that the user instructs the agent to search information in place of them, by declaring the categories and keywords that he/she wants to be searched. The results are usually given on email when matching vacancies appear. Hence, agents relieve the user from searching among the many posting sites and applications available on the net. This is a huge time-saver, and may also produce more extensive results. Today's agents mostly work by the same kind of word matching as stated above, so the result is often inaccurate and needs extensive filtering. So the time saved during the search could easily be wasted in screening the answers received and in revising the items selected for the search.

## Semantic Job Agents

“The real power of the Semantic Web will be realised when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services (including other agents) become available. The Semantic Web promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data come with semantics.”

– Tim Berners-Lee<sup>6</sup>

In the true vision of the Semantic Web, an 'intelligent' job agent that crawls the Web looking for suitable jobs on various job boards is one of the ultimate goals. If standards are established and companies start using the technologies, this goal is indeed not far fetched. Not only can there be agents for job seekers, but also agents working for the recruitment agencies, communicating with the job seeker's agents, and acting out the interesting scenario of setting up potential interviews where human interaction is limited to agreeing on a time and place.

---

<sup>6</sup> Tim Berners-Lee; James Hendler; Ora Lassila, *The Semantic Web*, Scientific American, 17.05.01 <http://www.sciam.com/article.cfm?id=the-semantic-web>

## Thesis outline

Although there are many streamlined tools that ease the task of job publishing and applicant management, there is still great potential within the appearing realm of Semantic Web technologies, which brings new ways of organising data in a way that hasn't been available before. In this thesis I will introduce the reader to these emerging techniques, and how it can improve the recruitment process and provide better results for the end users. I will survey some of the technologies and tools available today, and look at things to come that is going to take the Semantic Web even closer to reality.

The first section of chapter 2 will start with an introduction to the Semantic Web, followed by a look at what benefits are to be had from knowledge bases and ontologies, as well as reasoning with logics. Then follows a proposal of use cases on how semantically enabled searching, posting and matching of applications could work, and look at the benefits and drawbacks of these proposals. A survey of the current systems and ontologies will follow, with a list of examples related to the area of HR management.

A survey of the underlying technologies will then be given in chapter 3, describing details and syntax of different competing technologies, as well as a discussion on how to migrate from current technology.

A discussion on research methods follows in chapter 4, with argumentation for the choices taken, details on system development methods and project plan, as well as a discussion on delimiting the scope of the problem.

A detailed walk through of the implemented system will be given in chapter 5, with a discussion of the designing and building of the ontology, the Web solution, accompanied by screen shots and code examples.

Results and evaluation will follow in chapter 6, looking at if and how the achieved work can be generalised to the design as a whole as well as other domains, finishing off with a summary and concluding remarks in chapter 7.

## 2. What is the Semantic Web?

“I have a dream for the Web in which computers become capable of analysing all the data on the Web - the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialise.”

- Tim Berners-Lee<sup>7</sup>

“The entertainment system was belting out the Beatles' “We Can Work It Out” when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: “Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments.” Pete immediately agreed to share the chauffeuring.

At the doctor's office, Lucy instructed her Semantic Web agent through her hand-held Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating of excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasised keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it - University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Almost instantly the new plan was presented: a much closer clinic and earlier times - but there were two warning notes. First, Pete would have to reschedule a couple of his *less important* appointments. He checked what they were - not a problem. The other was something about the insurance company's list failing to include this provider under *physical therapists*: “Service type and insurance plan status securely verified by other means,” the agent reassured him. “(Details?)”

Lucy registered her assent at about the same moment Pete was muttering, “Spare me the details,” and it was all set. (Of course, Pete couldn't resist the details and later that night had his agent explain how it had found that provider even though it wasn't on the proper list.)”

- Tim Berners-Lee<sup>8</sup>

The Semantic Web, summed up in one statement, is the idea of making the Web more readable for machines and so called software agents. Most content on the Web today is designed to be processed by and interacted with humans, and the Semantic Web is about making the enormous and fast-growing amounts of information available on the net more accessible, by letting computers sort out the parts we are interested in and using the vast knowledge available in a more efficient manner. The semantic side of it is getting the computer to 'understand' the meaning of the content, and this huge and difficult task is what semantic technologists are trying to address, making the future Web a better one. The Semantic Web as such, is not a particular technology in itself, but a term describing several technologies and standards with the common goal of enabling the computer to extract information from this content. One area where better organising and annotation of data is needed, is in the online job market, which will be explored in this master thesis.

---

<sup>7</sup> Tim Berners-Lee; Mark Fischetti, *Weaving the Web*, 1999, <http://www.w3.org/People/Berners-Lee/Weaving/>

<sup>8</sup> Tim Berners-Lee; James Hendler; Ora Lassila, *The Semantic Web*, Scientific American, 17.05.01, <http://www.sciam.com/article.cfm?id=the-semantic-web>

## Syntax and Semantics

Syntax (from Ancient Greek *syn-*, "together", and *taxis*, "arrangement") is the school about the structure of phrases, sentences and discourse, and determine their relative grammaticality. The term syntax can also be used to refer to these rules themselves, as in "the syntax of a language" (e.g. "the syntax of French" or even "the syntax of C++"). Modern research in syntax attempts to describe languages in terms of such rules, and to find general rules that apply to all languages. Syntax says nothing about the meaning behind these phrases, so to find out the context and relations of words, one has to turn to semantics.

## Semantics

In linguistics, semantics is the sub field devoted to the study of meaning, above the syntactic levels of words, phrases, sentences, and even larger text. Semantics has been a part of several scientific disciplines, including Computer Science. Research areas addressing issues concerning semantics include Information Retrieval, Information Extraction, Computational Linguistics, Knowledge Representation (KR), Artificial Intelligence and Database Management. Most of these areas have very different views of what "meaning" is, and these views are all built on some meta-theoretical and epistemological assumptions (Lytras2006). Information retrieval, extraction and computational linguistic techniques primarily draw upon analysis of unstructured texts in addition to document repositories that have a loosely defined and less formal structure. In these sorts of data sources the semantics are *implicit*, and not explicitly defined as in knowledge representation, artificial intelligence and database management, which rely on well defined syntactic structures to represent information/knowledge. These structures have definite semantic interpretations associated with them, and there are rules of syntax that controls the ways the syntactic structures can be combined to represent the meaning of these structures, limiting the expressiveness to gain computational benefits. This is the type of semantics called *formal* semantics (Sheth2005).

For some time, formal semantics has been the main area of defining the Semantic Web, being used for automated approaches to exploiting Web resources, including the creation of Ontologies and the annotation of resources, complemented by reasoning and query processing. But since most KR mechanisms and RDBS models are based on set theory, they are not very capable of representing knowledge that is imprecise, uncertain, partially true, approximate and otherwise 'fuzzy'. But representing and utilising these fuzzy types of more powerful knowledge is the 'next step in evolution', and what the Semantic Web is trying to achieve. These techniques are called *Powerful soft Semantics*, and are distinguished from formal and implicit semantics by incorporating rules and probabilistic mechanisms to deal with this 'fuzziness'. This field is still in its infancy, but much effort is being put into exploring the possibilities of applying rule languages that allows an extra layer on top of formal ontologies to cope with these issues.



## Ontologies

“What is an ontology? Short answer: An ontology is an explicit specification of a conceptualization.”

-Tom R. Gruber, 1993<sup>9</sup>

Ontologies, the concept most likely dating back to Plato's theories on metaphysics, concerns entities and the relationship between entities, sets and subsets of entities, and the relationships between these like 'is part of' and 'has part'. In computer science, concepts within a specific domain and the relationship between these concepts form an ontology. For example, within the domain 'computer hardware' it is useful to talk about concepts like 'motherboard', 'hard drive', 'RAM', 'Gigabyte', 'Gigahertz' etc.

Tom Gruber at Stanford University, proposed in 1992 that the meaning of ontology in the context of computer science is “a description of the concepts and relationships that can exist for an agent or a community of agents.” He goes on to specify that an ontology is generally written, “as a set of definitions of formal vocabulary.”<sup>10</sup>

Common components of ontologies include (wikipedia):

- **Individuals** : instances or objects (the basic objects)
- **Classes** : sets, collections, concepts or types of objects
- **Attributes** : properties, features, characteristics, or parameters that objects (and classes) can have
- **Relations** : ways that classes and objects can be related to one another
- **Function terms** : complex structures formed from certain relations that can be used in place of an individual term in a statement
- **Restrictions** : formally stated descriptions of what must be true in order for some assertion to be accepted as input
- **Rules** : statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- **Axioms** : assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In these disciplines, axioms include only statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.
- **Events** : the changing of attributes or relations

Ontologies, at a basic level, should be able to specify "is-a" relationships like "A Labrador is a dog". As such, this is a Taxonomy or classification, which is like a relational model and implies "Is subclassification of".

---

<sup>9</sup> <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>

<sup>10</sup> <http://www.metamodel.com/article.php?story=20030115211223271>

More expressive than a Taxonomy, a Thesaurus is defined as

"a controlled vocabulary arranged in a known order and structured so that equivalence, homographic, hierarchical and associative relationships among terms are displayed clearly and identified by standardised relationship indicators....The primary purposes of a thesaurus are to facilitate retrieval of documents and to achieve consistency in the indexing of written or otherwise recorded documents and other items," (ANSI2003)

On the level above Thesaurus we find Conceptual models. This is the level where it becomes possible to express subclass relations between parent and child classes.

The highest level of ontology definition is described as Logical Theory. At this level, ontologies become semantically interpretable by software, and is where we find OWL, the ontology language used to form this thesis' ontology.

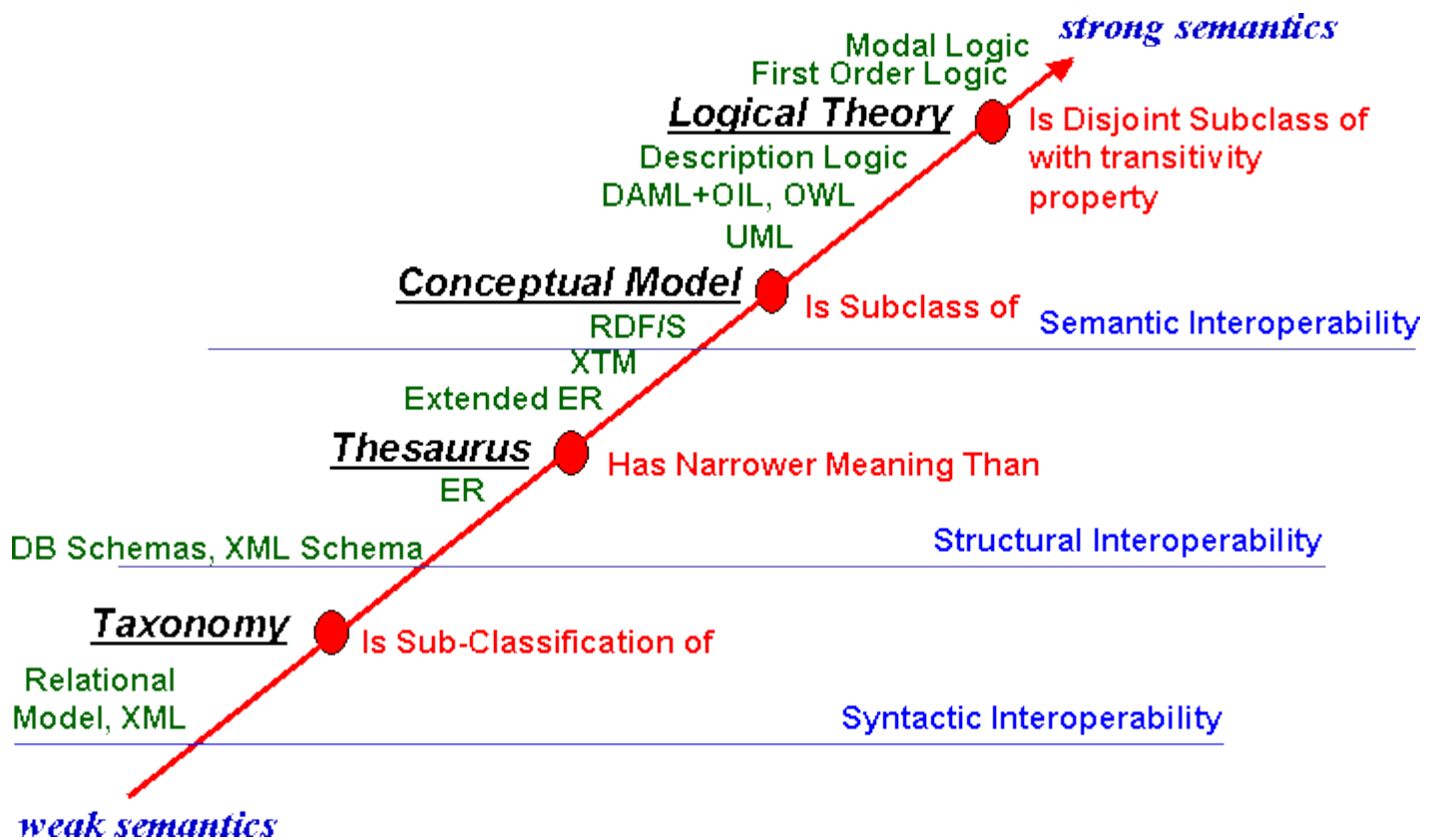


Illustration 4: Ontology Spectrum (© Applied Minds)

## The use of ontologies

Tim Berners-Lee named ontologies as one of the key technologies in his vision of the Semantic Web. In the Semantic Web, shared meanings are ascribed to terms used in published Semantic Web documents, thus enabling resources on the Internet to be processed by machines, instead of resources being merely human-readable text(OOAHR2007). Ontologies are a basis for data- sharing, processing, and

integration, and knowledge representation in a chosen domain. With that, ontologies surpass taxonomies in that in addition to defining sub- superclass hierarchies it also tries to describe the domain completely, and uses rules and constraints to classify, determine distinctions and identify relationships between entities or resources.

In 2002 the Gartner Group predicted a massive usage of ontologies for business application integration in the time frame 2005-2010, foreseeing a road map starting with lightweight ontologies or taxonomies evolving into strong knowledge representations in 80% of application integration projects within this time frame. The use of ontologies is likely to expand enormously in two particular ways: data browsing, search and retrieval (moving from retrieving documents to retrieving specific data, and enhancing search technologies with semantics), and in terms of inferencing.<sup>11</sup>

## **The use of ontologies in HR**

The potential advantages for utilising Semantic Web technologies in the HR business are many. First of all, organising competencies and skills in an ontology gives a lot of implicit information and relations not easily modelled in regular relational database systems. Other advantages includes the added accurateness when using a controlled vocabulary when categorising, expandability with ontology alignment, dealing with synonyms etc., something that will be discussed more in detail throughout the thesis.

When thinking about modelling the domain of human resource management, the question is 'what do we talk about, when we talk about HR?'. If we are to make an HR ontology, it must represent what the business of human resources is about. This will be discussed in more detail in chapter 5.

## **Logic**

Logic, the term in this case indicating formal logic, has a special role in the Semantic Web. It plays many different roles, examples being:

- specifying ontologies and vocabularies
- applying and evaluating rules
- inferring implicit facts from the explicitly known ones
- explaining why conclusions have been reached
- detecting contradictory statements and claims
- representing knowledge, what kinds of things may be said about a subject, and how statements are to be understood
- stating and executing queries to obtain information from Semantic Web data
- combining information from distributed sources to a coherent whole

Logic is important in ontologies in particular, where concepts and terms have logical relationships to each other that needs specifying, which means that any ontology system must adopt to some form of logic, formal or not. Ontologies contains concepts and terms, and logic provides ways to make statements using these concepts and terms, and also the ability to reason about collections of statements

---

<sup>11</sup> [http://www.gartner.com/DisplayDocument?doc\\_cd=109295](http://www.gartner.com/DisplayDocument?doc_cd=109295)

that use these concepts and terms, may they be from other resources like web pages, databases or other ontologies and knowledge bases.

Not all logics are suitable for the Semantic Web however, because they are too expressive. First order logic for example, considered the most complete logic, has constructs that causes problems for computers that wants to find a solution to a problem, in that it may never end its reasoning. This means that we need a logic that is both computable and decidable, which mean we can be sure to get an answer within reasonable time of the execution. A subset of first order logic, a family of logics called Description Logics, imposes enough restrictions to be both computable and decidable, and is what is used in the ontology language used in this thesis. I will not get into deeper details concerning logic, other than listing and explaining the constructs used to acquire the desired results.

In the next section I will propose some use cases portraying how some of the visions presented above can be achieved in the Human Resource Management domain using Semantic Web technologies.

## **Use cases**

Use cases are useful for describing in a formal way how a system should work, its requirements, what happens if the user stray from the intended work flow, alternative ways of achieving the same results etc. They help discovering flaws in the system, and use cases is often used to describe how the flaw was discovered. I will mainly look at use cases proposed for how my system could work in a completed state. The real system will probably not come close to this functionality, but it is still useful to have 'perfect' use cases to strive after. Before going into specific use cases and sequence diagrams, I will discuss the most important scenarios in broader terms to give a background on how I see the system working in a finished state. As I am doing this implementation as a part of my thesis, and thus isn't meant to be a complete functional system, I won't implement things like login procedures etc., so I won't be describing such use cases either.

## **Creating a vacancy**

Vacancy creation is done via a web form where users can log in and organise their vacancies. In addition to creating new vacancies, you can view a list over current active ones and past vacancy ads. Creating a new vacancy gives the user a page with category selection (on as many levels needed, dynamically), level of experience for each chosen category, and desired educational degree. (maybe also indicators on the elements which are 'flexible' and to what degree) This way, we ensure that it can be matched against CVs tagged with the same information. in addition to this controlled mark-up comes the vacancy title and description area, stored in a comment field.

## **Creating a CV**

CV construction is also used with a controlled vocabulary. We have the same categories, experience levels and educational degree as in the vacancy creation page, so we can do a computational deduction on how close a vacancy and a CV is, using weighting on traversing sub/superclass levels and sibling levels.

A user chooses categories and experience levels from drop downs as with vacancy creation, and has a 'non-semantic' field which he can insert additional information not computed by the ontology.

## **Saving a CV**

Ideally, after building his CV using this system, the user can save a copy of his semantically marked CV locally, and re-use this in other job portals. For this to work, a common standard on categories, experience and degree levels (among others) needs to be agreed on. Standardisation of concepts solves a lot of problems generated by impreciseness of search terms, and an agreement on the meaning of terms must be in place to effectively communicate at all.

Concept standardisation would allow users to maintain the same postings and application formats from one service provider to another, thus saving time, whereas sharing a full common language would allow employers to simply put their job posting on their websites without sending them to each reference service provider selected. In fact, in this way, the job-search and job-CV matching service providers would be able to autonomously crawl the companies' websites, extract the proper information identified, and make suitable inferences. (OOAHR2007)

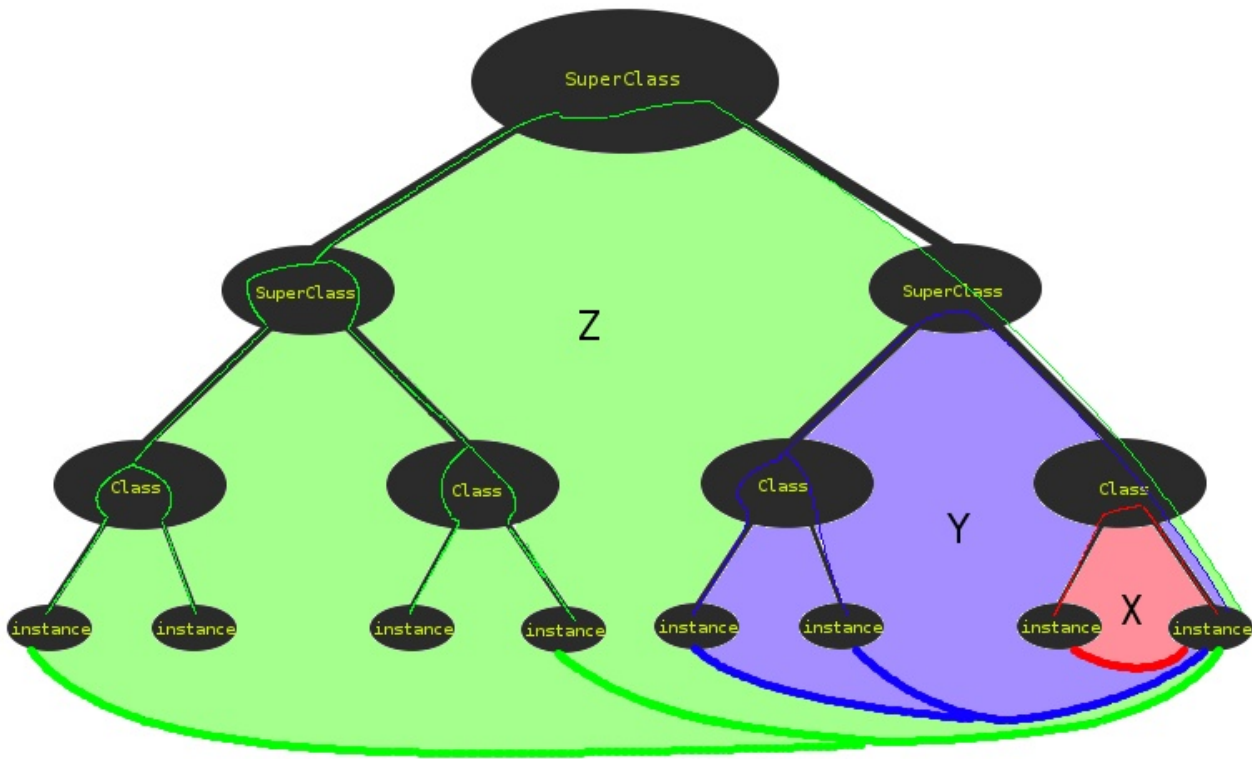
## Searching and matching

As a job seeker, you're interested in getting search results that are matching your competency. If you're a programmer with 3 years work experience in Java looking for a new job, a probable search string would be 'Java programmer'. Suppose that query returns 0 hits. There are no available Java programming jobs, so you either go to another job portal, or you try to improve your search. In an ontology-based system, one could present to the user a slider in the interface, and by dragging this slider you would 'relax' the query, moving upwards in the ontology to a more general level; the superclass. Depending on the ontology, this generalisation could mean 'less experience', 'less education', or a 'more general category'.

In my model ontology, Java programming's more general category, or superclass is 'object oriented programming'. If you include this in the query result, you would get vacancies concerning other object oriented languages like C++, C#, TurboPascal and Visual Basic. As a Java programmer, there is a high probability that you have at least some experience in one of these languages, or at least would be able to learn them quickly. This also work the other way, for the recruitment agent looking for a suiting candidate for a position. If there are no candidates with a perfect match, one can relax the query and look at candidates with similar competencies.

Having vacancy postings and job seekers described using controlled vocabulary from a HR-ontology would allow us to perform this kind of semantic matching, i.e. the calculation of the degree of semantic similarity between an applicant's profile and job requirements. What this means is that based on rules, the system can perform statistical calculations of 'closeness' by computing the distance between the nodes in the hierarchy, and infer which CV is the be best match for a given job ad. Suggestions on how to compute scores have been proposed by several parts as discussed above, but here's an example I used as a starting point:

- instances of the same class gives X (the most) points (siblings)
- instances of sibling classes gives Y (some) points (cousins)
- instances of sibling superclasses gives Z (the least) points (2<sup>nd</sup> cousins)
- extra points for experience at or above required competency level
- minus points for experience below required competency level
- extra points for levels above degree
- minus points for levels below degree
- extra points for location close to the workplace location



*Illustration 5: point distribution where X is the most and Z is the least*

This is an illustration that shows the suggested point distribution above. This includes three class levels in the matching, but in the end I chose to just include one class and one superclass level in my implementation.

## Use case scenarios

The use case model is a model that describes a system's functional requirements in terms of use cases. It consists of the users of the system and the various use cases by which the user interact with the system, thereby describing the functional behaviour of the system. There are many models of varying complexity, and I have chosen a fairly simple model that I think is sufficient for describing the various uses of the system.

<b>1. Use Case Name</b>	A use case name provides a unique identifier for the use case. It should be written in verb-noun format (e.g., <i>Register User</i> ), should describe an achievable goal (e.g., <i>Register User</i> is better than <i>Registering User</i> ) and should be sufficient for the end user to understand what the use case is about.
<b>2. Goal/Intent</b>	Without a goal a use case isn't worth much. There is no need for a use case when there is no need for any actor to achieve a goal. A goal briefly describes what the user intends to achieve with this use case.
<b>3. Description</b>	Describes the interaction between the user and the system, summarising what process is to be achieved and therefore the functionality to be met by the solution. The main flow is included within this section.
<b>4. Precondition</b>	State any conditions that need to apply prior to this use case being able to start i.e. applicant must have submitted an attached CV as a word document.
<b>5. Post Conditions</b>	State any conditions that must be in force after the use case has been completed to enable the flow of the system to continue
<b>6. Alternative Flow</b>	State the course if the main flow is not followed i.e. the CV is not submitted

## Sequence Diagrams

Sequence diagrams provide a graphical representation of object interactions over time. These typically show a user or actor, and the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case scenario's flow of events. Sequence diagrams are a great way of documenting usage scenarios, capturing requirements and verifying object use later in the design. The diagrams can vary in the degree of detail, from a user view using 'lament terms', to a programmers view going directly into variable names, classes, objects, methods and return values. Originally, my idea was to create the sequence diagrams as only a superficial way to describe the user flow, but as I made them after I had implemented most of my system, I found it useful to contain a bit more detail such as showing where SPARQL is sent from, what kind of data is returned, and at what stage it is parsed and so on. So even though these diagrams come pretty early in the thesis, they contain some information that will be explained in more detail later. Alternate flow has been omitted from the sequence diagrams for clarity, but could easily have been displayed as a return value with an error message after doing the query/queries.



1. Search for jobs
2. The goal is to find vacancies matching the job searcher's requirements
3. Description
  1. user chooses 'Job Ads' from the search drop down
  2. user enters competencies as keywords to describe requirements
  3. user clicks 'search' and is represented with available vacancies (ranked after best match)
4. user must have typed in keywords in order to narrow down the search
5. the results are represented as links for further details
6. nothing matching keywords, user is asked to reformulate

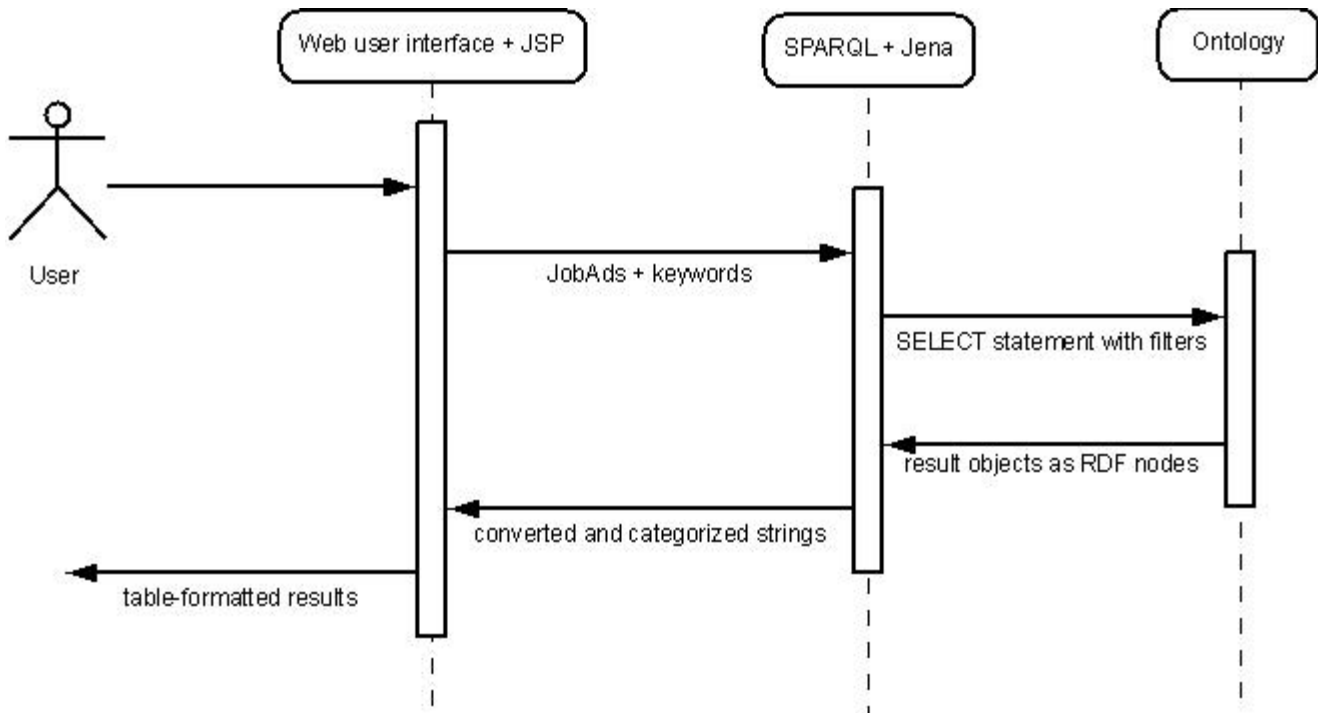


Illustration 6: 'search for jobs' sequence diagram

1. Search for candidates
2. The goal is to find candidates matching the recruiter's requirements
3. Description
  1. user chooses 'CVs' from the search drop down
  2. user enters wanted competencies as keywords to describe requirements
  3. user clicks 'search' and is represented with available candidates (ranked after best match)
4. user must have typed in keywords in order to narrow down the search
5. the results are represented as links for further details
6. nothing matches keywords, user is asked to reformulate

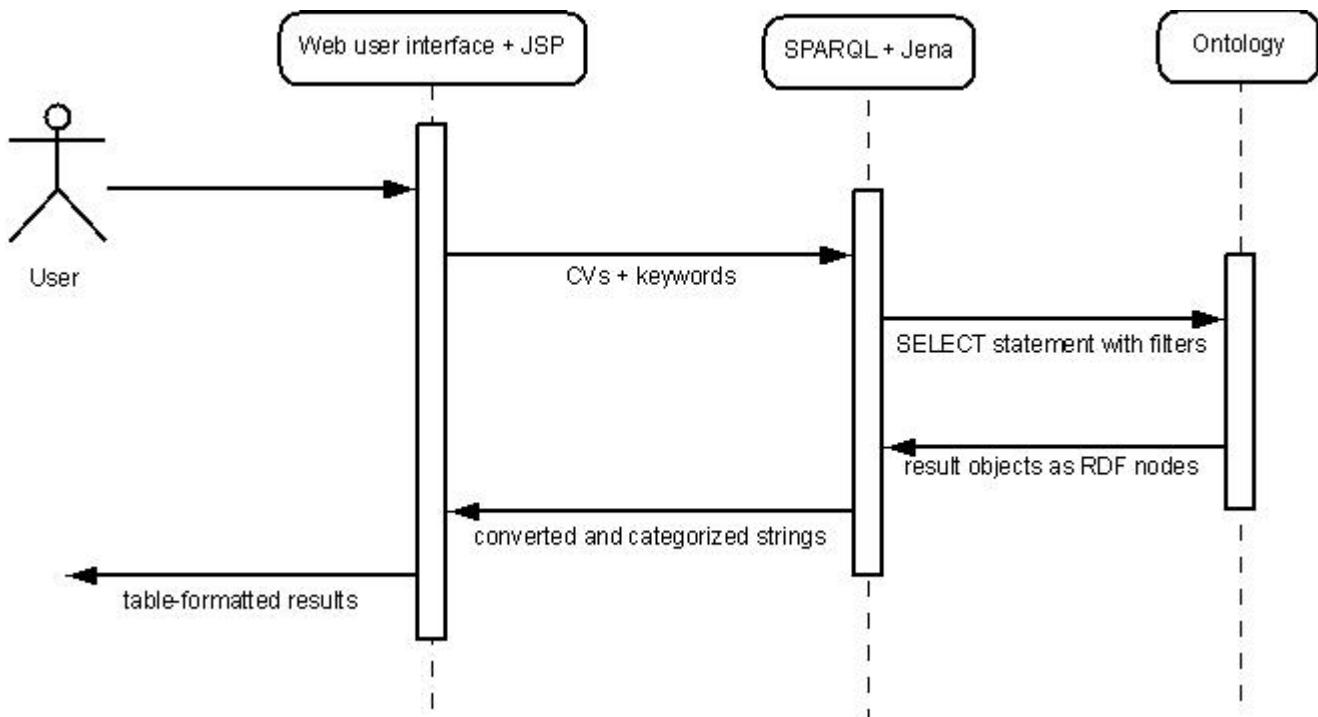


Illustration 7: 'search for candidates' sequence diagram

1. Match candidates with a vacancy
2. The goal is to find candidates suitable for the vacancy's requirements
3. Description
  1. user chooses 'Job Ads' from the search drop down
  2. user enters competencies as keywords to describe requirements
  3. choose a vacancy from the results
  4. on the 'vacancy competencies' page, user clicks the 'matching CVs' link
  5. the user is represented with a ranked list of matching candidate CVs
4. the user must have created the vacancy, specifying what competencies are desired
5. the user can click on each result, to see what competencies the candidate has stated in the CV
6. two alternate flows
  1. nothing matches keywords, user is asked to reformulate
  2. nothing matches Job Ad results, user could be given the option of 'relaxing' the requirements

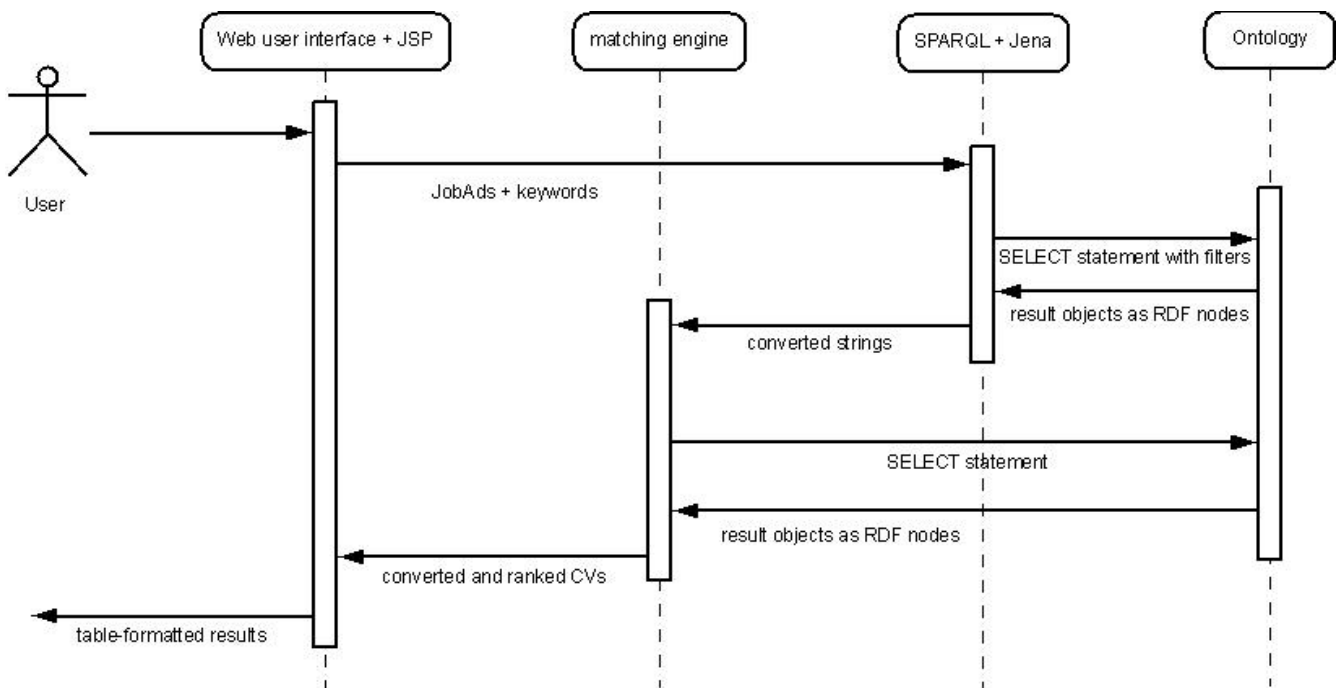


Illustration 8: 'find candidates matching vacancy' sequence diagram

Throughout these visions, one can see obvious benefits of incorporating semantic technologies. In the next section I will discuss benefits over today's systems, and suggest applications of semantic technology which will improve the quality of the process, and reduce the amount of human intervention in the system.

## **Benefits over current technologies**

What is to be gained from using Semantic Technology that is not achieved with traditional procedures and relational databases? After all, mapping class hierarchies can be done with relational databases. Here are only some of the advantages:

- constructing the hierarchical relations in RDBS requires a lot more effort than using ontologies, which provides a lot of these constructs relatively cheap, where there's no need to explicitly define all relations or use Ids, keys and foreign keys etc.
- the power of combining data with other data sets that don't follow the data model you've been using in your traditional system
- easier to add more data that necessarily doesn't fit the current table structures
- exchange data with any application that knows how to handle Semantic Web constructs (RDF) over the Web, email, web services or any other way you can exchange data; traditional database data have no equivalent way of doing this
- perform logical reasoning to find relationships not explicitly stated in the data
- import or use someone else's ontology to study relationships between properties and resources in *your* data
- add statements about publications and references defined elsewhere on the Web, needing only the URIs they have published
- all of the above using well-defined standards, enabling a wide range of software to process the data

## State of the art

At present, the UK and the US have the most sophisticated job-search and job-CV matching services. In these countries, the standardisation process on concepts began about 20 years ago and now they have powerful and shared systems nationwide. For example, the National Occupational Standards - NOS from the UK and the O\*NET from the US. The European Commission has also launched standardisation programmes such as Europass – concerning CV and qualifications standardisation - and the eSkills forum initiatives - addressing the ICT skills and competences standardisation with the forthcoming European eCompetence framework<sup>12</sup>. The challenge is to reach common standardised semantic systems nationwide and Europe-wide. For this reason, a strong cooperation between national and international institutions as well as the construction of multi-stakeholder partnerships is needed to facilitate and foster the labour market mobility and transparency. (OOAHR2007)

## Semantic HR systems

Through the next section I will do a brief survey of some of the current/recent HR projects applying Semantic Web technologies. The list is by no means complete, and some of these projects might be outdated already. These were the ones that had documentation online, or was mentioned in the OOAHR roadmap document and such could be considered central projects within the field. Ideally I should have gotten into deeper detail about each of them, and compared them to my implementation. Unfortunately because many of them are commercial closed-source projects with little information, but mainly because of time limitations, this hasn't been realized.

### LEA

Lingway e-Recruitment (LEA), part of LINGWAY HR Suite solution, is the only offer currently on the market that enables users to automatically analyse applicants' resumes and match them with the job offers available. LEA 4.0 release is HR-XML standards-compliant.

The solution is made up of 3 modules:

1- LEA CV (Automated resume acquisition):

Automated management of incoming resume flows: allows users to capture resumes in print format by OCR, and resumes in Word and PDF format sent by e-mail or through corporate websites. They are then classified in the resume base.

2- LEA Job Offer (Automated Job offers analysis):

Allows users to automatically control the quality of job offers according to marketing and legal (affirmative action/equal opportunity) criteria and to find the most relevant key words for job offers.

3- LEA Search (Semantic search engine on resume and job offers):

Semantic search engine for job offers and resumes (matching): enables users to automatically find the resumes that match the job offers, and to offer applicants jobs that correspond to their resumes.

### Matching on Competencies (MoC, replacing the traditional Job-CV matching)

Unfortunately there's very little info online about MoC other than the illustration used in the first chapter. It is part of a HR-XML competency work group presentation, held at the HR-XML Consortium Europe Conference in Paris the 18. May 2006, and is credited to VUB STARLab.

---

12 <http://www.ukstandards.org/> , <http://europass.cedefop.europa.eu/> , <http://communities.trainingvillage.gr/esf>

## **Ontotext's JOCI**

“Jobs & Contacts Intelligence – Recruitment Intelligence through Semantic Web Technologies” (Innovantage, Fairway Consultants) gathers recruitment-related information from web-sites of UK organizations, offers services on top of this data to recruitment agencies, job portals, and other. JOCI uses KIM for information extraction (IE, text-mining), and makes use of a domain ontology to support the IE process, to structure the knowledge base with the obtained results, and facilitate semantic queries.

## **Innovantage**

Innovantage launched in early 2006 to exploit semantic enabled search technologies to deliver actionable business intelligence harvested from the internet.

The first product, Insight, tracks and monitors online recruitment advertising - accelerating lead generation, business development and market research for recruitment agencies and job boards. The in-house technology platform aggregates and normalises jobs posted on over half a million direct employer websites as well as many of the UK's leading job boards. Clients access and manipulate data through an internet browser or at a more sophisticated level through custom XML feeds. Innovantage has invested heavily in next generation Web technologies based on Natural Language Processing. This solves the problem of finding and structuring jobs posted on many thousands of direct employer websites. This is an exciting and evolving technology and will be the basis of many of Innovantage's new products.

## **DOGMA**

DOGMA is an ontology approach and framework that is not restricted to a particular representation language. This approach has some distinguishing characteristics that make it different from traditional ontology approaches such as (i) its groundings in the linguistic representations of knowledge and (ii) the methodological separation of the domain-verses-application conceptualization, which is called the ontology double articulation principle. The idea is to enhance the potential for re-use and design scalability. Conceptualizations are materialized in terms of lexons. A lexon is a 5-tuple declaring either (in some context G):

1. taxonomical relationship (genus): e.g., < G, manager, is a, subsumes, person >;
2. non-taxonomical relationship (differentia): e.g.', < G, manager, directs, directed by, company >.

Lexons could be approximately considered as a combination of an RDF/OWL triple and its inverse, or as a conceptual graph style relation.

## **Learning in Process: Context-Steered Learning Framework**

LIP (Learning In Process) is a European project aiming at providing highly contextualized e-learning systems addressing the needs of the knowledge intensive organization. LIP focusses on the idea that the context matter in order to deliver more effective learning. This context can be organizational, individual, or related to the learning process. LIP is able to take into account this context (represented using ontological modelling) in order to «assemble» specific learning courses out of a set of learning objects, and / or to intervene proactively. An ontology is used for the representation and evolution of a complex user model.

## **The eCCO System : An eCompetence Management Tool Based on Semantic Networks**

According to several national Institutions', ICT company associations' and the Italian Ministry of Innovation and Technology's requests for standardisation of ICT job profiles, an ICT job profiles model was defined. It is based on ontologies principle to describe “Knowledge Objects”, “Skills”, “Competences” and their relations. The model takes into account the several formalisations from national and company ICT job profiles frameworks and refers to the scientific literature about ontology and competences definitions. A software tool has been implemented on the basis of the model defined. It allows collecting individual users' job profiles and analysing the gaps against a set of ICT standard profiles. The semantic network behind gives the system flexibility. Furthermore, the system enables the dynamical enrichment of the

semantic network by uploading new skills by users, who can also suggest linkages between the new nodes. The system administrator will later evaluate and accept them.

## PROLIX

The objective of PROLIX is to align learning with business processes in order to enable organisations to faster improve the competencies of their employees according to continuous changes of business requirements. To reach this goal, PROLIX develops an open, integrated reference architecture for process-oriented learning and information exchange.

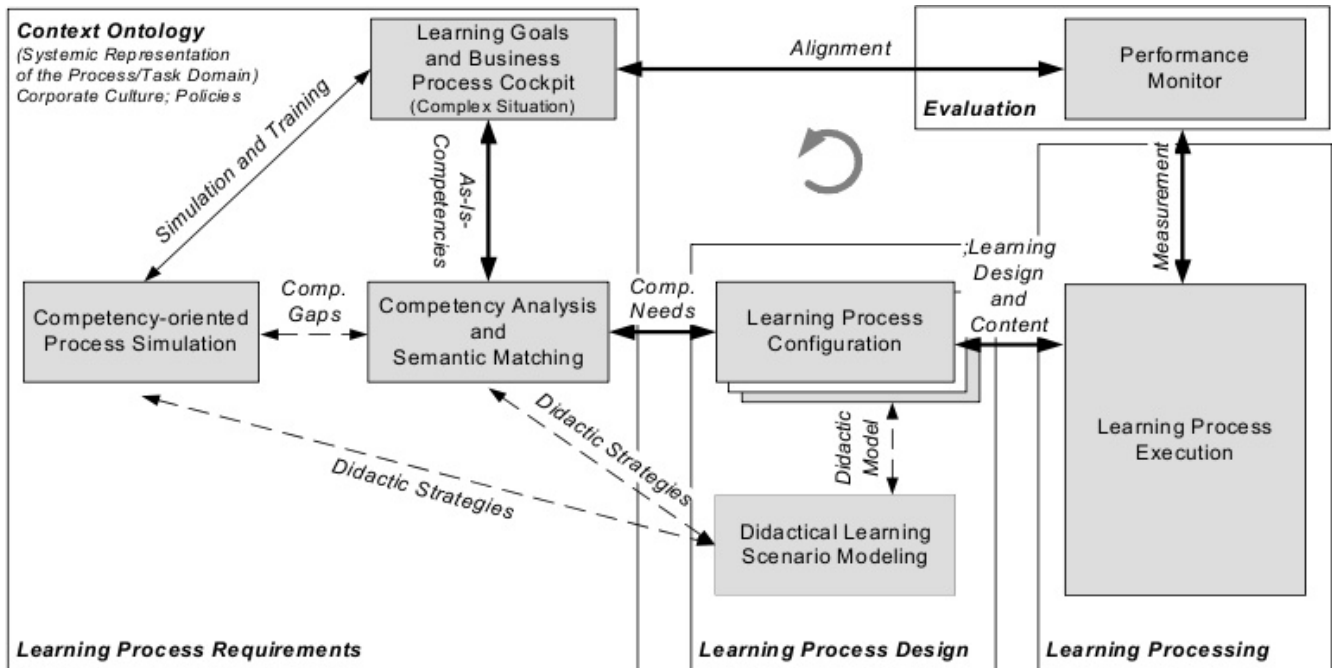


Illustration 9: Context Ontology (© PROLIX)

## CoDrive

The CODRIVE project is a competency elicitation project for vocational education. It aims to develop a new competency driven approach to knowledge in vocational education, which will facilitate and innovate interoperability and matching between Learning Content Management Systems (LCMS) and Public Employment Service Applications (PESA) through intelligent competency ontology design. The CODRIVE Project is part of the Leonardo Da Vinci Community Vocational Training Action Programme (Phase 2), an initiative by the European Commission Education and Culture DG.

## PoCeHRMOM

The PoCeHRMOM (Project omtrent Competenties en functies in e-HRM voor technologische toepassingen op het Semantisch Web door Ontologie en Meertalige terminologie) project aims to provide small to medium-sized enterprises (SMEs) with competency management possibilities. Existing e-HRM applications (e.g. automatic translation of job openings) are mostly relevant for companies with an in-place competency management. The development of this knowledge is a tough job, which scares SMEs away from competency management. As a result, they cannot benefit from existing (and continuously improving) e-HRM applications. The main focus of this project is to develop a common database that SMEs can use to build their own competency profiles.

## TRACE

The central aim of the EU Leonardo sponsored project TRACE is to improve the transparency of competences across EU

states, regions and sectors. TRACE is investigating the current state and use of existing competence systems and, from this investigation, is developing a methodology and technique to create interoperability between competence systems, especially between different competency frameworks. One of the anticipated outcomes of TRACE is the definition of an intermediate competency “language” or description which will enable users to reference competency descriptions to a common repository of competencies, though within the scope of the TRACE project this will only be achieved for a subset of domains.

## **Existing Ontologies**

As with the previous section, I will present a list of ontologies with relevance to the Human Resource domain. Neither this list should be considered complete or dated, but merely a compilation of work that has been done on the subject.

### **ProPer Ontology**

The ProPer ontology is a simple ontology providing exemplary facilities for skill inferencing. It is available in OIL/DAML, in German, but it's not developed any more.

### **KOWIEN Ontology**

Another German ontology not developed further, focusing not only on HR issues, but also consisting of a generic top-level ontology with a domain-specific profile of competence and skills management. Its strength lied in using F-Logic as an ontology formalism.

### **Knowledge Nets**

Based on the KOWIEN ontology and the German translation of HR-XML, this ontology was developed within the Knowledge Nets project at the Free University of Berlin. It contains national and international classifications for jobs and branches.

### **ePeople**

This ontology was developed at DaimlerChrysler 2003-2006 in KAON, in the context of the ePeople project (in cooperation with FZI Karlsruhe), aiming at established an integrated competence management system at Daimler Chrysler. It primarily represents Skills, Skill Profiles of Employees and Job Skill Requirements in order to allow for exploiting similarity measures on competency profiles for skill profile matching .

### **LIP Ontology**

This ontology was developed within the EU project Learning in Process (2002-2004) in order to support embedding learning processes into work and business processes. Its focus is on the automatability of on-demand learning support and is directed towards relating employees, their organizational context and relevant learning resources (which can range from learning objects up to immature documents or colleagues). It specifically aims at bridging the gap between e-learning, knowledge management, human resource development and performance support.

### **CommOnCV**

CommOnCV is a project, which aims at dealing with the problematics of e-recruitment by considering a new approach based on competency management. The idea consists in allowing a job seeker (respectively a recruiter) to identify and formally represent the competencies underlying its Curriculum Vitae (respectively its job offer). These competencies, which allow to make knowledge, skills, abilities, traits and motives acquired by a person (respectively required for a job) explicit, are then used to refine the matching process between "supply and demand". From a technical viewpoint, these competencies correspond to annotations, which are formally represented by using Semantic Web languages.



## **TOVE**

Developed at the University of Toronto, the Toronto Virtual Enterprise Ontologies represents a set of integrated ontologies for the modelling of commercial and public enterprises, implemented in PROLOG using first-order logic. The machine-readable files are not publicly available.

## **Professional Learning Ontology**

The successor of the LIP ontology, this is the result of merging LIP with competence management approaches for improving training planning processes. It is modelled in OWL-DL, but the majority of the ontology is also in OWL-Lite. This ontology is freely available under a Creative Commons license from [http://www.professional-earning.eu/competence\\_ontology.shtml](http://www.professional-earning.eu/competence_ontology.shtml).

## **PROTON (PROTO-Ontology)**

PROTON was developed by Ontotext Lab as a light-weight upper-level ontology, which serves as a modelling basis for a number of tasks in different domains. The ontology was designed not for a fairly complete modelling of the domain, but rather for information extraction purposes for automated metadata extraction and other techniques. The ontology was developed in OWL-Lite and is publicly available from <http://proton.semanticweb.org>.

## **COKE**

COKE is a three-level ontology containing a top-level Human Resources ontology, a middle-level Business Process ontology and a lower-level Knowledge Objects ontology which are related to organizational entities. It tries to connect the organizational frame with individual knowledge objects. It is developed by the University of Calabria with DLP+ as a formalism. The ontology is not publicly available .

## **Knowledge Bases**

A knowledge base provides the means for the computerised collection, organisation, and retrieval of knowledge. A knowledge base may use an ontology to specify its structure (entity types and relationships) and classification scheme. An ontology, together with a set of instances of its classes, constitutes a knowledge base.

### **WordNet**

WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. The database and software tools have been released under a BSD style license and can be downloaded and used freely. The database can also be browsed online.

### **DBPedia**

DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows users to ask expressive queries in SPARQL against Wikipedia and to interlink other datasets on the Web with DBpedia data. The DBpedia project uses the Resource Description Framework as a flexible data model for representing extracted information and for publishing it on the Web. As of September 2007, the DBpedia dataset consists of around 103 million RDF triples, which have been extracted from the English, German, French, Spanish, Italian, Portuguese, Polish, Swedish, Dutch, Japanese, Chinese, Russian, Finnish and Norwegian versions of Wikipedia.

### **Dublin Core**

The Dublin Core metadata element set is a standard for cross-domain information resource description. It provides a simple and standardised set of conventions for describing things online in ways that make them easier to find. Dublin Core is widely used to describe digital materials such as video, sound, image, text, and composite media like web pages. Implementations of Dublin Core typically make use of XML and are Resource Description Framework based. Dublin Core is defined by ISO in 2003 ISO Standard 15836, and NISO Standard Z39.85-2007.

### **FOAF**

FOAF (Friend of a Friend) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe him or herself. FOAF allows groups of people to describe social networks without the need for a centralised database. FOAF is an extension to RDF and is defined using OWL Web Ontology Language. Computers may use these FOAF profiles to find, for example, all people living in Europe, or to list all people both you and a friend of you know. This is accomplished by defining relationships between people. Each profile has a unique identifier (such as the person's e-mail addresses, a Jabber ID, or a URI of the homepage or weblog of the person), which is used when defining these relationships.

## SHOE

SHOE is a small extension to HTML which allows web page authors to annotate their web documents with machine-readable knowledge. SHOE makes *real* intelligent agent software on the Web possible. HTML was never meant for computer consumption; its function is for displaying data for humans to read. The "knowledge" on a web page is in a human-readable language (usually English), laid out with tables and graphics and frames in ways that we as humans comprehend visually. Unfortunately, intelligent agents aren't human. Even with state-of-the-art natural language technology, getting a computer to read and *understand* web documents is very difficult. This makes it very difficult to create an intelligent agent that can wander the Web on its own, reading and comprehending web pages as it goes. SHOE eliminates this problem by making it possible for web pages to include knowledge that intelligent agents can actually read.

## Other Semantic HR tools

Here are some other Semantic Web enabled tools designed to improve HRM tasks, and which are used in practice.

### Computas Skills Manager

The skills manager is a part of the Intranet at Computas, and every employee has access to it. You can select a skill from a taxonomy of around 250 different technical skills, related to the core competence of the company. When you select a skill, you can find which of seven skill levels people have, from "expert" to "irrelevant". In addition to indicating their skill level, people also indicate which level they want to have in the future. When viewing skills, details are shown in black if people are on the level they would like to be on, red if this is a topic they do not wish to work on in future, or green if they want to develop their skills in this direction.

Employees are prompted in the front page of the Intranet to evaluate themselves when new skills are introduced in the tool. They are also told to update the information when they have completed a project. Anyone can suggest new skills to the tool, which will be included by the manager of the "competence center" process.<sup>13</sup>

### Ontoprise

"Ontoprise is the leading provider of industry proven Semantic Web infrastructure technologies and products used to support dynamic semantic information integration and information management processes at the enterprise level. With its mature and standards-based products and its know-how Ontoprise is delivering a key portion for the upcoming Semantic Web. Ontoprise has developed a comprehensive product suite designed to support the deployment of semantic technologies in the enterprise. At the core of the Ontoprise platform are the OntoBroker® and OntoStudio® middle ware products. These products incorporate highly efficient inference engines, as well as a modelling environment for the development of ontologies."<sup>14</sup>

---

13 <http://www.kunnskapsnettverk.no/C13/Kompetansekartlegging/Document%20Library/Managing%20Hard%20Skills-5.doc>

14 <http://www.ontoprise.de/de/en/home/products.html>

## Other competency matching systems

### SkillZoom - The recruiter's competency matching tool

“Skillzoom helps recruiters match job seeker competencies to specific position requirements with unprecedented reliability and accuracy. Job seekers no longer need to maintain several versions of their resume, one for each of their targeted occupations. SkillZoom's recruiter talent acquisition tools address most of the problems inherent to resume-based electronic searching:

- Maintaining multiple resume files for the same person
- Removing keyword variability problems: did the job seeker use the same word choices as the recruiter? How much impact did these word variations have on the results' relevance sort? How much higher (or lower) on the list?
- Some keywords may be spelled exactly the same, yet they may not be from the right context or industry sector.
- Qualifier/adjective-styled word choices (i.e. dependable, reliable, punctual, effective, ... etc.) are too hard to use.
- Accepted acronyms, spelling variants, ... etc. etc.

SkillZoom tools help recruiters search for difficult competency combinations, even for your most exacting clients. Find multi-skilled individuals with precision, all in one place, rather than multiple filing systems containing several resume variants and differing levels of detail for the same individual.

Job Competency Matching	Job Skills Matching
Competency Management	Talent Acquisition
Recruiting Tools	Job Matching” <sup>15</sup>

The Skillzoom website is still under construction, so there is unfortunately very little information about it and whether it is based on Semantic Web technology or not.

---

<sup>15</sup> <http://skillzoom.com/>

## 3. Underlying Technologies

### Technology decisions

When I was starting out trying to decide what technologies to use for achieving improvements over regular search/matching of CVs and job ads, I was quickly introduced to 'The Description Logic Handbook' by my first supervisor. After reading about it and searching on the Web, I quickly realised that this was part of the reasoning power of OWL, and with this the path had been laid out. Later on I had heard about Topic Maps (TM) associated with Semantic Web technology, but didn't consider it as I had already started out with Protégé.

In the following section I will try to compare the two 'schools', but as I don't have much knowledge about Topic Maps, the comparison might be lacking.

### Topic Maps and Semantic Web

First off, one must ask if the technologies are comparable, which in this case they are, though only to some extent. In some aspects, they aren't really comparable as Topic Maps focus on ontologies made for human readability and organisation, and Semantic Web is made for computers to read and infer new knowledge over. They are however comparable in some other aspects, as they are both identity-based, the concept in both technologies are symbols representing 'things' which statements can be made about. These 'things' are called *subjects* in TM and *resources* in RDF, and these concepts are equivalent. In addition to subjects/resources, we have the symbols, which in TM are represented as *topics*, and in RDF as *nodes*.

The interesting part is what we can say about these 'things', and this is where the differences are emerging. Whereas RDF has only one way to make assertions about things, TM have three different kinds of *topic characteristics*, called names, occurrences, and associations. RDF assertions are called statements, which are represented in subject-predicate-object triplets (discussed more thoroughly later), where subject is the node the statement is about, the object is the value of the subject (either a resource or a literal), connected by the property-node that defines the relationship.

In Topic Maps, Names are the simplest form of assertion, and associates a name-string with the topic. In RDF assigning a name to a resource is done by using a name property (e.g. foaf:Name) to assign a literal to the resource, and in order to recognise the property as a name, knowledge about the RDF vocabulary being used is necessary, which isn't the case with TM. The next level are occurrences, which can be properties of the topic stored as strings inside the topic map, or references to information resources that are relevant to the topic. Occurrences have a type, which is a topic. Occurrences are structurally identical to RDF properties, ignoring the support for qualification. Semantically however, there is a significant difference, since RDF properties have less constrained semantics than occurrences do in Topic Maps. The last characteristics are Associations, which represent relationships between things. Associations have a type, and each topic participating in the association plays a role within the

association, meaning that any number of topics can participate in an association, each playing its own role. Associations are completely different in structure from RDF statements, as they have roles representing the involvement of each topic in the association, and they go both ways. To achieve this we need the inverse property of OWL. Another consequence of this is that in Topic Maps associations can involve more than two topics, something that is impossible for RDF statements. Hence, associations are structurally more complex than RDF statements, and also contain more information in the form of the role types.

The information represented in a topic map, expressed in one of the XML interchange syntaxes for Topic Maps, can, at some level of detail, be translated into information that is expressed in one of the XML interchange syntaxes for RDF information. The translated information can then be used in the context of RDF applications that would otherwise not be able to use it<sup>16</sup>.

### Why Semantic Web is better suited for my project than Topic Maps

Topic Maps are better suited to the kinds of tasks first developed for – collocation, navigation and integration of concepts and addressable resources. These objectives also cover many of the needs for using information on the Semantic Web, but RDF may be a better choice for representing other kinds of tasks, such as supporting logical reasoning over large datasets. There is a large area in which they overlap however, thus both have been used to define the structure and detail of entire web sites. Another point is the matter of standardisation. RDF has a standardised way of expressing an ontology and the constraints upon it, while as of present time, Topic Maps Constraint Language, TMCL, is not finished. RDFS + OWL allow more inferences to be drawn than the basic Topic Maps standard (TMDM) does, and according to statements by Lars Marius Garshol<sup>17</sup> more inferences than TMCL will be able to.

Here is a visual comparisons of the two schools. However, it's not quite accurate, as TMQL is still not developed and Topic maps as such doesn't have its foundation based on formal logics.

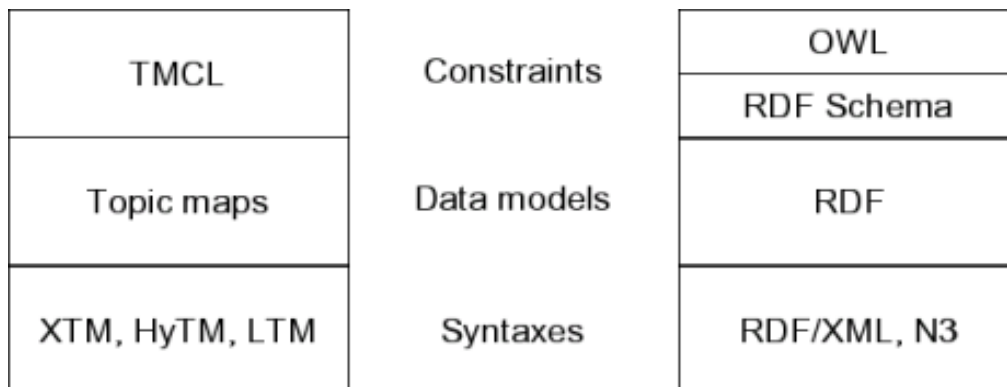


Illustration 10: Topic Maps and Semantic Web (© Ontopedia)

An interesting analogy however, is made in (Passin2004), where RDF is referred to as 'information by the atom' where the predicate is the chemical bond between the two atoms, whereas Topic Map is referred to 'proteins of knowledge' which fit together in specific ways like complex molecules.

16 XML Topic Maps through RDF Glasses - <http://www.cogx.com/cogx/xtm2rdf/extreme2001/>

17 <http://www.ontopia.net/topicmaps/materials/tmrdf.html>

## Semantic Web layer cake

Here is the latest incarnation of the official Semantic Web layer cake:

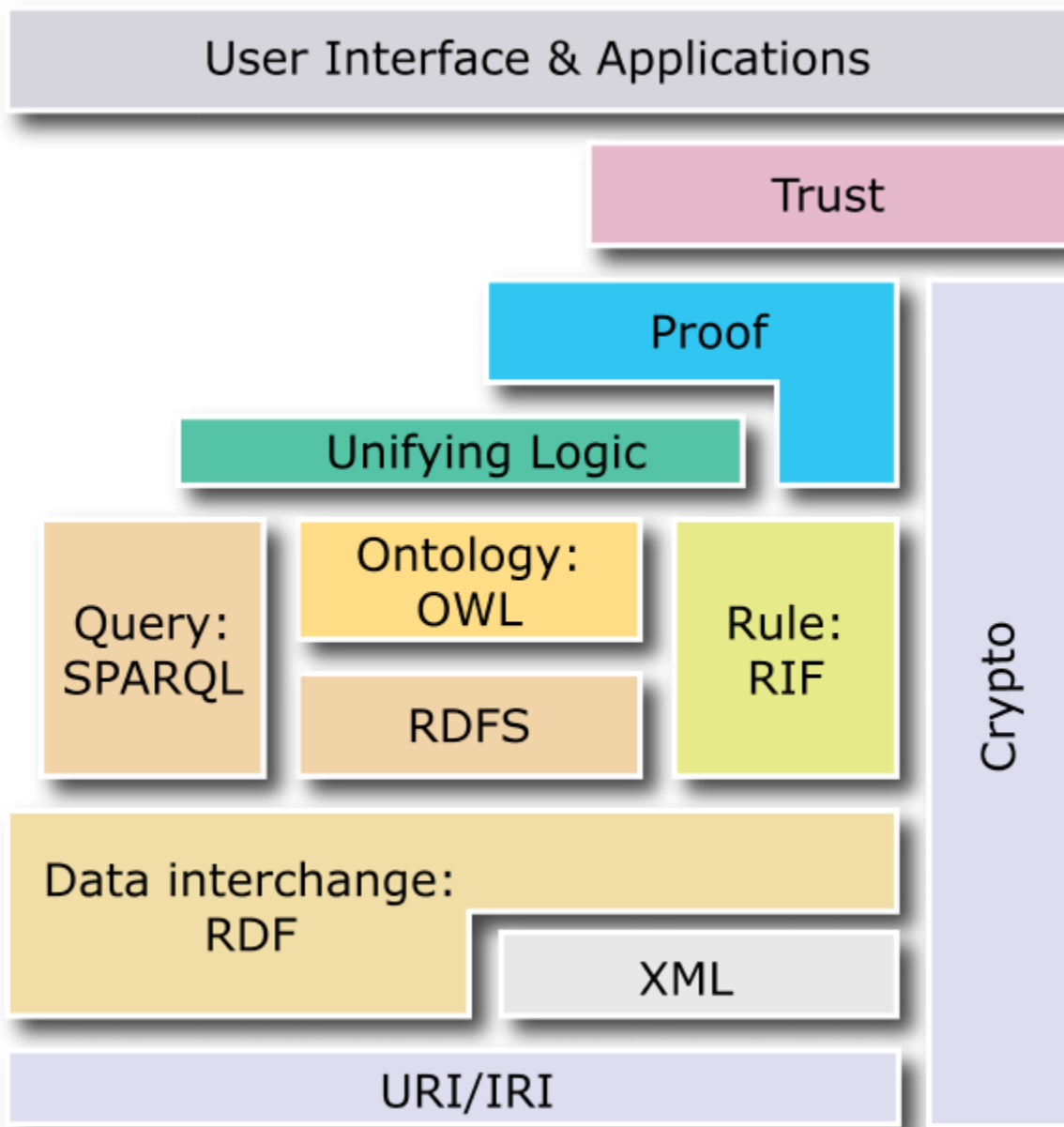


Illustration 11: Semantic Web Layer Cake (© W3C)

Through the next section I will go through the different layers, and discuss some alternative and complementary technologies where appropriate.

## URIs and XMLNS

Uniform Resource Identifiers, URIs, enables the Web by providing a naming scheme. In order to describe things, we need a way to identify them. Uniform Resource Locators or URLs, are constructs most people who have used the Web are familiar with. You type an URL into a web browser, and are taken to a specific website. URLs are in fact one type of URI. The URI thus serves as the foundation of both the current and the Semantic Web because of its use in identifying resources, and needless to say, an important principle of the Semantic Web is the ability to name and describe anything. URIs and XML namespaces provides the information representation constructs needed, including classes, properties and individuals, provide uniformity extensibility and simplicity, and are thus perfect for naming things (resources) in ontologies. A useful thing about URIs when it comes to OWL ontologies and SPARQL querying, is the fact that an URI is divided into two parts. These two parts, called the 'base' and 'fragment' are separated by a '#'. The base URI is the same throughout the ontology, whereas the (optional) fragment identifies different parts of the ontology. Hence instead of writing the full URIs

`http://folk.uio.no/fredrhal/ontology/Data_IT.owl#Object_Oriented`

`http://folk.uio.no/fredrhal/ontology/Data_IT.owl#Scripting`

`http://folk.uio.no/fredrhal/ontology/Data_IT.owl#Teaching`

etc., we can use the shorthand notation of relative URIs as '#Object\_Oriented', '#Scripting' and '#Teaching' using the `xml:base` attribute.

## XML

Extensible Markup Language evolved from SGML and became a W3C recommendation in 1998. XML is a structured, tag-based metalanguage made for defining languages. It is a metalanguage because it allows authors to create tags that specify the structure and syntax of documents. XML is widely used for a number of applications as well as web data exchange. Some examples are program configuration files, recipe collections, Scalable Vector Graphics, musical notation and RSS web feeds. More importantly, XML provides a concrete syntax for OWL, Web Ontology Language, which is the most used language for creating ontologies for use in the Semantic Web. Although XML provides the syntax for content structure in documents, it doesn't provide any semantics with the meaning of this content.

## XMLS

XML Schema, a W3C recommendation in 2001, accompanies XML documents providing rules these must conform to in order to be 'valid' according to this schema. XMLS is written in XML syntax, and is therefore easy to reuse, and extensible through additional XMLS documents. XMLS also brings a set of defined 'data types' for use in XML documents. They restrict values and associate representations with values. The data types part of OWL are:

- string
- boolean
- decimal
- float
- double
- dateTime
- hexBinary
- base64Binary
- anyURI

Although the number of data types allows for quite some expressibility, it is still not expressive enough to express the kind of semantic relations we want.



## RDF

The RDF Specifications are build on URI and XML technologies. The W3C RDF suite of specifications consist of:

- RDF/XML Syntax Specification
- RDF Vocabulary Description Language 1.0: RDF Schema
- RDF Primer
- Resource Description Framework (RDF): Concepts and Abstract Syntax
- RDF Semantics
- RDF Test Cases

Represented in XML, the Resource Description Framework uses triplets to make metadata statements about resources on the 'subject predicate object' form. The subject, or first element of the triplet, denotes the *resource*, the predicate denotes *properties* of the resource and the relationship between the subject and the object, the latter representing the 'property value' of the resource. An example can be:

[The Moon][isMadeOf][Cheese]

Here, the resource, the thing we want to talk about, is [The Moon]. In RDF, every resource has a URI, an unique identifier. It may be a URL, but not necessarily. The property describes the relation between resources, in this case that our resource [The Moon] has the property [isMadeOf], and this value is [Cheese]. Together they form an RDF statement. This triple can also be thought of as a logical formula:

$P(x,y) = \text{isMadeOf}(\text{theMoon}, \text{cheese})$

where P is a binary predicate relating the object (subject) x to the object y. This is a core fact of RDF, it only offers binary predicates.

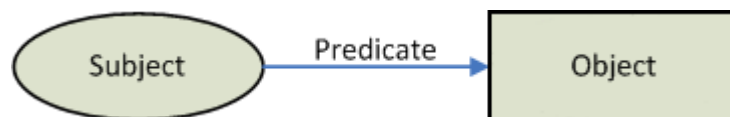


Illustration 12: RDF graph example

RDF is often illustrated through graphs (above), but to provide a standardised syntax that's readable for both humans and computers, RDF is serialised through XML. A viable representation of the triple above could be:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:theMoon="http://dbooth.org/2007/moon/">
  <rdf:Description rdf:about="http://dbooth.org/2007/moon/">
    <theMoon:isMadeOf>Cheese</theMoon:isMadeOf>
  </rdf:Description>
</rdf:RDF>
```



Illustration 13: RDF graph auto generated by W3C validator

This example was validated by the W3C validator<sup>18</sup>, thanks to David Booth for making a URI declaration for the Moon.

## Complex RDF statements

RDF statements wouldn't be very valuable appearing by themselves. The real power of expressibility comes when joining statements together to form relation structures, tying together resources.

**Nested descriptions** is where a property value is another resource. This is the most common construct, and here is an example depicting the statement “Fred has a friend called John, who lives in Rome”:

```
[Fred][hasFriend][John]
[John][livesIn][Rome]
```

**Anonymous nodes**, also called blank nodes or b-nodes, are resources which isn't identified by an URI, and are usually assigned by the RDF processor. The anonymous node can be used to represent the object in one statement to the subject in another, for statements like “Fred has a friend who lives in Rome”:

```
[Fred][hasFriend][_node1]
[_node1][livesIn][Rome]
```

**RDF containers** are used for collection of things, and RDF defines three resource types to hold such collections, along with properties to relate the collection to its members. The different types are *Bags* for unordered resources which may contain duplicates, *Sequences* containing ordered resources, and *Alternatives*, where only one of the members can be selected.

**Reification** is a mechanism that allows us to create statements of statements such as

```
[Fredrik][believesIn][moonCheeseStatement]
```

where the [The Moon][isMadeOf][Cheese] statement is *reified* into a [moonCheeseStatement] resource. This construct is best avoided if possible, as it is easy to wrongfully define resources this way.

## RDFS

Representing statements in this way is flexible and gives us some expressiveness, but we need constraints for it to become really useful. Take this example for instance:

<sup>18</sup> <http://www.w3.org/RDF/Validator/>

[John][motherOf][Jane]

This is a perfectly well-formed statement, but semantically, it's not valid. RDFS, RDF Schema, provides information and constraints on RDF statements, by introducing classes, is-a relationships between classes and properties, and domain and range restrictions for properties. With these tools at hand, it's easy to to define the classes Male and Female, and constrict the motherOf property only to apply to Female class instances. The name 'RDF Schema' is a bit misleading, as opposed to XML Schema which constraints the structure of XML documents, the RDF Schema defines the vocabulary used in RDF data models. We still need more expressive power to express that the Male and Female classes are disjoint, and this is where OWL comes in.

## RDFS constructs

### Classes and subclasses

The construct `rdfs:Class` allows to declare a resource as a class for other resources. Typical example of an `rdfs:Class` is `foaf:Person` in the FOAF vocabulary. An instance of `foaf:Person` is a resource linked to the class using an `rdf:type` predicate, such as in the following formal expression of the natural language sentence: 'John is a Person'.

ex:John `rdf:type` foaf:Person

The definition of `rdfs:Class` is recursive: `rdfs:Class` is the `rdfs:Class` of any `rdfs:Class`.

The construct `rdfs:subClassOf` allows to declare hierarchies of classes. For example, the following declares that 'Every Person is an Agent':

foaf:Person `rdfs:subClassOf` foaf:Agent

Hierarchies of classes support inheritance of a property domain and range (see definitions in next section) from a class to its subclasses.

### Property domain and range

- `rdfs:domain` of an property declares the class of the subject in a triple using the property as predicate
- `rdfs:range` of an property declares the class or datatype of the object in a triple using the property as predicate

For example the following declarations are used to express that the property *employer* is linking a subject which is a *Person*, to an object which is a *Organisation*:

ex:employer `rdfs:domain` foaf:Person

ex:employer `rdfs:range` foaf:Organisation

## N3

Notation3, or N3 as it is more commonly known, is a shorthand non-XML serialisation of RDF models, designed with human-readability in mind. N3 is much more compact and readable than XML RDF

notation. The format is being developed by Tim Berners-Lee and others from the Semantic Web community. N3 has several features that go beyond a serialisation for RDF models, such as support for RDF-based rules. Turtle is a simplified, RDF-only subset of N3. (wikipedia)

Here is the cheese-moon example in N3 format, converted via the Mindswap RDF converter<sup>19</sup>:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix theMoon: <http://dbooth.org/2007/moon/> .
theMoon: theMoon:isMadeOf "Cheese" .
```

## OWL - Web Ontology Language

OWL is a W3C project trying to standardise an ontology-framework more capable than RDFS. OWL can be seen as the next version of DAML+OIL, as it is built upon this syntax. DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) is an ontology project developed by DARPA (US Defense Advanced Research Projects Agency), and OWL development is done by many of the same people who developed DAML. It is layered on top of RDF, and can be used to describe the set of facts that makes up an ontology. It uses RDF namespaces to integrate the many different and incompatible ontologies that exists.

### Three flavours of OWL

The primary goal of OWL is to be able to describe concepts and relationships as unrestricted as possible, but also allow computers to infer and reason about them. If OWL were to put no restrictions on RDF statements, the advantage of computability and decidability would be lost, and so OWL is divided into a family of three languages; Full, DL and Lite. OWL Full is the complete language, while the others are subsets or restrictions of OWL Full. The reason for three versions is that OWL Full is the union of OWL syntax and RDF, and allows an 'RDF datastore' (a collection of RDF statements), and is complex enough to give a logical reasoner serious computational problems. The two subsets are less expressive, but reduces the demands on processing drastically. OWL DL supports Description Logics, a logic family applying carefully selected restrictions on what can be expressed, to gain computability. This makes sure a DL-processor can execute a computation, and be sure to get an answer, hence, it is complete and decidable. From the OWL Language Reference document:

"In particular, the OWL DL restrictions allow the maximal subset of OWL Full against which current research can assure that a decidable reasoning procedure can exist for an OWL reasoner."<sup>20</sup>

---

<sup>19</sup> <http://www.mindswap.org/2002/rdfconvert/>

<sup>20</sup> <http://www.w3.org/TR/owl-ref/>

OWL Lite has even more limitations, and the idea is to make it easy to get going when creating a system, easy to implement processors for, and easy to learn. This way, one can start using OWL right away, and advance to a higher level OWL for more complicated tasks if needed.

OWL uses RDFS classes and properties in addition to newly defined ones, and OWL modifies the definition of a class. An OWL-class is mainly the same as in RDFS, but in DL and Lite instances of a class needs to be individuals, and can not be other classes. In OWL Full and RDFS instances of a class can also be classes.

## **OWL = RDFS**

As RDFS, OWL uses RDF to express everything, something that works both ways. Any RDF-graph is also valid OWL Full, though OWL not always assign specific meaning to all resources. OWL Full puts no restrictions on RDF statements, whether they use OWL constructs or not. It is possible to create statements in RDF that is not valid OWL-DL or OWL-Lite. OWL DL uses all constructs in Full, but applies restrictions on their use. In the same way, a set of RDF statements is compatible with OWL-Lite also compatible with OWL-DL, but not the other way around. OWL DL introduces the instance rule mentioned above, but in OWL-Lite cardinality restrictions can only have 0 or 1 as a value, for instance.

## **OWL Ontologies**

The data described by an OWL ontology is interpreted as a set of "individuals" and a set of "property assertions" which relate these individuals to each other. An OWL ontology consists of a set of axioms which place constraints on sets of individuals (classes) and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. For example, an ontology describing families might include axioms like

- "isFemale" and "isParent" it can be inferred that the individual also has the property "isMother"
- "hasFather" and "hasBrother" and a rule that states that "if individual X hasFather Y, and Y hasBrother Z, then X hasUncle Z", uncle relations can be inferred without being explicitly stated

## **Open World assumption**

Semantic Web languages such as RDF(S) and OWL operates within the Open World assumptions. Open world assumption apply within the field of knowledge representation to express the extent to which knowledge within a system is viewed to be complete, as opposed to Closed world assumption found in XML and SQL. What this means in practice, is that OWA assumes that a system's knowledge is incomplete, so that if a statement cannot be inferred from what is expressed in the system, then it still cannot be inferred to be wrong or false, and are instead considered unknown. Relating this to the Semantic Web, the absence of a particular statement within the Web means that the statement has just not been explicitly made yet, regardless of whether it would be true or not.

## Description Logic

In order to ensure clear definitions and standardised ways to model ontologies, OWL has an explicit logical basis for the language, based on Description Logics.

Description Logics (DLs) are a family of logics that are decidable fragments of first-order logic. In computational linguistics, decidability and computability are important factors, as usually when processing information on the Web we want to be able to get a result. Reasoning in first order logic can lead to undecidability, the situation where the reasoning process is not necessarily going to terminate. By applying constraints on first-order logic, one can get subsets that are both computable and decidable.

## Letters of DL

The expressibility of description logics are denoted through various letters, and the combination of these make up different DL languages. Here is an overview:

1. AL – attributive language. This is the base language, which allows atomic negation, concept intersection, universal restrictions and limited existential quantification
2.  $FL^-$  - a sub-language of AL, disallowing atomic negation
3.  $FL_{\circ}$  - a sub-language of  $FL^-$ , disallowing limited existential quantification
4. C – complex concept negation
5. S – abbreviation for AL and C with transitive properties
6. H – role hierarchies (rdfs:subPropertyOf)
7. R – limited complex role inclusion axioms (reflexivity and irreflexivity, role disjointness)
8. O – nominals (enumerated classes of object value restrictions - owl:oneOf, owl:hasValue)
9. I – inverse properties
10. N – cardinality restrictions (owl:Cardinality, owl:MaxCardinality)
11. Q – qualified cardinality restrictions (OWL 1.1, cardinality restrictions can have other fillers than owl:thing)
12. F – functional properties
13. E – full existential qualification (Existential restrictions that have fillers other than owl:thing)
14. U – concept union
15. (D) – use of datatype properties, data values or data types

The Semantics for OWL are given through translation to a particular DL. Therefore OWL is both a Syntax for describing and exchanging ontologies and has a formally defined Semantics that give the meaning. For example, OWL-DL corresponds to the SHOIN(D) description logic, whose worst case computational complexity is non-deterministic exponential time (NEXPTIME), while OWL1.1

corresponds to the SHROIQ(D) logic. In addition, sound, complete, and terminating reasoners (i.e. systems which are guaranteed to derive every consequence of the knowledge in an ontology) exist for many DLs including those underlying OWL. (wikipedia)

## Modelling in Description Logics

In Description Logics, we make a distinction between the TBox (terminological box) and the ABox (assertional box). The TBox contains sentences describing concept hierarchies (relations between concepts) while the ABox contains sentences stating where in the hierarchy individuals belong (relations between individuals and concepts).

TBox examples:

1. Every CV has at least one competency
2. Every competency has a competency level

ABox examples:

1. Bob has the competency 'Java programming' with competency level 3 to 5 years
2. Bob has a bachelor degree

Syntactically, these are expressed through rules in OWL, and look like this:

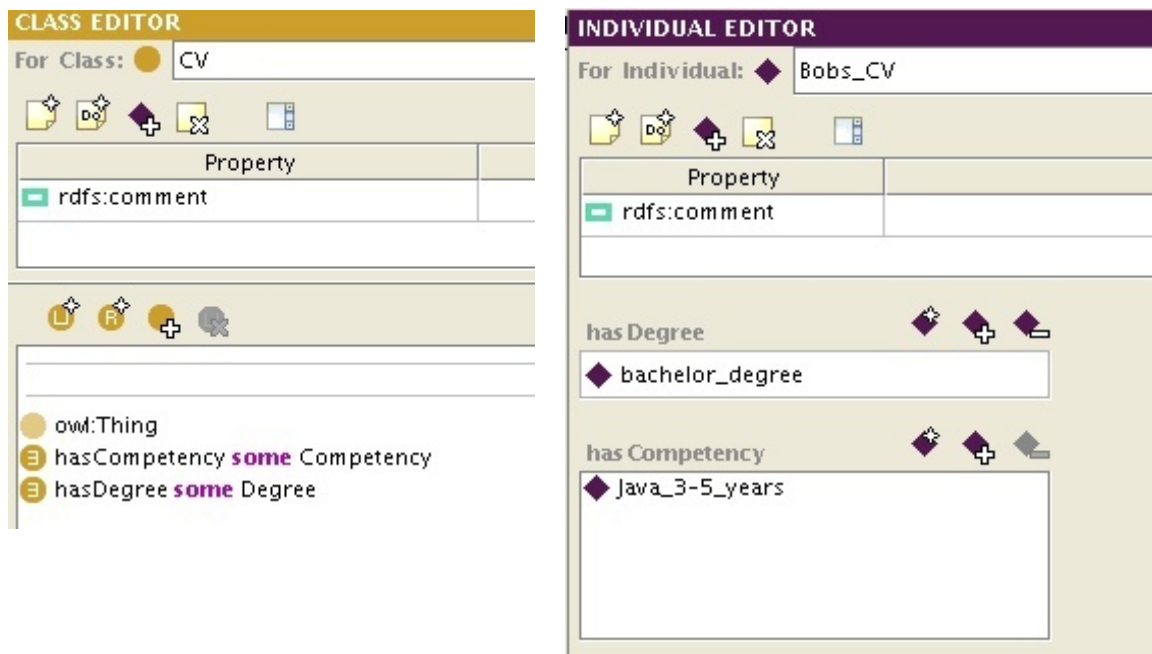


Illustration 14: OWL screenshots

## Description Logics syntax

I won't go into detail about Description Logics syntax, formalisms or algorithms, as this would be outside the scope of this thesis. I will instead show some examples of how RDFS and OWL is expressed (Horrocks2005).

### Ontology axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent <sup>-</sup>
transitiveProperty	$P^+ \sqsubseteq P$	ancestor <sup>+</sup> $\sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN <sup>-</sup>

### Class/Concept constructs

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \dots \wedge C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \dots \vee C_n$
complementOf	$\neg C$	$\neg$ Male	$\neg C$
oneOf	$\{x_1 \dots x_n\}$	{john, mary}	$x_1 \vee \dots \vee x_n$
allValuesFrom	$\forall P.C$	$\forall$ hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	$\exists$ hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leq nP$	$\leq 1$ hasChild	$[P]_{n+1}$
minCardinality	$\geq nP$	$\geq 2$ hasChild	$\langle P \rangle_n$



## Semantic reasoner

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalises that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language, and often a description language. Many reasoners use first order predicate logic to perform reasoning; inference commonly proceeds by forward chaining and backward chaining. (wikipedia)

Regarding the Open World assumption, a deductive reasoner can't infer that the statement is false even if the statement is absence. Thus the fact can't be proven to be false, it is merely unknown.

## Reasoners

- **FaCT++** is an efficient Description Logic reasoner compatible with OWL DL, jointly developed by Ian Horrocks and Dmitry Tsarkov in C++.<sup>21</sup>
- **Pellet** is a reasoner developed in Java, based on the tableaux algorithms for expressive Description Logics (DL), Pellet supports the full expressiveness of OWL DL, including reasoning about nominals (enumerated classes). As of version 1.4, Pellet supports all the features proposed in OWL 1.1, with the exception of n-ary data types.<sup>22</sup>
- **RacerPro** is a commercial knowledge representation system that implements a highly optimised tableau calculus for a very expressive description logic. It offers reasoning services for multiple T-boxes and for multiple A-boxes as well. The system implements the description logic SHIQ.<sup>23</sup>
- **KAON2** is an open-source ontology management infrastructure targeted for business applications. It includes a comprehensive tool suite allowing easy ontology creation and management and provides a framework for building ontology-based applications. An important focus of KAON is scalable and efficient reasoning with ontologies.<sup>24</sup>
- **Bossam** is an inference engine (a Semantic Reasoner) for the Semantic Web. It is basically a RETE-based rule engine with native supports for reasoning over OWL ontologies, SWRL ontologies, and RuleML rules. Additionally, Bossam includes several expressivity features including: 1) URI references as symbols, 2) 2nd-order logic syntax, 3) disjunctions in the antecedent and conjunctions in the consequent (both via Lloyd-Topor transformation), 4) URI-based Java method attachment, 5) support for both negation-as-failure and classical negation. Bossam loads, performs reasoning over, and answers to the queries over a knowledge set, which can include any combination of the following document types: 1. RDF(S) documents (in RDF/XML or in N3) 2. OWL documents (in RDF/XML or in N3) 3. Bossam rule documents 4. SWRL(+OWL) documents (in OWLX or in RDF/XML)<sup>25</sup>

---

21 <http://owl.man.ac.uk/factplusplus/>

22 <http://pellet.owldl.com/>

23 <http://www.racer-systems.com/>

24 <http://kaon2.semanticweb.org/>

25 <http://bossam.wordpress.com/>

## **DIG**

The DIG Interface is a standardised XML interface to Description Logics systems developed by the DL Implementation Group (DIG). This project provides XML Schemas, Java XML Beans for parsing, creating, and manipulating instances of the schemas and an Java Reasoners API using the these tools to actually communicate with Reasoners communicating over the DIG interface.

## **Rule Languages**

Rules can be used on top of the formal semantics provided by OWL and description logics, taking into account some of the fuzziness contained in natural language. It can also serve as a tool to set up complex conditions for vacancies, or to check for relations that is hard to discover with OWL alone. Below is a survey of some proposed rule languages. Although no standard has been chosen at the time of this thesis writing, RIF is probably the closest to becoming one.

### **RIF**

Rule Interchange Format (RIF) is a W3C recommendation-track effort to develop a format for interchange of rules in rule-based systems on the Semantic Web. The goal is to create an interchange format for different rule languages and inference engines. The RIF initiative is closely related to ontologies. Whereas ontologies describe distributed information objects in a computer executable manner, rules in this sense combine such information and derive new information on top of ontologies.

### **SWRL**

Semantic Web Rule Language (SWRL) based on a combination of the OWL DL and OWL Lite sub-languages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sub-languages of the Rule Markup Language. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sub-languages of OWL. A model-theoretic semantics is given to provide the formal meaning for OWL ontologies including rules written in this abstract syntax. It also gives an XML syntax based on RuleML and the OWL XML Presentation Syntax as well as an RDF concrete syntax based on the OWL RDF/XML exchange syntax. SWRL retains the full power of OWL DL, but at the price of decidability and practical implementations.

As there was no defined standard rule language with an established API when I did my project (although RIF is on its way to become a W3C recommendation), I haven't been able to incorporate the power of powerful soft semantics in my implementation.

## Querying the Semantic Web

For ontologies to be useful in search and matching, it must be possible to query them with specific queries allowing us to define what we are looking for.

### SQWRL

SQWRL (Semantic Query-Enhanced Web Rule Language; pronounced squirrel) is a SWRL-based query language that can be used to query OWL ontologies. SQWRL provides SQL-like operations to format knowledge retrieved from an OWL ontology. SQWRL does not alter SWRL's semantics and uses the standard SWRL presentation syntax supported by the SWRLTab.<sup>26</sup>

SQWRL is defined using a library of SWRL built-ins that effectively turns SWRL into a query language. These built-ins are defined in the SQWRL Ontology. It has the default namespace prefix 'sqwrl'. A copy of this file can be found the standard Protege-OWL repositories, and can be imported through the 'Import Ontology' option in the Metadata tab.

The downside with SQWRL/SWRL is that not many applications support it. At the time I was investigating what technologies I could use in my project, the only reasoner that supported SWRL was Pellet and Bossam, and even those only partially or rudimentary.

### nRQL

The language nRQL augments and extends Racer's functional API for querying a knowledge base (a knowledge base, KB for short, is simply a T-box/A-box tuple (T , A). For instance, Racer provides a query function for retrieving all individuals mentioned in an A-box that are instances of a given query concept (Haarslev2004). As nRQL is part of the commercial Racer API, it was never considered as an option to be part of my project.

### SPARQL

SPARQL is an RDF query language standardised by the RDF Data Access Working Group (DAWG) of the W3C which has been a recommendation since 15<sup>th</sup> January 2008. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middle-ware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can either be results sets or RDF graphs<sup>27</sup>.

After looking at the option of using SQWRL with SWRL, the choice quickly fell on SPARQL. As very few reasoners supports SWRL, I didn't want it to be a part of my ontology, so using SQWRL wasn't really an option anyway. Luckily SPARQL in its current form has all necessary features to do query, update and insert operations, and it is supported by a wide range of applications.

---

<sup>26</sup> <http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL>

<sup>27</sup> <http://xml.coverpages.org/ni2008-01-16-a.html>

## SPARQL syntax

One of the goals when developing SPARQL was that it should resemble the widely known syntax of SQL querying, and so they share many of the same keywords like SELECT, FROM, WHERE, ORDER BY etc. But instead of selecting rows or columns matching specific keywords or IDs from tables, one queries about triplets. One such collection of triplets can be:

```
PREFIX data: <http://www.fredont.com/freds\_silly\_ontology#>
```

```
SELECT * WHERE { ?object data:isMadeOf data:Cheese }
```

would match our RDF example earlier, and could for example return :

```
+-----+
|  object  |
+-----+
|  TheMoon |
+-----+
| CheeseCake |
+-----+
```

if we choose to get the result in a traditional table format. In the WHERE clause, a set of N-Triples define the query pattern of a query, and the ? notation represents variables. To address specific matches in a ontology, we use prefixes, denoting what ontology and where to find the node.

For a more interesting example, we could query the ontology for something like:

```
SELECT * WHERE { ?subClasses rdfs:subClassOf ?superClasses }
```

In the example above we would get all subclasses that have superclasses, since both are free variables. If we want all subclasses of one specific class, we have to specify it directly, again with prefixes:

```
PREFIX data: <http://www.fredont.com/programming\_languages\_ontology#>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT * WHERE { ?subClasses rdfs:subClassOf data:Programming\_languages }
```

this would return all classes being subclasses of the class `Programming_languages`, such as Java, Perl, Python, C# etc. Note that `rdfs:subClassOf` is a built-in rdfs construct, as stated earlier.

SPARQL queries used in my project will be given in chapter 5.

## Migrating from RDBS

As Semantic Web technologies are getting mature, there is a growing need for RDF applications to access the content of non-RDF, legacy databases without having to replicate the whole database into RDF. D2RQ is a declarative language to describe mappings between relational database schemata and OWL/RDFS ontologies. A research group at the Free University of Berlin headed by Christian Bizer has worked on this database-to-RDF mapping since 2002. The D2RQ Platform uses these mapping to enable applications to access a RDF-view on a non-RDF database through the Jena and Sesame APIs. Its companion, D2R Server, is used by a growing number of Web data sources to provide public SPARQL endpoints and Linked Data views on relational data. Versions of D2RQ have been integrated with the TopBraid Composer ontology tool set and the Gnowsis Semantic Desktop. D2RQ is licensed under the terms of the GNU General Public License, and other licenses are granted upon request.<sup>28</sup>

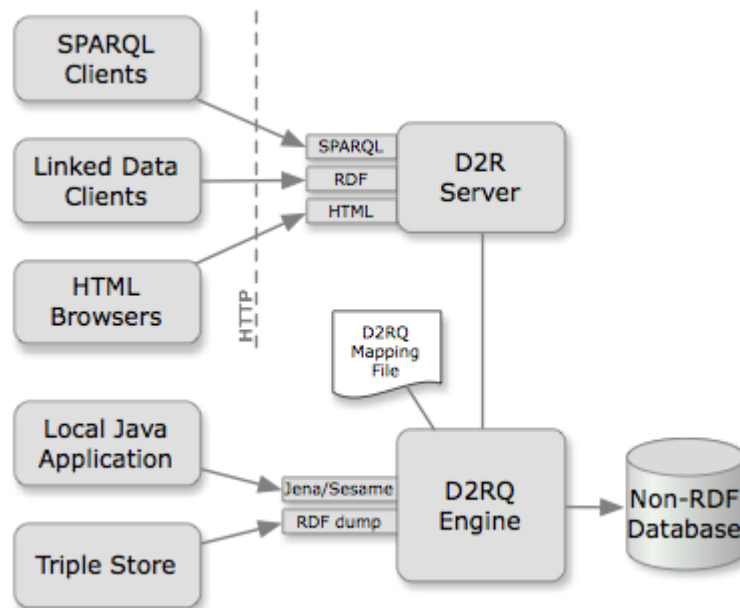


Illustration 15: D2RQ application

Using The D2RQ Platform you can:

- query a non-RDF database using the SPARQL query language,
- access information in a non-RDF database using the Jena API or the Sesame API,
- access the content of the database as Linked Data over the Web,
- ask SPARQL queries over the SPARQL Protocol against the database.

D2RQ has been tested with Oracle, MySQL, and PostgreSQL, and works with any SQL-92 compatible database. There are however, some fundamental limitations with D2RQ. First off it is read-only, which means no support for CREATE, DELETE or UPDATE queries. It has very little support for inference, and it requires external tools to do integration of multiple databases or other data sources.

<sup>28</sup> <http://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/>

## 4. Research Method

In the Merriam Webster online dictionary, research is defined as “studious inquiry or examination; *especially* : investigation or experimentation aimed at the discovery and interpretation of facts, revision of accepted theories or laws in the light of new facts, or practical application of such new or revised theories or laws”, whereas yourdictionary.com states “careful, systematic, patient study and investigation in some field of knowledge, undertaken to discover or establish facts or principles”. Traditional research methods are usually based upon gaining knowledge about an existing phenomenon, where one chooses one or more research methods best fit for the problem. The choice of research methods also affects the ability to generalise the results, like whether there is qualitative and/or quantitative data available.

### **System development**

In some fields of research however, collection of empirical data can be difficult, but even implicit data, like data embedded in a system design or implementation can add value to the field. To develop a Semantic Web system, one doesn't only need to understand the Semantic Web, one also needs to know how to use it. As I am creating a system from scratch myself, I don't just gain knowledge about a phenomenon and the development process concerning it, I also am part of creating or extending to the body of the phenomenon, and also propose suggestions on how to solve it. The solution itself needs to be critically examined, but doing system development should be recognised as a valid research method as it can produce valuable results in itself. The dilemma when using development as a research method is the risk ending up with a result that is only satisfying my needs within this thesis, or maybe not even that. There's no guarantee that the solution satisfies the actual needs of the target area, nor satisfies these needs in a qualified manner. Using scenarios and possible use cases gives a better chance of the results being useful, as solutions to given scenarios might be generalisable.

Using system development as a research method will also involve several other research methods. To gain knowledge about the area of interest you conduct document analysis, document the development process, justify choices, plan and execute testing, as well as use methods to analyse and compare the results.

## **Stages of systems development research**

Following Helen Hasan (Hasan2004), system development research can be divided four main stages, and these will be briefly discussed in the next section.

### **Concept Design**

In the first stage, a survey of 'state of the art' needs to be conducted in the area of interest, and the researcher needs to adapt to the current advances. The researcher must gather existing knowledge about the area to identify possibilities of improvement, and develop a research objective. This stage often involve a substantial literature review, and perhaps discussions with practitioners and researchers in the field. Finding material that is 'up to date' can be a difficult task, and the researcher needs to be aware that research papers and projects quickly becomes outdated.

### **Architecture**

Constructing the architecture of the system is the next stage, and Hasan argues that this should accepted as genuine research, as it contributes to knowledge creation. The researcher engages in creative design activities of architecture development, creating models, designing algorithms and data structures, as well as defining core components.

### **Prototyping**

This is the stage where so called 'proof of concept' is often used to demonstrate that a system can in fact be built based on the results from the previous stages. Prototyping can be done in several ways, with a single working prototype, or in a iterative process, analysing, designing and implementing an evolving prototype. Learning occurs throughout the system building process, and insight is gained about the problem and the complexity of the system. In many cases, this is where the research stops at this stage because the system fails to meet expectation or is not feasible to be developed further.

### **Development**

This is where theory and prototyping formalise into a system specification, and deployment work begins. At this stage of research it's not unusual to have a company sponsoring the project, being interested in adopting the system being produced. This often involves commercialisation, project knowledge become confidential, and technology is scaled for general use.

In the last two stages (if the last one is being realised), it's possible to evaluate the use of the system with case studies and testing, compiling experiences learnt and developing new theories of use, which may lead to another research cycle, going back to the first stage. From a research perspective, failure might still be a valid research outcome, with gained knowledge of things that won't work, and thus still a contribution to knowledge in the field. In the next section I will go into more detail in my particular procedures.

## **Background knowledge and motivation**

The decision to do something based on computational linguistics related to the Web came early on. I've taken several courses on both logic and computer linguistics, but none of them explored the possibilities offered by the internet. Discussing with my supervisor if there was a field which exploited the Web and XML in computational linguistics, he suggested I looked into web ontologies and description logics. At the same time, I was working part-time in StepStone, and had done some work on the vacancy categories under which they publish the various vacancies, a process done mostly by hand. This being a manual job, it is entirely dependant on the skills of the people publishing the vacancies if the vacancy is published under the correct category. Conducting a search for some specific job positions, it was soon revealed that it isn't always easy to categorise the vacancies, with the fuzziness of free form descriptions.

### **Finding a core case**

My initial idea was thus to create a prototype system that given a vacancy text would suggest what category it should be published under. This would need some kind of competency ontology, and a text parser to extract the competencies from a vacancy text. Realising after reading some papers and books on text mining (Cimiano2006) that I would probably not achieve anything substantial in the given time. I lacked a lot of the theory required, and most of the programming would be in a functional language like Lisp and not a web based one like I wanted. Already having started the work on the competency ontology, I looked at other possibilities of improving the vacancy posting and got the idea of finding CVs that would match a specific job ad. After working on the ontology and reading about other semantic based HR systems, discovering the match score figure presented in the first chapter, I decided to incorporate this ranking functionality in my system as well.

Another issue noticed when doing job searching using some of the major companies in the business, was that the search result was often way off, and returned results that had nothing in common with the keywords entered. I wanted to look deeper into what could improve the search results. Most job board search is based on word matching between what the user had typed in the text area and the vacancy contents, something that could be improved with Semantic Web technology.

The next section will go through the initial phases revolving gaining knowledge about the field of the Semantic Web, as well as the work throughout the thesis writing.



## **Research resources & data collection**

An important phase of the research is gathering enough data to get the best image possible of how to improve the area of interest. Articles, books and online resources plays an important role, but also thoughts from the intended users and qualified persons (like my supervisor), W3C recommendations and roadmap documents, as well as similar projects.

An obvious source of data came from testing various online job boards. I studied the results returned from searches done at the largest online job boards here in Norway, namely Finn, Monster, StepStone and NAV. Although most of them supply free keyword search, it is apparent that the best results comes when narrowing down to specific categories using drop downs. The category division level varies among the different sites, but usually consists of a main category followed by a list of subcategories, making up 2 levels.

## **Gaining knowledge about the field**

An important principal within the kind of literary work a master thesis concerns, and research in general, is to acknowledge that it is a collective effort. When writing a master thesis, you use other people's work as your starting point, trying to find new knowledge on the subject. It is important to start gathering documentation about the chosen topic as soon as possible, and get a good overview of the state of the art.

## **Selection, availability and validity**

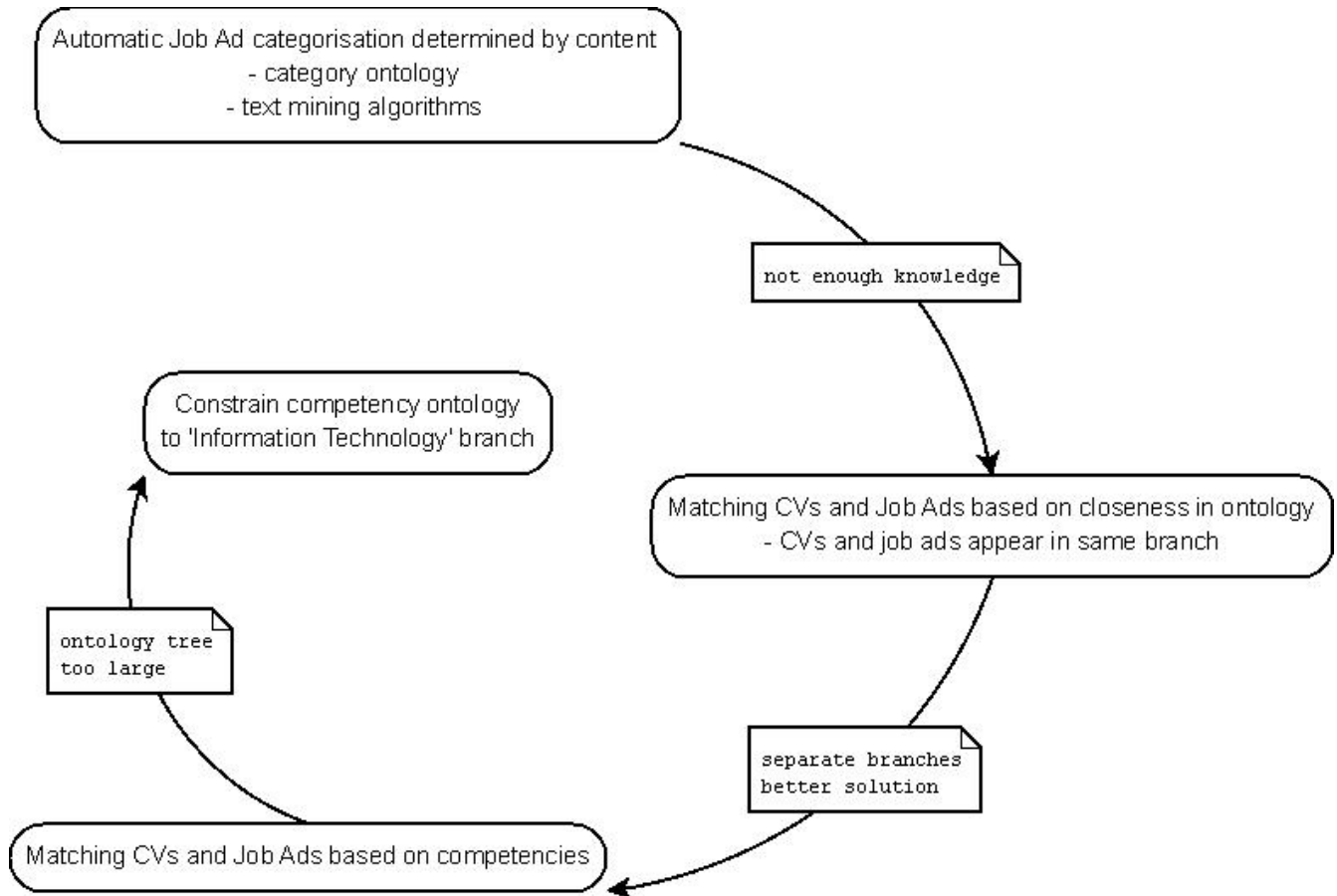
In addition to the initial recommendations from the supervisor, I started looking for articles online. Through Google and wikipedia I found a lot of articles on both ontology design and implementation in general, as well as Semantic Web projects specific to HRM. It's important to stay critical when reading articles, as this is a fast moving field, and knowledge discovered just a year ago can be outdated. A good 'sign of validity' is names that are reappearing in many articles. This shows that the authors are active within the field, and it's often easy to read articles chronologically and notice progress and new discoveries from one text to the next. Another source that is reliable is the documentation from large organisations in the field, like W3C, HR-XML Consortium, IBM, Vrije Universiteit Brussel etc., and documentation from the developers of software currently used, like Jena, Sesam, Protégé and Eclipse.

## **Project specific knowledge**

I already had an idea about doing something within computational linguistics within the HRM domain, as I was working part time in StepStone. This gave me the advantage of knowing how an online recruitment tool works from the inside, from the user interface to the actual source code. After getting an external supervisor in Computas I also got the benefit of having some of the leaders in Semantic Web technology usage here in Norway, with active participants in W3C among other things. Having this solid foundation assured me that I could trust the knowledge passed on to me by the surrounding competent persons.

## Delimiting the field of interest

Here I will just briefly reiterate the progress of delimiting the core problem in my thesis as a 'delimiting spiral':



*Illustration 16: delimiting spiral*

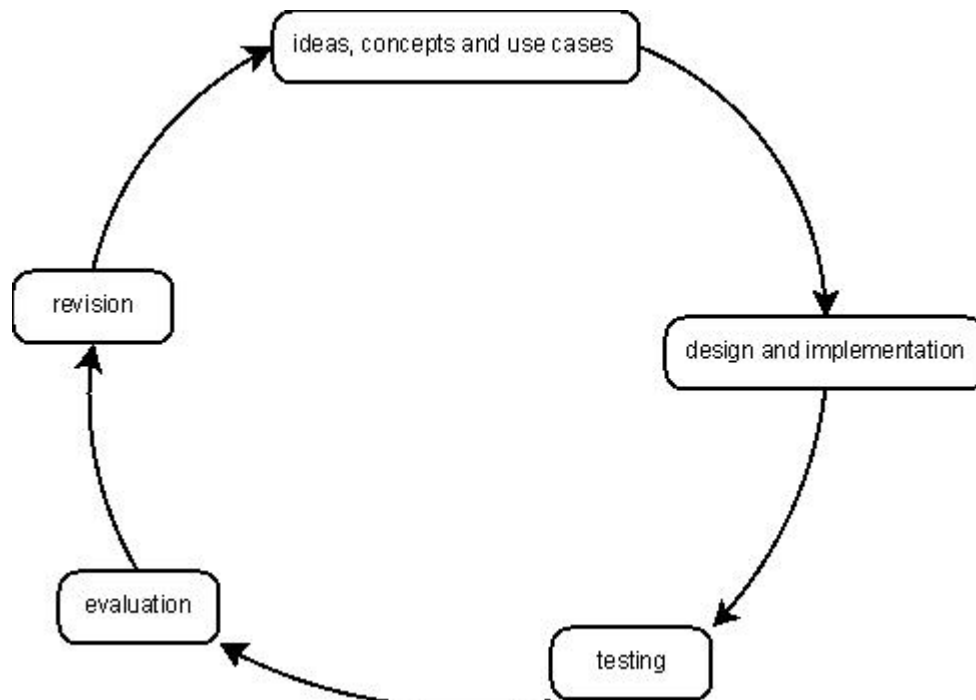
## Project plan

Here is a rough schedule of the project and thesis writing, more details on the implementation will be given in chapter 5:

February 2007	Initial book, article and internet research
February 2007 – May 2008	gaining knowledge about the field through reading books and articles
December 2007	setting up Protégé and following the tutorials
January 2008 – July 2008	designing and populating ontology
January 2008	accessing ontology through command line Java with IODT
February 2008	Migrate from IODT+Pellet to Jena
March 2008 – June 2008	designing and implementing web interface
March 2008 – June 2008	documenting implementation and ontology design choices
March 2008 – July 2008	writing the main body of the master thesis

## System development method

The ontology and web application development cycle followed a pattern to a certain degree, loosely based on the agile system development method. While the ontology designing was done as a continuously flowing improvement, the web application was developed in clearly defined versions or iterations. Each iteration went through a cycle where ideas, needs and features was planed, designed and implemented, tested, evaluated and revised. An ended cycle resulted in discovery of new needs, which lead to new ideas and thus started a new version development cycle.



*Illustration 17: developement cycle*

Although the system development stopped at version 3 (discussed in the next chapter), there are enough ideas and areas of improvement to develop it further through more cycles. The system as it is now isn't practically usable, but with enough time and effort I think it could at least be used as an environment for trying out better matching algorithms, more advanced queries, inserts etc. This will be discussed in more detail in the next chapter.

## Audience

As this thesis is based around a prototype information system concerning HRM, the target audience could be HR companies looking for a survey on Semantic Web technology in general, or maybe looking for ideas on how Semantic Web technology can improve their CV/job ad matching specifically. It could also be interesting for someone in the Semantic Web community looking at how to model competencies, or how to set up a web based system that utilises Semantic Web technology. A lot of the areas are not limited to the HR domain, and features like the matching and score calculation of RDF collections can be generalised to many fields.

## 5. Ontology design and project implementation

In a master thesis implementation there is not enough time to implement a complete working system that users can use practically. To avoid drowning in implementation work, one has to delimit early on, and choose some key points that one wants to prove the benefits concerning the thesis focus points. This way the implementation can be incomplete, but still illustrate the problems discussed.

The central point in this thesis is the CV/Job ad matching. I wanted my implementation to illustrate the benefits of using OWL and SPARQL to show that there are considerable gains to be had with semantic matching. I also wanted to improve keyword search, in that it takes advantage of the ontology structure as well. By having the sub/super class relationships I could implement a broader search than text matching, and show that even more refined search is possible with OWL properties. This chapter will go through both the design process of the ontology, as well as the implementation of the web interface, queries and matching algorithm.

### Forming an HR Ontology

When creating an ontology model for the HR domain, one has to think about the common knowledge that underlies meaningful conversations about human resources. This knowledge needs to be captured and exploited in process implementations and data stores. The language used to describe jobs, functional roles and staff vacancies is generally well understood and formalised. Standards for the description of tasks and responsibilities are emerging, and these efforts support capturing the combination of tasks and responsibilities that make up a typical job vacancy.

Through the next section I will discuss how I approached ontology design, and some of the key points to consider when forming an ontology.

### What is a competency?

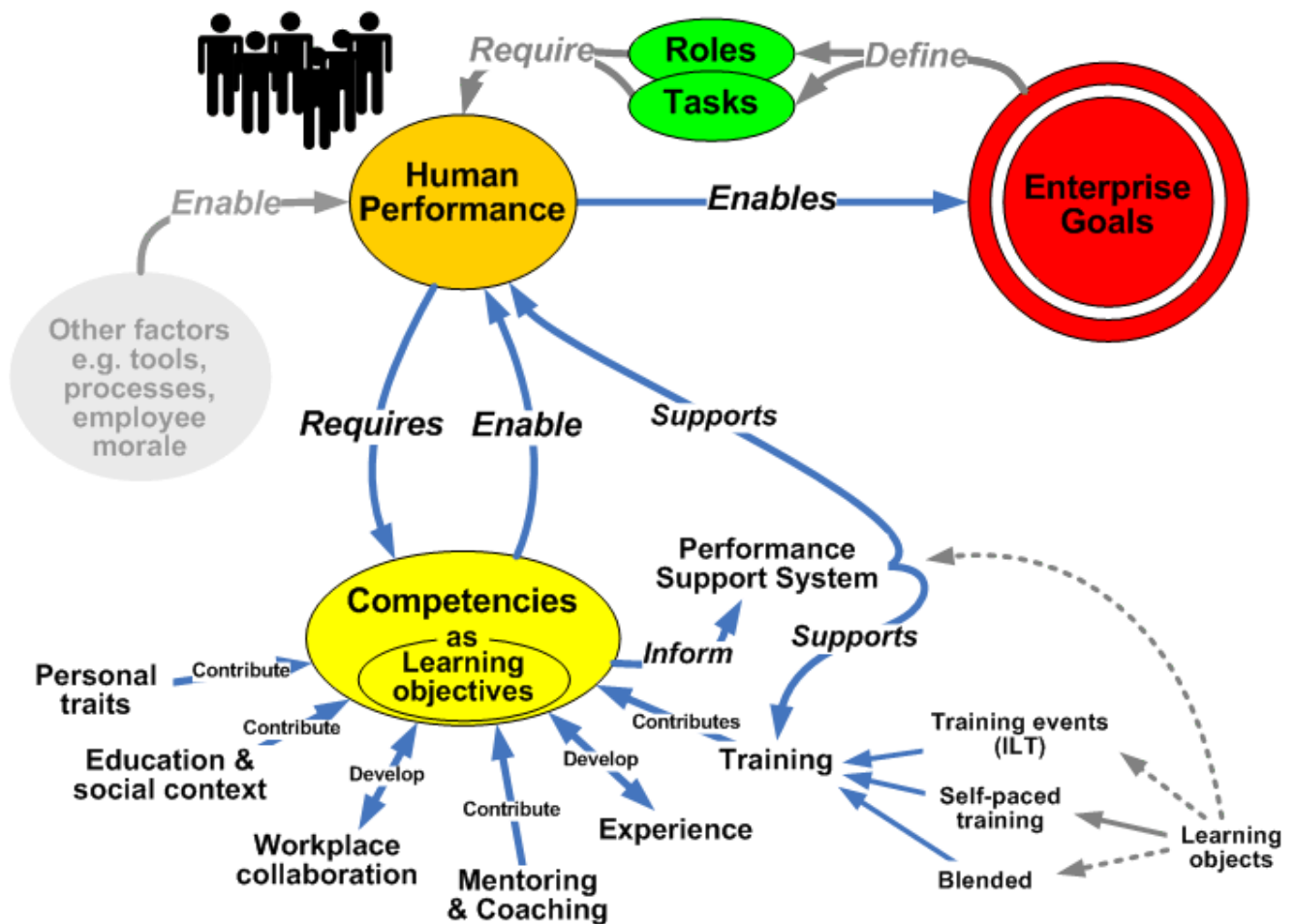
Performing a task requires competence, and (Cheetam2005) defines competence as “effective performance within a domain/context at different levels of proficiency” . The definition given at HR-XML<sup>29</sup> tries to extend the previous one by saying its “a specific, identifiable, definable, and measurable knowledge, skill, ability and/or other deployment-related characteristic (e.g., attitude, behaviour, physical ability) which a human resource may possess and which is necessary for, or material to, the performance of an activity within a specific business context”. While competence is difficult to define in operational terms, there is a general agreement that competence is to a large extent the product of a number of specific competencies. Such competencies must in turn be specified at levels of granularity that support specific operational processes such as targeted assessments, staffing, training or performance support.

Competencies are typically described in a natural language which cannot be properly processed by machines. This means that competency information is usually not available in any formal notation that supports automation, especially for operations like semantic matching.

---

<sup>29</sup> [http://www.hr-xml.org/hr-xml/wms/customers/pdf/HR\\_XML\\_Competencies\\_1.pdf](http://www.hr-xml.org/hr-xml/wms/customers/pdf/HR_XML_Competencies_1.pdf)

This diagram is an attempt to show how learning management, training management and performance support systems can be aligned with business goals as part of an "ecosystem of competencies"<sup>30</sup>.



© 2005, 2006 Claude Ostyn - This work is licensed under a Creative Commons Attribution-ShareAlike 2.5 License (see <http://creativecommons.org/licenses/by-sa/2.5/>)

Illustration 18: learning management (© Claude Ostyn)

30 [http://www.ostyn.com/standardswork/competency/ecosystem\\_of\\_competency.htm](http://www.ostyn.com/standardswork/competency/ecosystem_of_competency.htm)

## Competency modelling

A useful model uses competency definitions as building blocks in competency information modelling and related operations. Defining competencies can be a difficult task - consider a person who is very good at 'defusing conflicts between customers and developers'. This is both a valid and valuable competency, and one that clearly implies a lot of sub-skills and related competencies, some of which are dependent on a specific context such as specific tasks or type of work environment.

“Capturing the semantics of competencies requires semantic models that are specific to particular contexts. The domains covered by such models may be more or less broad, such as a trade or profession, or a task that is performed by many people in a particular context. Such systems must be able to manage millions of competencies and interrelationships, and to support operations on the knowledge itself, rather than just the massaging of fragments of text.” (OOAHR2007)

Although the competency models here are simplifications of the real world, they are a better approximation than the notion of 'knowledge' in traditional knowledge management approaches, as they can represent sets of skills, knowledge and abilities that belongs together. In the articles I have read, I've seen several examples of how these models can look like, here exemplified from the Schmidt & Kunzmann article<sup>31</sup> (CC-GNU LGPL licensed):

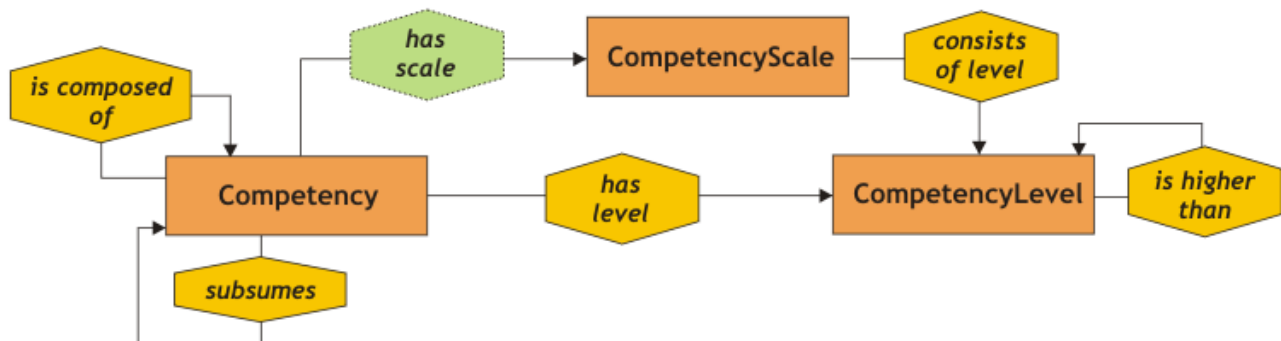


Illustration 19: competency model (© Andreas Schmidt)

31 [http://professional-learning.eu/competence\\_ontology](http://professional-learning.eu/competence_ontology)

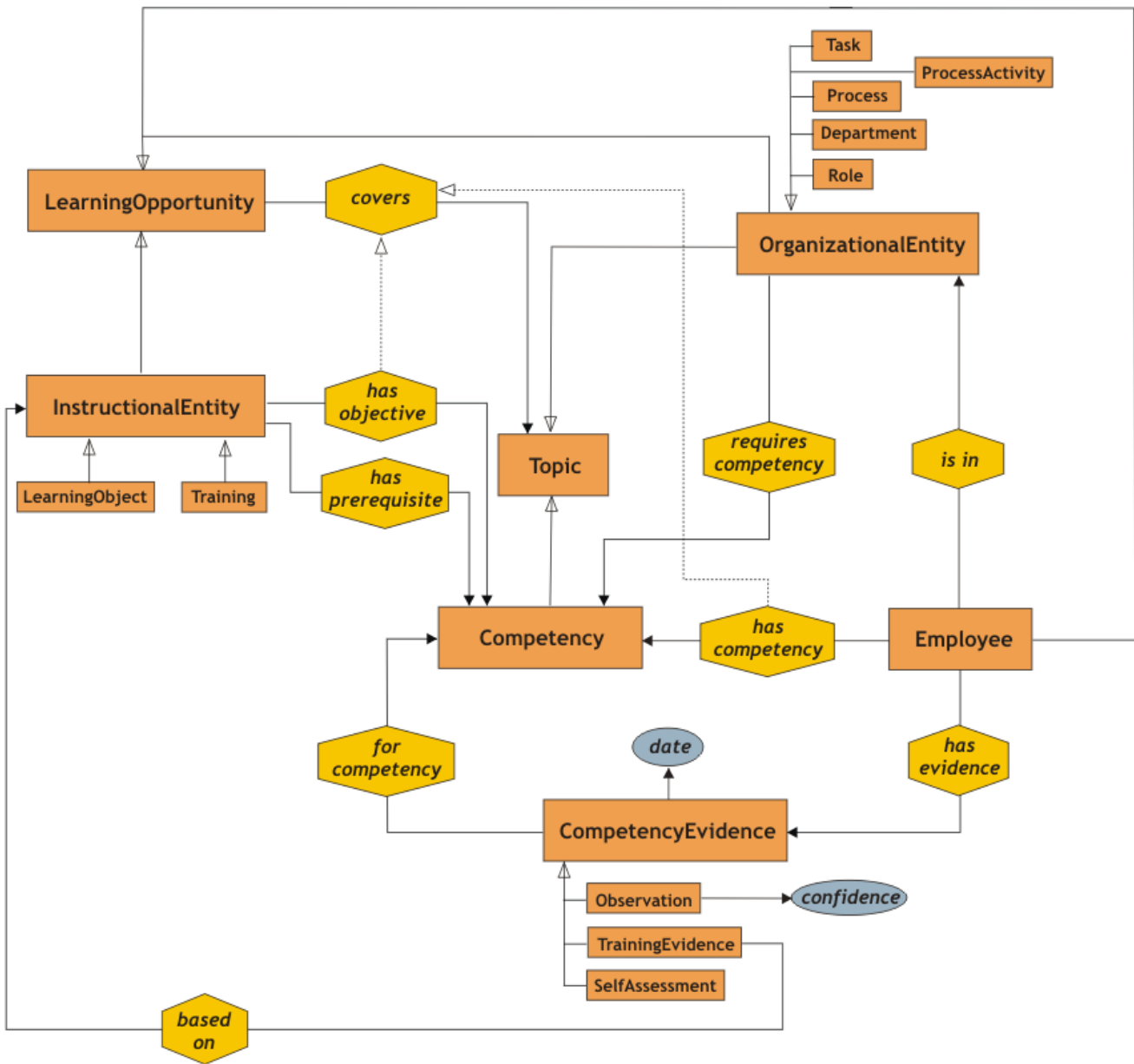


Illustration 20: competency ontology (© Andreas Schmidt)



Aggelos Liapis from VUB STARLab presented this 'mind map' on ontologies at the HCSIT and Maastricht OOA Workshops<sup>32</sup>:

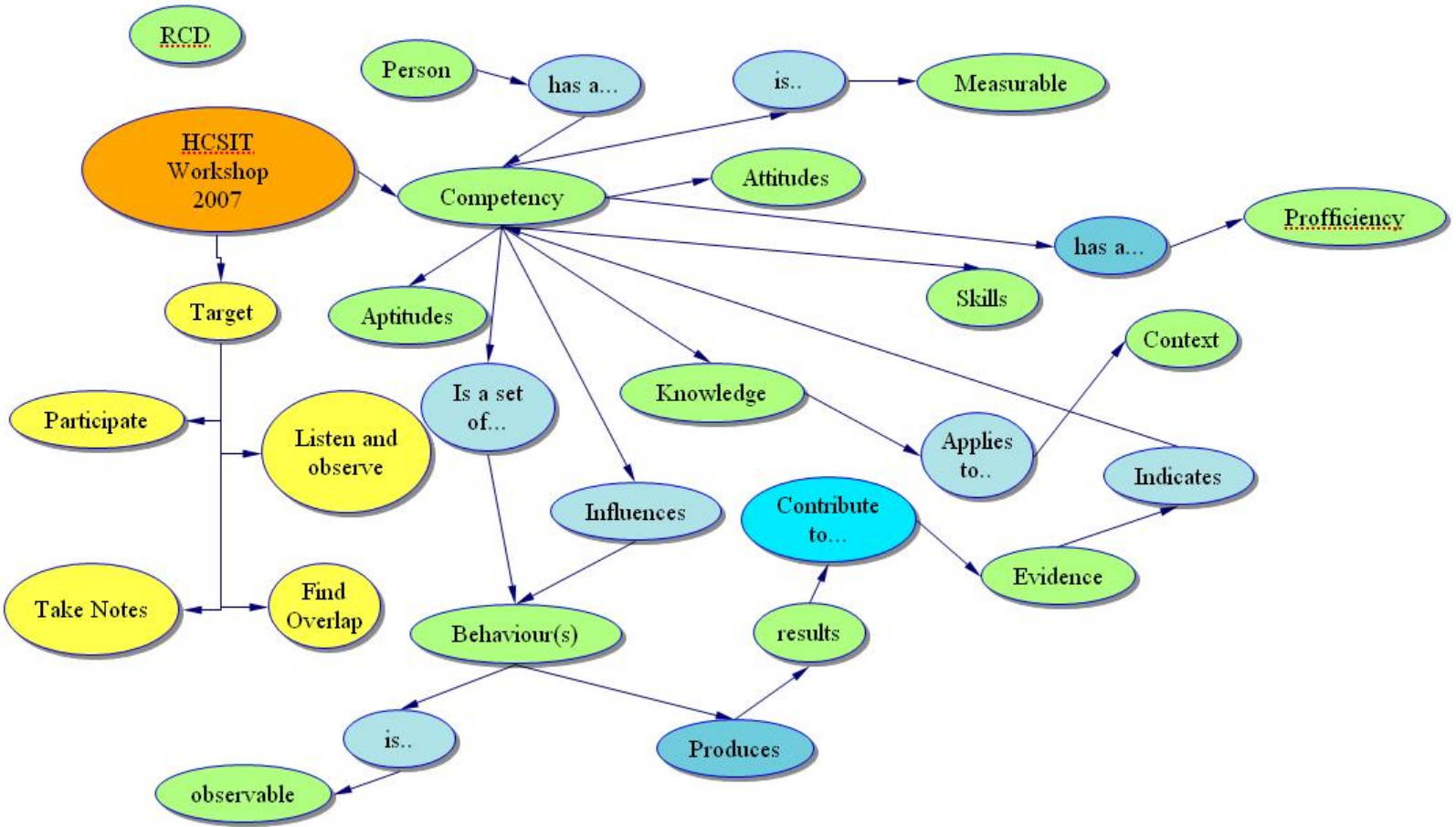


Illustration 21: competency mind map © VUB STAR.Lab

Since this is a master thesis done by one person, my model will not have a high level of complexity, and I have mainly concentrated around the domain of Computer Science. The level of granularity is also limited, as this system is for testing purposes, and proof of concept that semantic matching gives better results than syntactic. A point one should ask oneself is how fine-grained does the model need to be to be useful, and whether this level of complexity is feasible.

32 <http://www.ontology-advisory.org/node/72>

Here is an excerpt from CommOnCV white paper<sup>33</sup>, giving a good description of what a competency is:

A competency is the effect of combining and implementing *resources* in a specific *context* (including physical, social, organisational, cultural and/or economical aspects) for reaching an *objective* (or fulfilling a mission). Three types of resources are distinguished:

1. *knowledge* which includes
  - theoretical knowledge (*e.g.* to know the second law of thermodynamics)
  - knowledge on existing things (*e.g.* to know the operation of test-bed equipment for combustion engines)
  - procedural knowledge (*e.g.* to know the mounting procedure of a particular electronic card)
2. *behaviour* which includes human trait, quality and attitude (*e.g.* leadership, integrity, persuasiveness or adaptability)
3. *know-how* which includes formalised know-how (*e.g.* the application of working procedures) and empirical know-how (*e.g.* trick, ability or talent).

This shows that a competency can be a very broad range of traits, and even though my ontology is limited to competencies within the field of Information Technology, my competency tree will be quite inferior.

---

33 [http://www2003.org/cdrom/papers/poster/p013/Bourse\\_al.html](http://www2003.org/cdrom/papers/poster/p013/Bourse_al.html)

## Ontology design

The central question when starting to design an ontology, or any class-based system for that matter, is 'what classes and properties should the ontology incorporate?'. A good practice when starting to think about modelling a domain, is to do a brainstorm or survey of what key concepts defines the domain. Even looking at material concerning human resource management, picking out reoccurring nouns and terms, generally indicates what components should be part of the ontology.

Here are some frequent occurring keywords :

- vacancy
- competency
- recruitment
- talent
- education
- skills
- company
- CV
- screening
- location
- job agent
- position
- job posting
- candidate

The next step is trying to figure out which ones represents classes and instances of the ontology. There's not one 'correct' model, it much depends on the needs and use of the ontology, but it will eventually become obvious if something doesn't fit while building the ontology.

During the implementation I've changed and added to my ontology model several time since the initial version, and there are many ways to model ontologies. Several factors influenced the choices taken, one the more important issues deciding what should be modelled as classes and what should be modelled as instances, and me and my supervisor had several discussions on what made sense. The next section describes the classes, properties and instances of my ontology.

## Classes and instances

Classes in OWL functions as templates for the instances, and when checking the ontology with a reasoner, makes sure every instance follows the rules which are defined in each class.

### CV

This class is where the CV instances are created. It has rules that says that every instance is required to have a Competency instance, and a Degree instance (minimum cardinality 1). The Description text is stored in `rdfs:comment`. It has no sub- or sibling classes. An important point when modelling ontologies is to keep in mind what you are actually modelling. In that respect, a CV is a persons collection of skills and competencies, so it is natural that a CV belongs to a person. As such, it would be reasonable that an `belongsTo` property is connected to a FOAF entry. I haven't added this connection to my system, but it wouldn't take much effort to implement.

### JobAd

This class is where the Job adverts go. This class consists of approximately 20 subclasses, modelled after Stepstone's categories imported via Protégé. I decided early on to narrow my scope to just a few categories, and considering that most of my lies in the computer science domain, I chose to model that. The actual free form job description is again stored in `rdfs:comment`.

### Competency

The competency class is the superclass for all the competency categories in the ontology. It has a `hasCompetency_level` requirement on all its subclasses. Subclasses include Programming, Leadership, Teaching, Operative\_system and Applications. Each subclass is divided into more subclasses, specifying concepts further. For instance, the Programming class is superclass to `Programming_languages`, which contain class trees for Functional, Object\_oriented, Procedural and Scripting, which again is divided in actual programming languages. This list is by no means complete, not even for my relatively small domain. One thing is the job ad categories which can be fairly standardised, but there is yet to be defined a standard for what competency categories could contain. This contains various competencies a IT-skilled person could have, from experience in Windows NT, being service-oriented towards customers, to programming experience in Lisp.

### Competency\_level

This is the grading of the level of expertise for each competency. When I started modelling this class, I divided it into levels of inexperienced, 1-3, 3-5, 5-8 and 8+ years of experience. This is of course an overly simple grading of competencies. Not all competencies can be just summed up in year-span, but as a coarse classification it is useful. After starting to work on the CV – Job Ad matching code however, I realised that having competency levels as a sub/superclass hierarchy with instances spanning several years was a bad idea. The solution was to remove the `Competency_level` class completely, and instead have `hasCompetency_level` use the datatype `int` as value, ranging from 0 to 9, representing years. This allowed for easier comparison of values when calculating the match score.

## **Degree**

This class refers to the educational status possessed/required in numbers of years. It has subclasses for none, low, mid and high, which has instances like, two-year, bachelor, master and PhD. These sibling instances could be considered as 'almost synonyms'. This class should be redesigned into using datatype integer as well, so it can be incorporated in the matching algorithm.

## **Workplace**

This class expresses the name of the Company the vacancy is representing. Ideally it should be possible to add workplaces to the CV as well, and this can be expanded using a CurrentWorkplace / PreviousWorkplace classes or similar. For now, it's only used to hold the company title for the job ad.

## **Location**

This class contains all counties in Norway, and is part of the rule that every workplace is located somewhere. I've also made relations which states what counties are neighbours with the isNeighbour property; this way one can also adjust the score of matching CVs and job ads with regards to how far away they are from each other. This could also give the possibility of limiting candidates on distance.

## **Properties**

In this ontology I use the same properties to connect competencies and degrees to both CVs and job ads. I could have distinct properties (requiresCompetency and requiresDegree), but as they express the same constrains, they are being reused for both purposes.

### **hasCompetency**

This property has CVbase and JobAds as its domain, and Competency as its range. It connects these in the rule that says that every CV/job ad has at least one competency/competency requirement.

### **hasCompetency\_level**

This property is part of the rules that states that every Competency requires a competency level, and only one level per competency. As discussed above, this was later changed to have the datatype integer property instead of object, thus allowing to compare integers for easier calculating the score when doing the CV – Job Ad matching.

### **hasDegree**

The property that relates the applicants educational degree, or the vacancy's requirement for one.

### **hasWorkplace**

As discussed in the Workplace class paragraph, this property represents the vacancy's title. Having both a class and property just to represent the workplace name might seem redundant, but as proposed

above, there could be more connections added later.

### isNeighbour

This property indicates which counties are next to each other, and is a factor when matching CVs and job ads.

### hasLocation

Points to the geographical location of the company that ordered the vacancy, and where the owner of the CV lives (their address).

### belongsTo

This is a proposed property that could link a CV to a FOAF entry. It's not yet implemented into the system.

## UML diagram

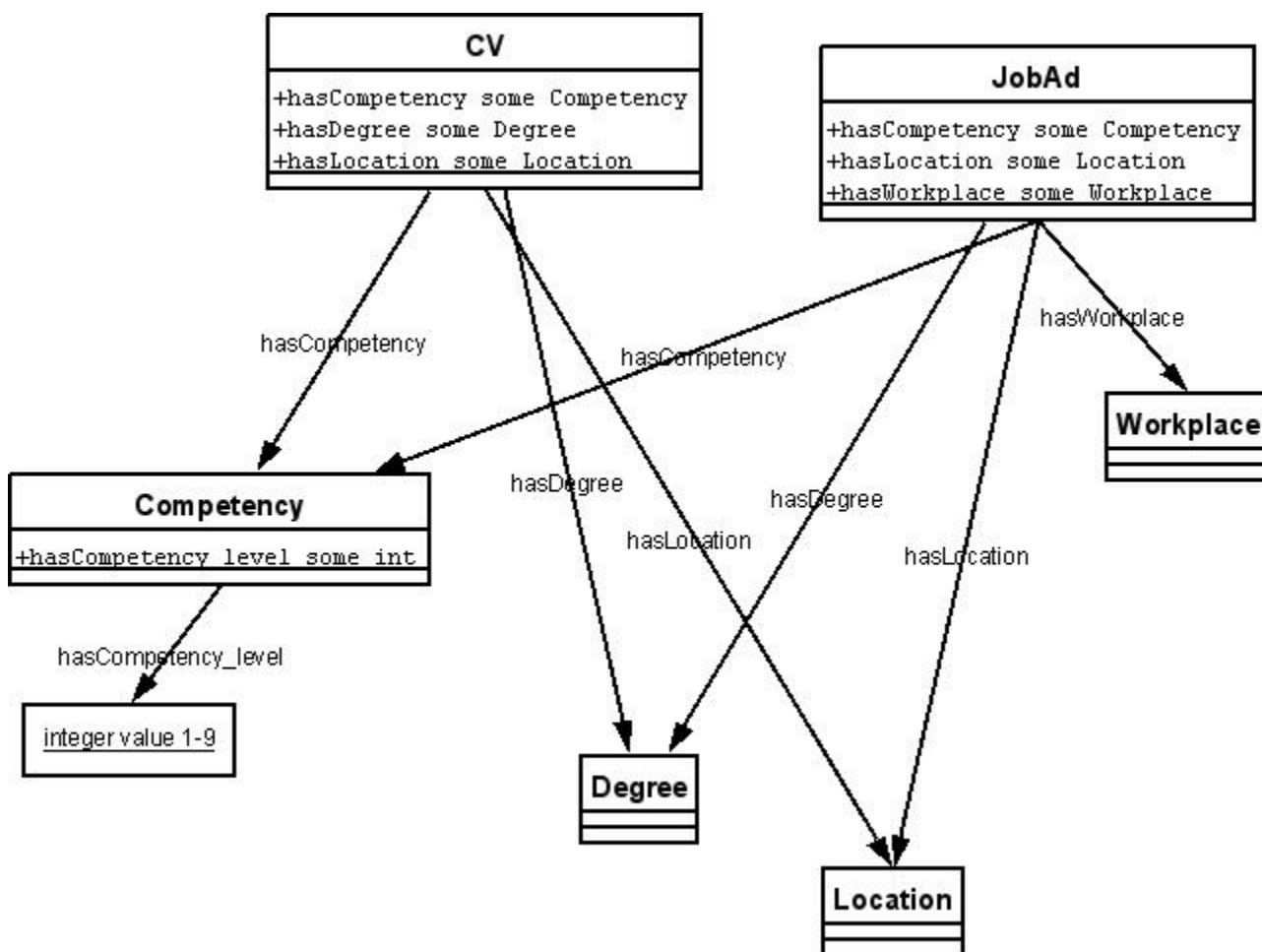


Illustration 22: UML diagram of ontology structure

## Rules

The rules in my project are limited to the scope of OWL. This means that semantically speaking, they are still within the scope of formal semantics, not expanding into powerful soft semantics. I tried incorporating SWRL rules into my project, but there was almost no support for processing it, and very few reasoners that supported it. On top of that, SWRL is in fact undecidable, which does not guarantee completion during the reasoning process. Rules could improve the matching capabilities by introducing criteria for what conditions needed to be met for a CV and a job at to be considered 'similar', expanding on just the closeness in the hierarchy.

## StepStone categories

The job categories used in StepStone are hard-coded as a multi-dimensional array in Perl, so first step was to create a script to transform this into a valid XML document. When this process was completed, the categories were imported into the ontology under the JobAd class via Protégé.

## Protégé

Protégé is a free, open source ontology editor and a knowledge acquisition system. Like Eclipse, Protégé is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the rather complex user interface. Protégé recently has over 80,000 registered users.

Protégé is being developed at Stanford University in collaboration with the University of Manchester. (wikipedia)

## Ontology building in Protégé

Protégé makes it pretty straight forward to build the ontology, once the design has been decided upon. There are tools for batch-creation of classes and subclasses, with options to make every sibling disjoint etc. The whole of the ontology has been built using Protégé exclusively, including the insertion of all classes and instances with respective data.

## OWL version control

OWL has support for version controlling of ontologies, with the owl:versionInfo, owl:priorVersion, owl:backwardCompatible and owl:incompatibleWith tags, to track the version history etc. of the ontology.

```
<owl:Ontology rdf:about="">
  <owl:priorVersion rdf:resource="http://www.ontology.com/ontology3"/>
</owl:Ontology>
```

This is a useful tool if large changes occur from one version to the next, where they might not be compatible with each other. I have chosen not to incorporate version control in my ontology, since it is just a fairly small test ontology.

# Ontology diagrams

Through the next section I will discuss my ontology design along with with illustrations of the various classes and categories. The graphs are created with Protégé's Jambalaya tab<sup>34</sup>, and organised manually for clarity. Names with a circle next to them are classes, and names with diamonds are instances.

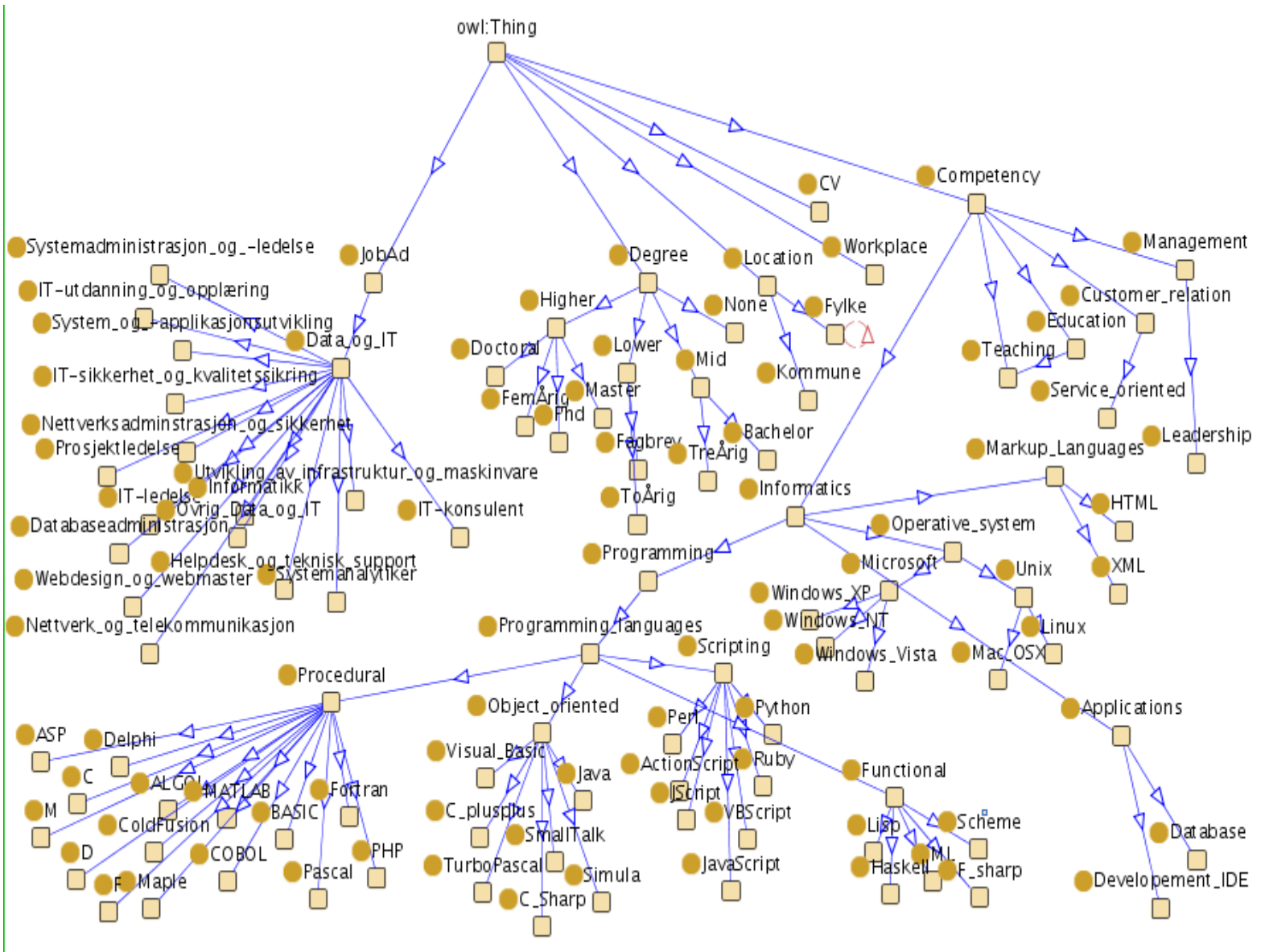


Illustration 23: the complete ontology structure

This figure shows the whole ontology, instances excluded. The top is represented by owl:Thing, the class of all individuals, which has the six subclasses JobAd, CV, Competency, Degree and Location. The CV and Workplace classes doesn't have a subclass hierarchy beneath them, but use the more complex subclasses which will be explained in more detail below.

34 <http://webhome.cs.uvic.ca/~chisel/projects/jambalaya/jambalaya.html>



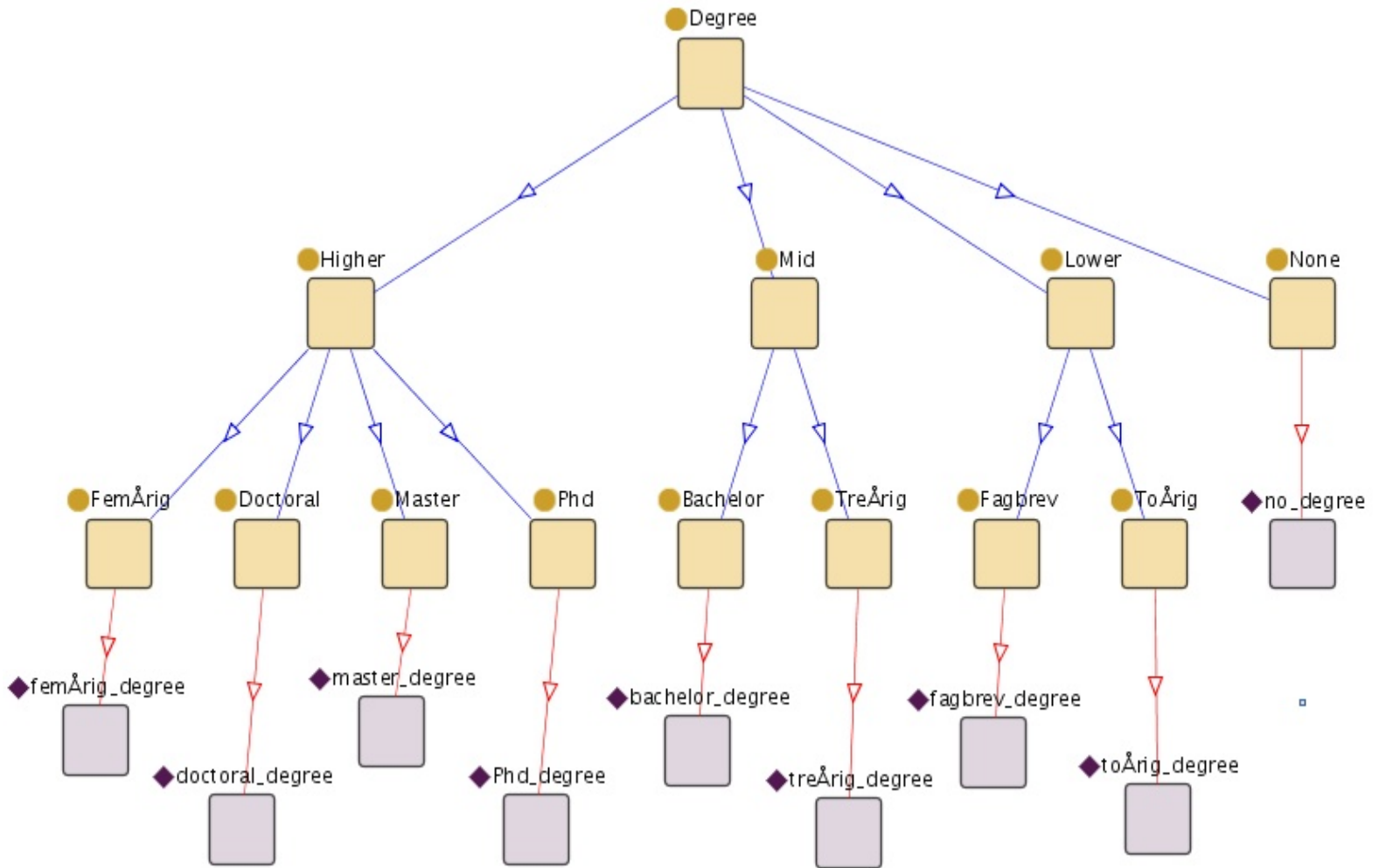


Illustration 24: Degree class with instances

This is a close-up of the Degree class, where subclasses of the three coarse divided Lower, Mid and Higher classes serves as synonyms. This is as stated a very coarse categorisation, and like the competency level class, it could be redone as a integer value. But in the event of doing some information retrieval work, it is useful to have these as categories as degree titles are standardised.

## Information Retrieval

Information retrieval is a huge and interesting field, but unfortunately out of the scope of this thesis. Much research is conducted in this area, for instance by a recent cooperative project called SeSam4 which is addressing this issue, “making it easier to connect information naturally belonging together, but is hard to connect and maintain because of incompatibility in terms and structure. By creating semantic tools, methods and gathering knowledge for information system owners, Sesam4 will ensure that the user easier gets access to relevant information across information owners.”<sup>35</sup>

<sup>35</sup> <http://www.ematch.eu/?q=taxonomy/term/40/0>

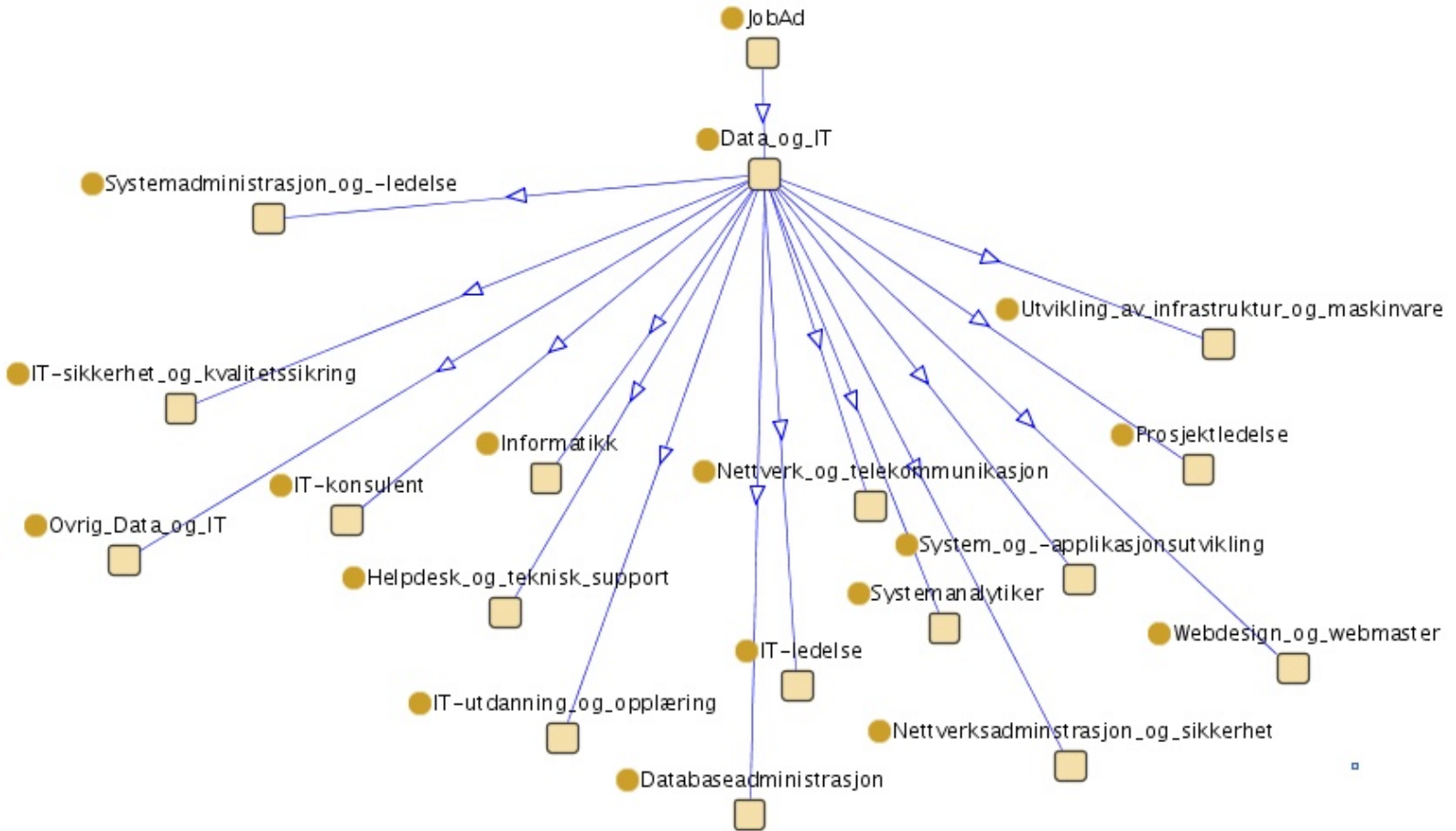


Illustration 25: JobAd class

This is a close-up of the JobAd branch of the ontology, and this is a part which is extracted from the StepStone categories. This branch will typically vary from business to business, as they have different ways of categorise their job ads. The reason for having job ads in a hierarchy could be debated. I have chosen to keep it since it was the first part of the ontology, although it could just as well be a single class, as the CV class. It could be useful to have a rough categorisation however, both to quicker find stored job ads, and in the case of implementing text mining algorithms that auto-categorise job ads. Hence it could be useful to have a categorised CV branch as well, although I have chosen not to do it because of time limitations.

## Ontology population

In order to be able to test the structure and queries, the ontology needs to be populated with instances, and the instances need to be complex enough to take advantage of all the features. Ideally, there should be instances populating every subclass of the competency, CV and Job Ad trees, but I have chosen to focus on a smaller number of instances with more content, than many less-featured ones to get the most realistic entries as possible. The disadvantage of this is that it is harder to determine if a query or matching behaves correctly, as it might have wrongfully included/excluded instances if more classes

had been populated. This is an important aspect when determining whether this method is more accurate than word-to-word matching, but still think the population is large enough to prove this point even if it isn't statistically significant.

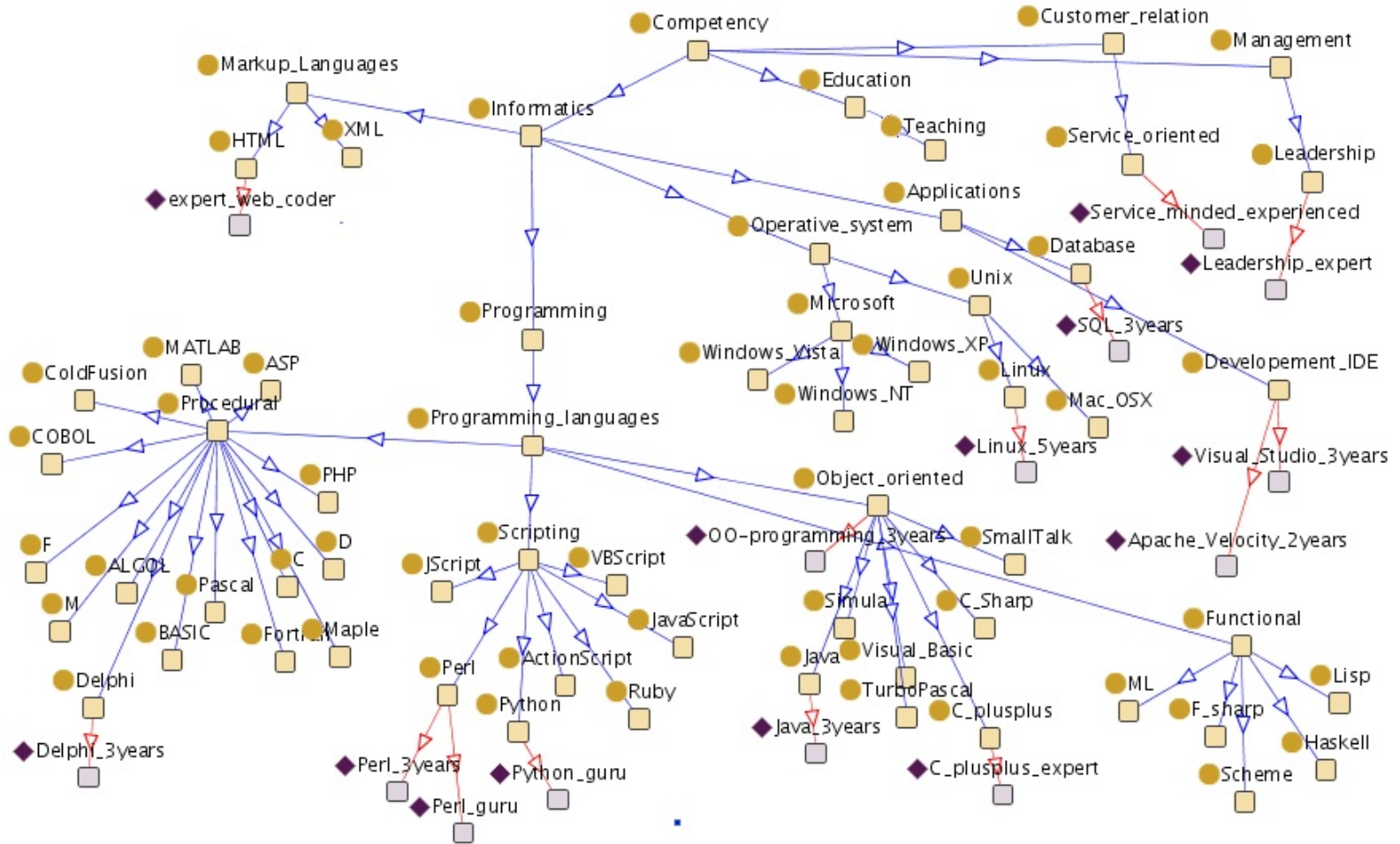
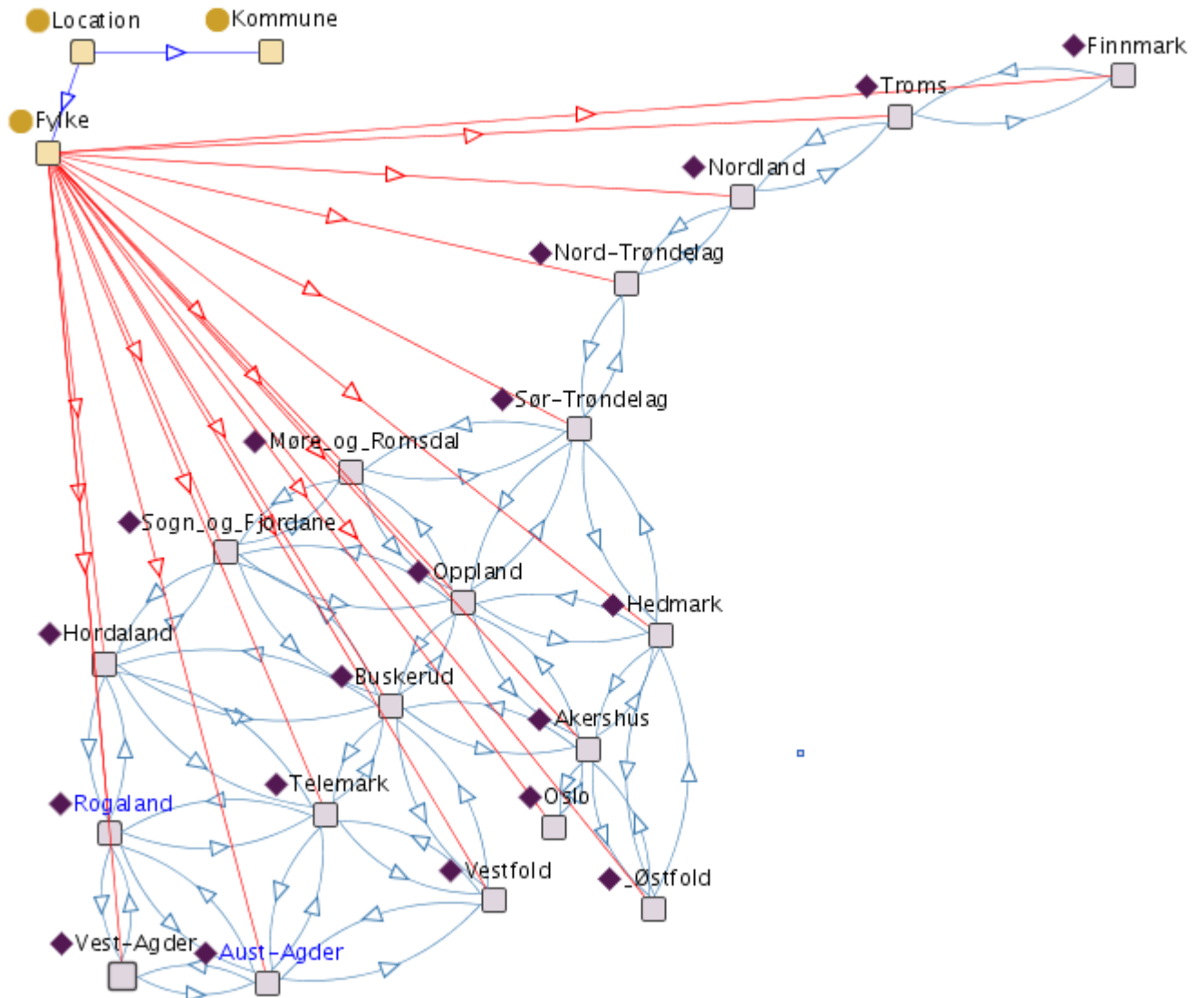


Illustration 26: Competency class

## Ontology Alignment

The tree above shows the competency branch. This class is not taken from any implementation or schema, but rather created from scratch. This works if the ontology is to be used as test ontology or a closed system, but for it to be useful in the Semantic Web vision it needs to be designed for alignment with other ontologies, so standardisation has to be considered. Ontology alignment is an automated process of resolving the 'semantic correspondence' between two ontologies' concepts, so that they can be linked heterogeneously together. This is a very large area, and falls out of the scope of this thesis and will not be discussed further.



*Illustration 27: Location class with instances*

This is the Location branch, where the blue graphs indicates the 'isNeighbour' relationship. When trying to arrange the view in Jambalaya organising it so that the nodes were equally distanced, it was natural that the complete graph would resemble the shape of Norway.

## Development frameworks

In this sections I will focus on the largest development frameworks written for the programming language Java. Although there are tools and APIs for other programming languages as well, it is beyond the scope of this thesis to discuss the benefits and drawbacks of these tools.

### Eclipse

Eclipse is an Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.<sup>36</sup>

### Eclipse + IODT

When I started working with the Java development, I originally started out using IODT, IBM Integrated Ontology development Toolkit<sup>37</sup>, which is a free closed-source toolkit for storage, manipulation, query, and inference of ontologies and corresponding instances, developed in Java. This has a nice EFM workbench (Eclipse Modelling Framework) called EODM (EMF Ontology Definition Metamodel) that allows serialising and manipulation of ontologies using Java objects within Eclipse. It also features Minerva, a high performance ontology repository built on relational databases.

### EODM

EODM Workbench is an Eclipse-based editor for users to create, view and generate OWL ontologies. It has UML-like graphic notions to represent OWL class, restriction and property etc. in a visual way. EODM workbench is built by using EODM, Eclipse Modelling Framework (EMF), Graphic Editing Framework (GEF), which provides the foundation for the graphic view of OWL. It supports drag and drop operations and drawing functions. It also provides two hierarchical views for both OWL class/restriction and OWL object/datatype property. In EODM workbench, one ontology can have multiple views to highlight its different perspectives. Operations in different views can be synchronised automatically (IBM2006).


### Minerva

Minerva is a high performance ontology repository built on relational databases, and it follows the OWL and SPARQL standards. Minerva uses Description Logic reasoners for TBox inference and a set of logic rules translated from Description Logic Programming for ABox inference. This means that inference completeness and soundness on DLP can be guaranteed. You can choose to use IODT's own reasoner, or use Pellet or Racer. Different from other systems, Minerva designs the schema of the back-end database based on both the translated logic rules and OWL constructs to support efficient ontology inference. Experiments on the extended Lehigh University Benchmark show that Minerva has high scalability and efficiency compared to “other well-known systems” (IBM2006).

---

<sup>36</sup> <http://www.eclipse.org/>

<sup>37</sup> <http://www.alphaworks.ibm.com/tech/semanticstk>

<b>IBM Integrated Ontology Development Toolkit</b>					
<b>EODM Extender</b> 			<b>Minerva Extender</b>		
Ontology Mapping Engine		Semantic Web Services Extension	<b>Minerva Workbench</b>		<b>Minerva Database Connectors</b>
<b>EODM Workbench</b>			<b>Minerva Core</b>		
<b>EODM Basic Workbench</b>	<b>EODM Visual Workbench</b>	<b>EODM RSA Workbench</b>	<b>OWL Ontology Storage Model</b>	<b>OWL ABox Inference Engine</b>	<b>SPARQL Query Engine</b>
<b>EODM Core</b>					
<b>RDF Core</b>			<b>OWL Core</b>		
<b>RDF Parser</b>	<b>RDF Reasoner</b>	<b>RDF Transformer</b>	<b>OWL Parser</b>	<b>OWL Reasoner</b>	<b>OWL Transformer</b>
<b>EMF-based RDF/OWL Models</b>					

*Illustration 28: Integrated Ontology Development Toolkit (© IBM)*

## **IODT from the Linux terminal**

After compiling and including the IODT libraries from within Eclipse for a few days, I felt I spent too much time on configuring Eclipse than on actual ontology development. So I started looking at how I could use IODT without Eclipse instead, and it proved fairly easy to upload ontologies to Apache Derby using Minerva, and query this with SPARQL, using the example .Java files. It was a small task configuring the classpath to include the IODT libraries, so I chose to not use Eclipse from then on, and concentrated on working with Emacs and the terminal.

## Jena

As discussed above, I initially used IODT, Pellet and Minerva with Apache-Derby to develop my ontology. Eventually I ran into problems and had questions that needed answers. Unfortunately, there wasn't much more than the supplied documentation to go by, as the online community is practically non-existent. Also, after changing to a new supervisor (David), switching to Jena was a natural thing, as they were using it in their projects. The transition went pretty smooth, since many of the concepts are similar, and the Jena documentation and community<sup>38</sup> is very helpful.

Jena is an open source toolkit for processing and working with Semantic Web data written in Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL and updated through SPARUL. It provides support for OWL, has internal reasoners, and also provides support for external reasoners through the DIG interface. In addition the Pellet reasoner (an open source Java OWL-DL reasoner) can be set up to work in Jena without using the DIG interface.

Getting Jena to work from the terminal was also pretty straight forward. After re-setting the classpath, following the numerous example files provides most of the basic functionality. I mainly use Jena for its querying, but developing ontologies via the API is also possible. After creating an ontology in Protégé, Jena provides several possibilities to access it.

### Jena saving owl file in persistent database

Jena also provides the possibilities to map the ontology to a persistent relational database, and has support for all major database software including MySQL, HSQLDB, Apache Derby, PostgreSQL, Oracle and Microsoft SQL Server. I chose to use MySQL in my testing, and using the examples and online resources, I quickly mapped my ontology to the database using the JDBC driver, connecting to the database with IDBConnection, and creating a model with testClassOWLFileModel using the ProtegeOwl.createJenaOWLModelFromInputStream() method, and then the testClassDBModel.add() function to add the ontology to the database. Querying the database is done in the same fashion using ARQ.

### Jena with JSP

After getting Jena to work with Java in terminal mode, the next step was to create a web interface representing the results in a clear manner to the user. Having worked a bit with JSP in the past, it seemed natural to use. Setting up Jena and JSP with Tomcat 5.5 took a few days of debugging, since Tomcat kept refusing to give read access to the owl files. The solution was to turn Tomcat Security off, a somewhat unsatisfactory solution.

### Jena querying an owl file with ARQ

ARQ is Jena's query engine which supports multiple query languages, including SPARQL, the one I use. It also supports multiple query engines, with abilities to do remote access and SQL transformation. The easiest way of querying an ontology, is to import it into memory directly, and use the Jena

---

<sup>38</sup> irc.freenode.net/6667 #swig



ModelFactory.createOntologyModel method to create an OntologyDocumentManager, and simply use the addAltEntry method to import the owl file. When this is done, you can create query strings, which you send to the QueryFactory for execution (see appendix for source code).

## Jena's Reasoning capabilities

Given an ontology and a model, Jena's inference engine can derive additional statements that the model doesn't express explicitly. Jena provides several reasoner types to work with different types of ontologies. Included in the Jena distribution are a number of predefined reasoners:

- **Transitive reasoner** provides support for storing and traversing class and property lattices. This implements just the *transitive* and *symmetric* properties of rdfs:subPropertyOf and rdfs:subClassOf.
- **RDFS rule reasoner** implements a configurable subset of the RDFS entailments.
- **OWL, OWL Mini, OWL Micro Reasoners** is a set of useful but incomplete implementation of the OWL/Lite subset of the OWL/Full language.
- **DAML micro reasoner** is used internally to enable the legacy DAML API to provide minimal (RDFS scale) inferencing.
- **Generic rule reasoner** is a rule based reasoner that supports user defined rules. Forward chaining, tabled backward chaining and hybrid execution strategies are supported.

Specific to my project, I have used the default reasoner, but I needed to disable the transitive inference mechanism. The reason for this will be explained in the next section.

## Sesame

Sesame is another open source Java framework for storage and querying of RDF data. The framework is fully extensible and configurable with respect to storage mechanisms, inference engines, RDF file formats, query result formats and query languages. Sesame offers a JDBC-like user API, streamlined system APIs and a RESTful HTTP interface supporting the SPARQL Protocol for RDF. Sesame supports SPARQL and SeRQL querying, a memory-based and a disk-based RDF store and RDF Schema inference engines. It also supports most popular RDF file formats and query result formats. Various extensions are available or are being worked at elsewhere. Originally, Sesame was developed by Aduna, a privately owned company, as a research prototype for the EU research project On-To-Knowledge. When this worked ended in 2001, Aduna continued the development in cooperation with NLnet Foundation, developers from Ontotext, and a number of volunteer developers who contribute ideas, bug reports and fixes. It is currently developed as a community project, with Aduna as the project leader.<sup>39</sup>

I discovered Sesame fairly late in my development stage, so it was never considered an option to migrate to this framework, even if it supposedly is faster and more flexible than Jena.

---

<sup>39</sup> <http://www.openrdf.org/doc/sesame2/2.1.3/users/userguide.html>



## Web interface

After setting up the Jena API with Tomcat and creating some test-cases, I started doing the web interface that would be the main part of my project. I created a header and style-sheet, and started implementing the JSP.

### Version 1 – SPARQL endpoint

The first version only provides a basic interface with a textarea input field for direct querying of the owl file. The results were simply returned using the `ResultSetFormatter.out` method appended to a `ByteArrayOutputStream` wrapped with `<xmp>`-tags to prevent URI's being hidden. Creating an endpoint like this is useful when testing, as updates and different queries can quickly be entered.



Illustration 29: SPARQL endpoint

## Version 2 – User friendliness and competency relations

The next step was making the query input more user friendly, and to provide the option of choosing between querying CVs and Job ads, and an input fields for keywords. Also, the results would be returned as links, and when clicking those links, all competencies/requirements for that entry is given. In order to do that, the basic returning of the query result was replaced by loops going through the result set using QuerySolution and RDFNode methods. This loop construct enables better use of style sheets and HTML, and the results can be displayed with links to further actions like looking at the particular result's competencies/competency requirements, descriptions and similar entries.

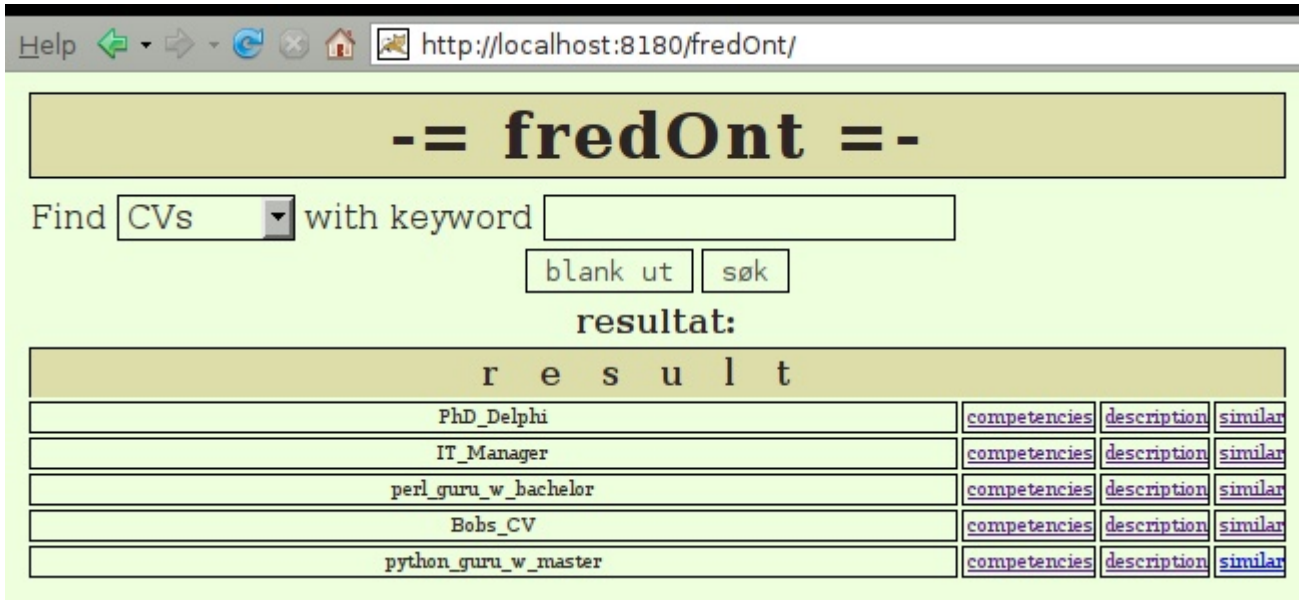


Illustration 30: updated search interface

clicking the 'competencies' button gives the competencies for the result:

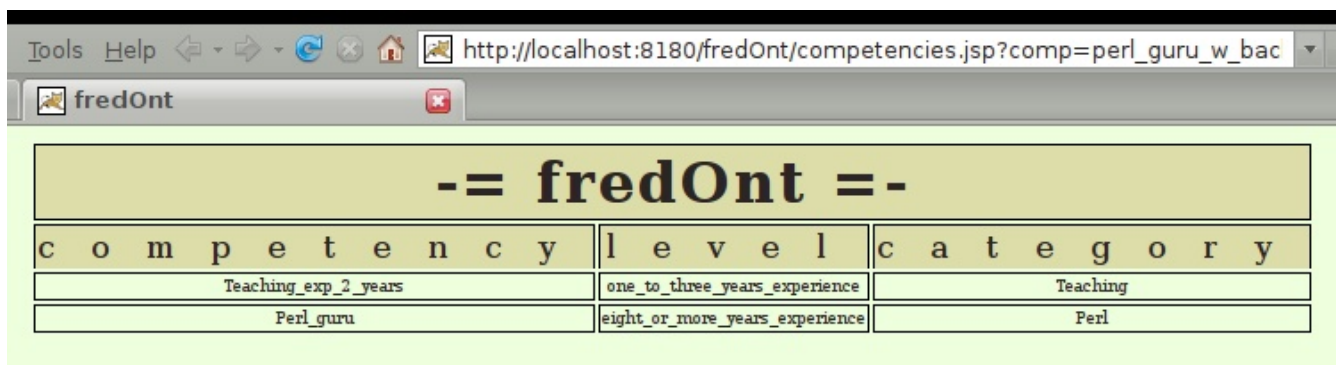


Illustration 31: competency details

clicking the description button gives the CV/job ad description contained in the `rdfs:comment` tag:



Illustration 32: *rdfs:comment*

and finally, the 'similar' button shows instances that have the same parent class as the selected one (sibling instances):



Illustration 33: *sibling instances*

This last feature was reworked into the matching feature introduced in version 3.

## SPARQL queries

SPARQL provides all the results fetched from the ontology, and the following is the list of queries which reflects some of these results:

prefixes:

prefix data: <http://folk.uio.no/fredrhal/ontology/Data\_IT.owl#>

prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

finding all CVs that contains competency in Perl:

```
SELECT ?cv ?p
WHERE { ?cv :hasCompetency ?p .
        ?p rdf:type :Perl . }
```

finding all CVs containing perl, using the keyword filtering

```
SELECT ?a
WHERE { ?a rdf:type data:CV .
        filter regex(str(?a), "perl", "i") }
```

find all CVs that contains the keyword 'perl' (as in Version 2):

```
SELECT distinct ?result where {
  { ?result rdf:type data:CV .
    filter regex(str(?result), "perl", "i")
  }
  union
  { ?result rdf:type data:CV .
    ?result data:hasCompetency ?competency .
    filter regex(str(?competency), "perl", 'i')
  }
}
```

find all sibling classes of 'Perl\_guru's class:

```
SELECT ?c WHERE { :Perl_guru rdf:type ?a .
?c rdfs:subClassOf ?b .
?c rdfs:subClassOf ?b }
```

find all instances of sibling classes of perl\_guru\_w\_bachelor's class, here we use 'distinct' to not get the same instance for each competency

```
SELECT distinct ?e where { :perl_guru_w_bachelor :hasCompetency ?a .
?a rdf:type ?b .
?b rdfs:subClassOf ?c .
?d rdfs:subClassOf ?c .
?e :hasCompetency ?f .
?f rdf:type ?d }
```

This last query proved to be problematic however. When executing the query from within Protégé, the correct result was obtained, and only the instances who had 'Scripting' as superclass was returned. However, when executing the same query from Jena, all instances was returned in the resultset. After some investigating, I found out that this was caused by differences in how the reasoners in Protégé and Jena handled transitivity. Transitivity states that if A is a subclass of B, and B is a subclass of C, then A is also a subclass of C. And as Jena's reasoner was transitive, it returned all sub/super class relationships all way up to the superclass owl:Thing. This posed a problem in my queries, as I only wanted the direct sub/super class relationship. As an attempt to solve this, I found that there was a prefix you could add to the query to just get the direct subclass, thus limiting the results:

```
PREFIX direct : <urn:x-hp-direct-predicate:http://www.w3.org/2000/01/rdf-schema#subClassOf>
```

while this works when you are querying 'downwards' from a specific class, it still fails when querying 'upwards' from a specific instance, as rdf:type is also transitive. So instead I need to configure the Jena reasoner to operate non-transitive. After consulting the Jena API, I found out that this is done by adding the option OWL\_MEM to the createOntologyModel method:

```
OntModel m = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, null);
```

which enables “A specification for OWL models that are stored in memory and do no additional entailment reasoning”<sup>40</sup>.

---

40 <http://jena.sourceforge.net/javadoc/com/hp/hpl/jena/ontology/class-use/OntModelSpec.html>

### Version 3 - Semantic Job-CV matching

If we look back at the figure introduced in the first chapter regarding matching and a job ad, I've implemented a light version of this with regards to the points mentioned in the use case. This required a bit of planning, and I decided to rewrite the competency query, so I would get the needed info to do the score calculations. Keeping track of the various results and comparing them to the new results that would match required some sort of storing the information, so I decided to use the Session object. This was straight forward just using the session.setAttribute and session.getAttribute methods. A problem with just storing the scores in session and just retrieving and adding to them if the same instance came up in different competency categories, was that if you would refresh the page or start a new query that contains these instances, the score would be kept from the previous search and just pile up. In order to start with a 'clean score' every time, the user needs to flush the session, something that is done using the 'flush' link in the header. This isn't very elegant, and ideally the session would flush after each query, or the score calculation should be better implemented.

Then I started on the algorithm that would calculate the match score for each matching CV. As stated earlier, I used the MoC-diagram as an inspiration, and wanted to compute and rate the results with a score, and as discussed in the use cases, I give a high score for sibling instances, a smaller score for cousin instances, with plus or minus points if the required competency level is satisfied (see source code):

Match criterias			
C_plusplus_expert	C_plusplus	8	Object_oriented
Visual_Studio_3years	Developement_IDE	3	Applications
SQL_3years	Database	3	Applications
Linux_5years	Linux	5	Unix

Matching CVs	
CV	Match score
<a href="#">Cplusplus bachelor</a>	22
<a href="#">Bobs CV</a>	10
<a href="#">python guru w master</a>	22

Illustration 34: matching has been executed for a job ad

I also improved the keyword search on the front page to include competency superclasses in the search. The first version only searched with keywords in the title and competency, and while this gives more

accurate results than free text search, it could be too restrictive. One of the main points in this thesis is the improved accuracy that searching in a controlled vocabulary gives, and the added benefits of searching in the semantic tree structure. By including the class and superclass in the keyword search, searching for a general term like 'object oriented programming' should get any CV with Java , C# or C++ competencies returned. This is now implemented, using the UNION construction in SPARQL to allow searching through the various branches of the competency tree. I also included location in the search, so that you can supply location as a keyword.

Here is the updated SPARQL query:

```
SELECT distinct ?result WHERE {
  { ? result rdf:type data:CV .
    filter regex(str(?result), 'perl', 'i')
  }
  union
  // competencies as keyword
  { ?result rdf:type data:CV .
    ?result data:hasCompetency ?competency .
    filter regex(str(?competency), 'perl', 'i')
  }
  union
  // class as keyword
  { ?result rdf:type data:CV .
    ?result data:hasCompetency ?competency .
    ?competency rdf:type ?class .
    filter regex(str(?class), 'perl', 'i')
  }
  union
  // superclass as keyword
  { ?result rdf:type data:CV .
    ?result data:hasCompetency ?competency .
    ?competency rdf:type ?class .
    ?class rdfs:subClassOf ?superclass .
    filter regex(str(?superclass), 'perl', 'i')
  }
  union
  // location as keyword
  { ?result rdf:type data:CV .
    ?result data:hasLocation ?location
    filter regex(str(?location),'oslo', 'i')
  }
}
```

Another limitation I had ignored earlier was that only one keyword could be inserted at each query, something which doesn't make sense. If you want to, for instance, search for CV candidates in Oslo and Akershus, you should not have to conduct two queries. To facilitate this I used the StringTokenizer construct, splitting and looping over each segment separated by a comma and a space (“, “).

## 6. Evaluation & future work

### Results

Although explaining the theory of searching and matching in a controlled-vocabulary ontology structure gives a good argument that this is a better way than traditional word-for-word search, it would be nice to have some results to show for as well. In this section I will provide some results when searching and matching with the system developed. As mentioned previously, the system should ideally contain a large number of differently configured CV and Job Ad instances, as this would reveal potential flaws. Unfortunately because of time limits, there are only instances in some of the categories, and the results are further crippled as they are almost all from the same narrow domain of computer programming competencies. Nevertheless, here is some result data when using the system:

Searching:

Title	competency	category	level	supercat	location
Searching for Job Ads with keyword 'database':					
Databaseadministrasjon_1	Oracle_Designer_3years	Database	3	Applications	Rogaland
	SQL_3years	Database	3	Applications	Rogaland
	Security_3years	Security	3	Informatics	Rogaland
Searching for Job Ads with keyword 'management':					
IT-Architect	Leadership_expert	Leadership	7	Management	Oslo
	Linux_5years	Unix	5	Operative_system	Oslo
	Security_3years	Security	3	Informatics	Oslo
	Perl_5years	Perl	5	Scripting	Oslo
	Network_3years	Network	3	Informatics	Oslo
	Shell_3years	Shell	3	Scripting	Oslo
Searching forCVs with keyword 'scripting':					
python_guru_w_master	Linux_5years	Unix	5	Operative_system	Akershus
	Python_guru	Python	8	Scripting	Akershus
Web_Coder	Macromedia_Flash_4years	Computer_Graphics	4	Applications	Aust-Agder
	ActionScript_3years	ActionScript	3	Scripting	Aust-Agder
	JavaScript_3years	JavaScript	3	Scripting	Aust-Agder
	Mac_OSX_3years	Unix	3	Operative_system	Aust-Agder



searching on location:

Title	location
Searching for Job Ads with keyword 'Oslo':	
java_programmer	Oslo
IT-Architect	Oslo
Game_developer	Oslo
Helpdesk_og_teknisk_support_1	Oslo
Searching for Job Ads with keywords 'Rogaland':	
Databaseadministrasjon_1	Rogaland
Searching for Job Ads with keywords 'Akershus, Buskerud':	
Nettverksadministrasjon_og_sikkerhet_1	Akershus
Perl_koder	Buskerud

matching:

competency	category	level	supercat	location
Perl_3years	Perl	3	Scripting	Buskerud
Matching CVs				
CV			Match score	
<b>Web_Coder</b>			10	
Macromedia_Flash_4 years	Computer_Graphics	4	Applications	Aust-Agder
ActionScript_3years	ActionScript	3	Scripting	Aust-Agder
JavaScript_3years	JavaScript	3	Scripting	Aust-Agder
Mac_OSX_3years	Unix	3	Operative_system	Aust-Agder
<b>python_guru_w_master</b>			5	
Linux_5years	Unix	5	Operative_system	Akershus
Python_guru	Python	8	Scripting	Akershus

Match criterias for Webdesigner				
competency	category	level	supercat	location
CSS_5years	CSS	5	Markup_Languages	Vest-Agder
expert_web_coder	HTML	8	Markup_Languages	Vest-Agder
Photoshop_guru	Computer_Graphics	8	Applications	Vest-Agder
Matching CVs				
CV			Match score	
<b>Skilled_Webdesigner</b>			71	
CSS_5years	CSS	5	Markup_Languages	Oslo
expert_web_coder	HTML	8	Markup_Languages	Oslo
Visual_Studio_3years	Development_IDE	3	Applications	Oslo
Photoshop_guru	Computer_Graphics	8	Applications	Oslo
SQL_3years	Database	3	Applications	Oslo
<b>Web_Coder</b>			13	
Macromedia_Flash_4years	Computer_Graphics	4	Applications	Aust-Agder
ActionScript_3years	ActionScript	3	Scripting	Aust-Agder
JavaScript_3years	JavaScript	3	Scripting	Aust-Agder
Mac_OSX_3years	Unix	3	Operative_system	Aust-Agder
<b>Microsoft_guru</b>			5	
Visual_Basic_3years	Visual_Basic	3	Object_oriented	Akershus
Windows_XP_5years	Microsof	5	Operative_system	Akershus
Windows_Vista_2years	Microsof	2	Operative_system	Akershus
Windows_2000_3years	Microsof	3	Operative_system	Akershus
Windows_NT_3years	Microsof	3	Operative_system	Akershus
MS_Office_3years	Productivity	3	Applications	Akershus
<b>Database_guy</b>			15	
Oracle_Designer_3years	Database	3	Applications	Oppland
SQL_3years	Database	3	Applications	Oppland
MySQL_expert	Database	7	Applications	Oppland
Network_3years	Network	3	Informatics	Oppland

As seen from the results, there are areas that could need some improvement. Renaming and reorganising certain parts of the ontology would improve results. The class 'Applications' for instance, contains too many diverse subcategories, as seen when the 'Webdesigner' vacancy with 'Photoshop – Application' get matched with Database\_guy which picks up points for Application instances not related to Photoshop. Despite this, the results show that 'Skilled\_Webdesigner' has the largest score by far, meeting all the requirements for the 'Webdesigner' vacancy.

## Evaluation

When conducting research in a specific area, you do tests and enquiries to get qualitative or quantitative data to prove or falsify your theories. When doing a system like I'm doing as the main part of proving/falsifying claims comes from the system implemented. This makes it a bit special, and very dependant on my skills as a designer and developer whether my claims can be verified or not. Still, there are some factors that are measurable, and I will discuss them in the following section, through a series of questions with answers.

### **how accurate are the results?**

The important point here is that the keyword searching is done through the competency names, which is tagged data. This means that you don't get most of the random word matches occurring at a regular search. When you enter a keyword, it is never matched with the vacancy or CV content, which is stored in the `rdf:comment`. You only search for specifically chosen competencies, selected to represent the entry. This is the huge benefit. You should only get search results that is tagged with the keywords.

### **how much more time does it take to retrieve the query results?**

The search time is computationally measurable, and in my implementation the search time is displayed under the search results. On average, with my small ontology, the search time usually stays between 100 and 400ms, except for the first search which usually takes 2-3 seconds. This is because the ontology (in my design) is being read into memory, which causes the initial long loading time. This only works for a small test ontology like mine, because a large ontology would require the server to have an extremely large amount of memory. But as stated above, Jena provides tools to make the ontology persistent in a regular database management system like MySQL storing on regular disks, with the ontology mapping providing a 'semantic layer' on top. It is also possible to store RDF in so-called Triplestores, a special kind of RDF database. These have proven to cope with large masses of data<sup>41</sup>, and with tools like FeDeRate it is possible to achieve the speed, compactness, and integrity constraints of conventional relational databases.<sup>42</sup>

### **trade-off – is the extra effort worth it?**

Creating a vacancy where all competency requirements are categorised takes time. And this comes in addition to the vacancy text itself, which serves as the main text processed by the human reader. In my small system, every competency requires a competency level, a workplace requires location etc., which needs to be filled out for every vacancy and CV. But in order for these things to exist, the ontology must already be built, and this is the largest task. My small test ontology caters only for a small branch of Information Technology competencies, and took quite a while to design. But once this is in place, it's a fairly small job to maintain and update the ontology. There might also be text mining algorithms available that classifies entries based on content, but I have chosen not to look into that. What you do get however, is a semantically tagged system with much higher accuracy when searching for candidates/vacancies, and you get the possibility of ranking CVs and vacancies on the basis of how close their competencies lies in the ontology tree. I believe this kind of semantic advantages will become more and

---

41 <http://esw.w3.org/topic/LargeTripleStores>

42 <http://www.w3.org/2004/04/30-RDF-RDB-access/>

more common in the time that comes, not only within the HR domain, but every area that tries to classify objects and properties of various kinds.

## **Future work**

The next section will propose some logical next steps for the system, and also discuss how it can be beneficial to other domains.

### **Matching on Location**

Another factor when matching CVs and job ads is location. It makes sense to give extra points to matches that lives in the same area as the company is located, so I intended to implement this with the 'Fylke' part of my ontology, governed by the 'hasLocation' and 'isNeighbour' properties. Here, being in the same county (Fylke) should give X points, and being in a neighbour county should give less point. As described above, the location class is used for workplace location, but time limitations forced me to omit it from the matching algorithm.

### **Giving access to the user**

For the system to be of any real value, it must allow users to fully interact with it. This means that in addition to searching and matching, the system must also be able to store CVs and job ads, as well as let administrators manage and add not only instances, but also new category classes. This is achieved using SPARQL insert and update queries, with drop downs in the web interface discussed in the next sections.

### **Users adding a CV**

This action is pretty straight forward, where job seekers can create a competency profile, using several levels of drop downs to specify their competencies and competency levels etc. This could be implemented as a dynamic solution, more specific drop down menus appearing when a more general category is selected, down to the desired specialisation. There should be restrictions on what operations a job seeker should be allowed to do, e.g. not create own category classes.

### **Customers adding a Job Ad**

This action is similar to the CV creation action, with the same competency drop downs for vacancy requirements. Neither the customer should get access to class creation in the ontology, or at least not without approval from a 'competency professional' which decides on the ontology design. This is to prevent the user to add erroneous categories, so called 'metacrap', a term coined by Cory Doctorow in his paper on metadata and 'the seven insurmountable obstacles between the world as we know it and meta-utopia'<sup>43</sup>. In short, it is impossible to insure the quality of meta data when many users can create it.

---

43 <http://www.well.com/~doctorow/metacrap.htm>

Other features and ideas that could improve the system includes broadening the ontology to include more competencies and categories, optimise the queries, incorporate rules and statistics to improve the matching, connecting to other ontologies like FOAF, skills management and e-Learning, and to look into text mining algorithms for automatic category placement.

## **Relevance for other domains**

Ontology-based property matching isn't something that is only useful in this particular domain, and it's being applied to many domains already. In fact, it is probably applicable to most areas where items are compared for closeness based on their 'internal' properties or parts. Here are several areas where this technology could be useful:

- shopping online, for instance mobile phones: the user inputs her desired specifications, and a list of mobile phones which properties are gathered from an ontology is retrieved, ranked on best match. The main advantage of doing this semantically is that the user can be less specific on technical details, having superclasses that cover synonyms and more specific features
- music matcher: artists can be structured several ontologies like 'genre', 'instrument', 'label' etc, and if you want to find artists similar to e.g. 'Hardfloor', you would get results based on the genre superclass 'House', with subclasses like 'Acid House', 'Electro House', 'Tech House', 'Disco House', 'Deep House' etc., instrument superclass 'Analog Synthesizer', 'Drum Machine' and 'Sequenced Synthesizer', with subclasses like 'Roland TB-303', 'Roland TR-808', 'Roland TR-909', 'Moog Minimoog', 'Korg MS-20', 'FutureRetro 777' etc., and notable labels releasing Acid House music
- semantic dating services : your personality is mapped in a personality features ontology, and you can find other people matching your interests
- medical expert systems, where a computer determines treatment and suggests medication based on the supplied symptoms

## 7. Summary and conclusion

In this master thesis I have tried to show how Semantic Web technology can improve searching and matching in the Human Resource Management domain.

Chapter 1 gave an introduction on how online recruitment works today, and some of the problems concerning word-based searching. The main problem is that most search engines use a word-for-word matching, and this is not guaranteed to give the correct results.

Chapter 2 presented some use cases on possible scenarios in a semantically powered system, proposing sequence diagrams on how a system could work. Enabling semantics in the form of metadata and using this as a search base, instead of word-for-word matching would improve the search results, as the search is conducted on tagged categories and not on free form text content. It also gave a survey of the state of the art, presenting some of the Semantic Web projects in general, but also some Semantic Web driven projects within HRM.

Chapter 3 gave a brief overview of the technologies that make up the Semantic Web layer cake, and discussed some of the competing technologies and tools and why these were not chosen. As more of the tools become accepted standards, it is more probable that they will be used to a larger extent, making the Semantic Web a practical realisation, and not only a theory.

Chapter 4 discussed the research method used when approaching the field of Semantic Web technology, and how decisions were made throughout the writing of the thesis and system implementation. Arguments for the chosen methods were given, along with discussions on delimitations and planning of the thesis.

Chapter 5 documented how a small ontology was designed, and the implementation of a small web-based test system to show how semantic searching and matching can be achieved.

Chapter 6 showed a set of results when using the test system, along with a discussion of the benefits and disadvantages of using Semantic Web technologies in a system like this. It also proposed logical next steps for my project if it were to be developed further, as well as how the techniques used could be generalised and applied in other domains.

### Conclusion

The benefits of incorporating Semantic Web technologies in online recruitment tools to improve searching and matching is apparent. Searching through competency tags in a hierarchy instead of vacancy or CV content seems to return more valuable results. Not only is the user presented more accurate results, but is also given a 'second best' list which might suffice if an exact match doesn't exist. To achieve this kind of results in practice however, requires a lot of work. What classes and rules should make up a work-related competency ontology needs to be standardised, and designing it would take a lot of effort. But if a HR competency ontology standard was to emerge, I am sure that the technologies available would suffice to create useful commercial HR tools.

What could be learned from this process of implementing a system and writing a thesis? Personally I have gained a lot of knowledge about the Semantic Web, not only through reading articles and books, but more importantly through implementing a small Semantic Web system myself. This process gives both technical insight as well as real life examples of the theories presented in the literature, both in the ontology design phase and the system implementation phase. I hope this thesis can be useful to anyone starting a Semantic Web project, looking into setting up a web based system from scratch, this thesis functioning as a small guide on how to set up a similar system. My other hope is to contribute with some arguments to the Recruitment Agency business about the benefits of using Semantic Web technologies within the HRM domain.

## **The future of the Semantic Web**

The advantages that the Semantic Web brings are obvious. Allowing the recognition of relevant information on a semantic level instead of word matching, performing semantic-based deduction with sound inference processes will make searching and matching more powerful. But in order to develop such advanced semantic solutions, the need for standardisation is still critical.

“The challenge is to reach common standardised semantic systems nationwide and Europe-wide. For this reason, a strong cooperation between national and international institutions as well as the construction of multi-stakeholder partnerships is needed to facilitate and foster the labour market mobility and transparency.” (OOAHR2007)

### **OWL not enough?**

As stated in (Sheth2005), it is likely that more will be expected from the Semantic Web than the current technologies like Description Logics and OWL. These are designed to balance expressiveness and computability, but as explained in the article:

“supporting expressiveness that meet requirements of practical applications and the techniques that support their development is crucial. It is not desirable to limit the Semantic Web to one type of representation where expressiveness has been compromised at the expense of computational property such as decidability.”

Any limitation of First Order Logic is also a limitation of Description Logics, and (Woods2004) states that “it is already clear that first-order logic is insufficient to deal with many semantic problems inherent in understanding natural language as well as the semantic requirements of a reasoning system for an intelligent agent using knowledge to interact with the world.”

Technological advances has been made since the publishing of these articles, and although the layer of powerful soft semantics with its rule languages and probabilistic methods are still immature and not widely supported, it is the natural next step in evolving the Semantic Web.

# Bibliography

## Books

(Antoniou2004)

Antoniou, G. & van Harmelen, F. **2004**, *A Semantic Web Primer*, The MIT Press, ISBN 0-262-01210-3

(Baader2003)

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D. & Patel-Schneider, P. F. **2003**, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, ISBN 0-521-87625-7

(Cheetam2005)

Graham Cheetham & Geoff Chivers **2005**, *Professions, Competence and Informal Learning*, Edward Elgar Publishing, ISBN 1-843-76408-3

(Cimiano2006)

Cimiano, P. **2006**, *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*, Springer, ISBN 0-387-30632-3

(Lacy2005)

Lacy, L. W. **2005**, *Owl: Representing Information Using the Web Ontology Language*, Trafford Publishing, ISBN 1-412-03448-5

(Passin2004)

Passin, T. B. **2004**, *Explorer's Guide to the Semantic Web*, Manning Publications, ISBN 1-932-39420-6

(Skagestein2002)

Skagestein, G. **2002**, *Systemutvikling -fra kjernen og ut, fra skallet og inn*, Høyskoleforlaget, ISBN 82-7634-503-4

## Articles

(ANSI2003)

ANSI/NISO Z39.19 – **2003**, *Guidelines for the Construction, Format, and Management of Monolingual Thesauri*

(Baader2003a)

Baader, F.; Horrocks, I. & Sattler, U. **2003**, *Description Logics as Ontology Languages for the Semantic Web*

(Biesalski2005)

Biesalski, E. & Abecker, A. **2005**, *Human Resource Management with Ontologies*

(Bizer2005)

Bizer, C.; Heese, R.; Mochol, M.; Oldakowski, R.; Tolksdorf, R. & Eckstein, R. **2005**, *The Impact of Semantic Web Technologies on Job Recruitment Processes*



- (Colucci2003)  
Colucci S., Di Noia T., Di Sciascio E., Donini F.M., Mongiello M., Mottola M. **2003**, *A formal approach to ontology-based semantic match of skills descriptions*
- (Dogac2002)  
Dogac, A.; Laleci, G.; Kabak, Y. & Cingil, I. **2002**, *Exploiting Web Service Semantics: Taxonomies vs. Ontologies*
- (Dorn2007)  
Dorn, J.; Naz, T. & Pichlmair, M. **2007**, *Ontology Development for Human Resource Management*
- (Garshol2007)  
Garshol, L. M. **2007**, *Living with topic maps and RDF*
- (Haarslev2004)  
Volker Haarslev; Ralf Möller & Michael Wessel **2004**, *Querying the Semantic Web with Racer + nRQL*
- (Hasan2004)  
Helen Hasan **2004**, *Knowledge creation through systems development, Introduction, Australasian Journal of Information Systems*
- (Horrocks2005)  
Ian Horrocks **2005**, *Description Logics in Ontology Applications*, KI/Tableaux
- (Horridge2004)  
Horridge, M.; Knublauch, H.; Rector, A.; Stevens, R. & Wroe, C. **2004**, *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*
- (Horridge2006)  
Horridge, M.; Tsarkov, D. & Redmond, T. **2006**, *Supporting Early Adoption of OWL1.1 with Protégé-OWL and FaCT++*
- (IBM2006)  
IBM **2006**, *Integrated Ontology development Toolkit documentation*
- (Lytras2006)  
Miltiadis Lytras & Ambjörn Naeve **2006**, *Semantic e-learning: synthesising fantasies*
- (Mochol2007)  
Mochol, M.; Jentzsch, A. & Wache, H. **2007**, *Suitable employees wanted? Find them with semantic techniques*
- (Mochol2004)  
Mochol, M.; Oldakowski, R. & Heese, R. **2004**, *Ontology based Recruitment Process*
- (Mochol2007a)  
Mochol, M.; Wache, H. & Nixon, L. **2007**, *Improving the accuracy of job search with semantic techniques*
- (OOAHR2007)  
Jarrar, M.; Vervenne, L.; Maynard, D.; hameed, A.; Schmidt, A.; Roche, C.; Marinoni, C.; Kunzmann, C.; Damiani, E.; Trichet, F.; Hoppenbrouwers, J.; Lundqvist, K.; Coillie, M. V.; Ronchetti, M.; Brown, M.; Leenheer, P. D.; Ravet, S.; White, S. & Christiaens, S. **2007**, *HR-Semantics Roadmap OOAHR*

(Schmidt2004)

Schmidt, A. **2004**, *Context-Steered Learning : the Learning in Process Approach*

(Schmidt2006)

Schmidt, A. & Kunzmann, C. **2006**, *Towards a Human Resource Development Ontology for Combining Competence Management and Technology-Enhanced Workplace Learning*

(Sheth2005)

Sheth, A.; Ramakrishnan, C. & Thomas, C. **2005**, *Semantics for the Semantic Web: The Implicit, the Formal and the Powerful*

(Woods2004)

William A. Woods **2004**, *Meaning and Links: A Semantic Odyssey*, Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference

### **Web resources**

Total Talent Management, e-Recruitment and Job Sites from StepStone

<http://www.stepstone.com/EN/> (09.01.07)

Semantic Web – Wikipedia

[http://en.wikipedia.org/wiki/Semantic\\_Web](http://en.wikipedia.org/wiki/Semantic_Web) (16.08.07)

Ontology (information science) - Wikipedia

[http://en.wikipedia.org/wiki/Ontology\\_\(computer\\_science\)](http://en.wikipedia.org/wiki/Ontology_(computer_science)) (16.08.07)

Jena (framework) – Wikipedia

[http://en.wikipedia.org/wiki/Jena\\_\(framework\)](http://en.wikipedia.org/wiki/Jena_(framework)) (30.01.08)

Description logic – Wikipedia

[http://en.wikipedia.org/wiki/Description\\_logic](http://en.wikipedia.org/wiki/Description_logic) (16.08.07)

Semantic reasoner – Wikipedia

[http://en.wikipedia.org/wiki/Semantic\\_reasoner](http://en.wikipedia.org/wiki/Semantic_reasoner) (16.08.07)

HR-XML

<http://www.hr-xml.org/> (14.02.07)

RIF

[http://www.w3.org/2005/rules/wiki/RIF\\_Working\\_Group](http://www.w3.org/2005/rules/wiki/RIF_Working_Group) (12.02.08)

SWRL

<http://www.w3.org/Submission/SWRL/> (16.08.07)

VUB STAR.Lab

<http://www.starlab.vub.ac.be/website/> (05.07.08)

Professional Learning Ontology and Competencies

[http://professional-learning.eu/competence\\_ontology](http://professional-learning.eu/competence_ontology) (20.03.08)

URI Declaration for the Moon

<http://dbooth.org/2007/moon/decl.html> (15.03.08)

The Protégé Ontology Editor and Knowledge Acquisition System

<http://protege.stanford.edu/> (08.03.07)

Jena Semantic Web Framework

<http://jena.sourceforge.net/> (30.01.08)

Jena Javadoc

<http://jena.sourceforge.net/javadoc/> (23.03.08)

Metacrap  
<http://www.well.com/~doctorow/metacrap.htm> (20.09.07)

PRocess-Oriented Learning and Information eXchange - PROLIX Consortium  
<http://www.prolixproject.org/> (15.03.08)

Lingway e-Recruitment  
<http://www.lingway.com/> (15.03.08)

Innovantage  
<http://www.innovantage.co.uk/> (15.03.08)

DOGMA  
<http://en.wikipedia.org/wiki/DOGMA> (15.03.08)

Learning in Progress  
<http://lttf.ieee.org/icalt2004/ppt/235.ppt> (15.03.08)

CoDrive  
<http://www.codrive.org/> (15.03.08)

PoCeHRMOM  
<http://cvc.ehb.be/PoCeHRMOM/Home.htm> (15.03.08)

Ontotext's JOCI  
[http://langtech.jrc.it/Documents/051005\\_Popov\\_Slides.ppt](http://langtech.jrc.it/Documents/051005_Popov_Slides.ppt) (15.03.08)

The eCCO System  
<http://www.springerlink.com/content/ym511g2463102x67/> (15.03.08)

WordNet  
<http://wordnet.princeton.edu/> (16.08.07)

DBPedia  
<http://dbpedia.org/> (16.08.07)

Dublin Core  
<http://dublincore.org/> (16.08.07)

FOAF  
<http://www.foaf-project.org/> (16.08.07)

SHOE  
<http://www.cs.umd.edu/projects/plus/SHOE/> (15.03.08)

SESAME  
<http://www.openrdf.org/> (05.07.08)

SkillZoom  
<http://www.skillzoom.com/> (05.07.08)

Applied Minds  
<http://www.appliedminds.co.uk/ontologies/intro.htm> (20.06.08)

# Appendix

## Source code for ontology

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://folk.uio.no/fredrhal/ontology/Data_IT.owl#"
  xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://folk.uio.no/fredrhal/ontology/Data_IT.owl">
  <owl:Ontology rdf:about="" />
  <owl:Class rdf:ID="Helpdesk_og_teknisk_support">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Data_og_IT"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Database">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Applications"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Ovrig_Data_og_IT">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Data_og_IT"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Visual_Basic">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Object_oriented"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Master">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Higher"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Fortran">
    <rdfs:subClassOf>
```

```

    <owl:Class rdf:ID="Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Productivity">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Applications"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Unix">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Operative_system"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FemÅrig">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Higher"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Management">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Competency"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Perl">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Scripting"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Lower">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Degree"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Security">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Informatics"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ML">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Functional"/>
  </rdfs:subClassOf>

```

```

</owl:Class>
<owl:Class rdf:ID="TreÅrig">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Mid"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Pascal">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Data_og_IT">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="JobAd"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class rdf:about="#Informatics"/>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasCompetency"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Microsoft">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Operative_system"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="IT-sikkerhet_og_kvalitetssikring">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="COBOL">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Shell">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Scripting"/>

```

```

</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="C_plusplus">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Object_oriented"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Delphi">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="JScript">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Scripting"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Doctoral">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Higher"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="None">
  <rdfs:subClassOf rdf:resource="#Degree"/>
</owl:Class>
<owl:Class rdf:about="#Applications">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Informatics"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Object_oriented">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Programming_languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="System_og_-_applikasjonsutvikling">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>

```

```

<rdfs:subClassOf>
  <owl:Restriction>
    <owl:someValuesFrom>
      <owl:Class rdf:ID="Programming"/>
    </owl:someValuesFrom>
  <owl:onProperty>
    <owl:ObjectProperty rdf:about="#hasCompetency"/>
  </owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="hasDegree"/>
    </owl:onProperty>
    <owl:someValuesFrom rdf:resource="#Degree"/>
  </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Databaseadministrasjon">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="ColdFusion">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Higher">
  <rdfs:subClassOf rdf:resource="#Degree"/>
</owl:Class>
<owl:Class rdf:about="#Operative_system">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Informatics"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Ruby">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Scripting"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ActionScript">
  <rdfs:subClassOf>

```



```

    <owl:Class rdf:about="#Scripting"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Lisp">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Functional"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="HTML">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Markup_Languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="F">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="MATLAB">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="F_sharp">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Functional"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Programming">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Informatics"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Systemadministrasjon_og_-ledelse">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Location"/>
<owl:Class rdf:ID="CV">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasLocation"/>

```

```

</owl:onProperty>
<owl:someValuesFrom rdf:resource="#Location"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Degree"/>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="#hasDegree"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:about="#hasCompetency"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:about="#Competency"/>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="IT-konsulent">
<rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Bachelor">
<rdfs:subClassOf>
<owl:Class rdf:about="#Mid"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Informatics">
<rdfs:subClassOf>
<owl:Class rdf:about="#Competency"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Developement_IDE">
<rdfs:subClassOf rdf:resource="#Applications"/>
</owl:Class>
<owl:Class rdf:ID="XML">
<rdfs:subClassOf>

```

```

    <owl:Class rdf:about="#Markup_Languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Functional">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Programming_languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Service_oriented">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Customer_relation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Nettverk_og_telekommunikasjon">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="M">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Network">
  <rdfs:subClassOf rdf:resource="#Informatics"/>
</owl:Class>
<owl:Class rdf:about="#Mid">
  <rdfs:subClassOf rdf:resource="#Degree"/>
</owl:Class>
<owl:Class rdf:ID="Systemanalytiker">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Webdesign_og_webmaster">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="ToÅrig">
  <rdfs:subClassOf rdf:resource="#Lower"/>
</owl:Class>
<owl:Class rdf:ID="CSS">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Markup_Languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Fagbrev">

```

```

<rdfs:subClassOf rdf:resource="#Lower"/>
</owl:Class>
<owl:Class rdf:ID="Education">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Competency"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="C_Sharp">
  <rdfs:subClassOf rdf:resource="#Object_oriented"/>
</owl:Class>
<owl:Class rdf:ID="Maple">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Procedural"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Procedural">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Programming_languages"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="IT-utdanning_og_oppl ring">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Leadership">
  <rdfs:subClassOf rdf:resource="#Management"/>
</owl:Class>
<owl:Class rdf:about="#Programming_languages">
  <rdfs:subClassOf rdf:resource="#Programming"/>
</owl:Class>
<owl:Class rdf:ID="Fylke">
  <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>
<owl:Class rdf:ID="Teaching">
  <rdfs:subClassOf rdf:resource="#Education"/>
</owl:Class>
<owl:Class rdf:ID="Kommune">
  <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>
<owl:Class rdf:ID="VBScript">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Scripting"/>
  </rdfs:subClassOf>

```

```

</owl:Class>
<owl:Class rdf:ID="Python">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Scripting"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Customer_relation">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Competency"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="C">
  <rdfs:subClassOf rdf:resource="#Procedural"/>
</owl:Class>
<owl:Class rdf:ID="TurboPascal">
  <rdfs:subClassOf rdf:resource="#Object_oriented"/>
</owl:Class>
<owl:Class rdf:about="#Markup_Languages">
  <rdfs:subClassOf rdf:resource="#Informatics"/>
</owl:Class>
<owl:Class rdf:about="#Scripting">
  <rdfs:subClassOf rdf:resource="#Programming_languages"/>
</owl:Class>
<owl:Class rdf:ID="PHP">
  <rdfs:subClassOf rdf:resource="#Procedural"/>
</owl:Class>
<owl:Class rdf:ID="ASP">
  <rdfs:subClassOf rdf:resource="#Procedural"/>
</owl:Class>
<owl:Class rdf:ID="Simula">
  <rdfs:subClassOf rdf:resource="#Object_oriented"/>
</owl:Class>
<owl:Class rdf:ID="JavaScript">
  <rdfs:subClassOf rdf:resource="#Scripting"/>
</owl:Class>
<owl:Class rdf:ID="Utvikling_av_infrastruktur_og_maskinvare">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Hardware">
  <rdfs:subClassOf rdf:resource="#Informatics"/>
</owl:Class>
<owl:Class rdf:ID="Computer_Graphics">

```

```

<rdfs:subClassOf rdf:resource="#Applications"/>
</owl:Class>
<owl:Class rdf:ID="Nettverksadminstrasjon_og_sikkerhet">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="Haskell">
  <rdfs:subClassOf rdf:resource="#Functional"/>
</owl:Class>
<owl:Class rdf:ID="Java">
  <rdfs:subClassOf rdf:resource="#Object_oriented"/>
</owl:Class>
<owl:Class rdf:about="#JobAd">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class rdf:about="#Competency"/>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasCompetency"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasWorkplace"/>
      </owl:onProperty>
      <owl:someValuesFrom>
        <owl:Class rdf:ID="Workplace"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasLocation"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#Location"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

```

<owl:Class rdf:about="#Competency">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="hasCompetency_level"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="Phd">
  <rdfs:subClassOf rdf:resource="#Higher"/>
</owl:Class>
<owl:Class rdf:ID="BASIC">
  <rdfs:subClassOf rdf:resource="#Procedural"/>
</owl:Class>
<owl:Class rdf:ID="Scheme">
  <rdfs:subClassOf rdf:resource="#Functional"/>
</owl:Class>
<owl:Class rdf:ID="ALGOL">
  <rdfs:subClassOf rdf:resource="#Procedural"/>
</owl:Class>
<owl:Class rdf:ID="IT-ledelse">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:Class rdf:ID="SmallTalk">
  <rdfs:subClassOf rdf:resource="#Object_oriented"/>
</owl:Class>
<owl:Class rdf:ID="IT-Arkitektur">
  <rdfs:subClassOf rdf:resource="#Data_og_IT"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#hasCompetency">
  <rdfs:range rdf:resource="#Competency"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CV"/>
        <owl:Class rdf:about="#JobAd"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>

```

```

</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasLocation">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CV"/>
        <owl:Class rdf:about="#JobAd"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasWorkplace">
  <rdfs:domain rdf:resource="#JobAd"/>
  <rdfs:range rdf:resource="#Workplace"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="#hasCompetency_level">
  <rdfs:domain rdf:resource="#Competency"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
              >3</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:rest rdf:parseType="Resource">
                      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                        >7</rdf:first>
                      <rdf:rest rdf:parseType="Resource">
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                          >8</rdf:first>
                      </rdf:rest>
                    </rdf:rest>
                  </rdf:rest>
                </rdf:rest>
              </rdf:rest>
            </rdf:rest>
          </rdf:rest>
        </rdf:rest>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
  <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >6</rdf:first>
  </rdf:rest>

```



```

    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</rdf:first>
  </rdf:rest>
</rdf:rest>
  <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >4</rdf:first>
</rdf:rest>
</rdf:rest>
  <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >2</rdf:first>
</rdf:rest>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:DatatypeProperty>
<owl:SymmetricProperty rdf:ID="isNeighbour">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Fylke"/>
  <rdfs:range rdf:resource="#Fylke"/>
  <owl:inverseOf rdf:resource="#isNeighbour"/>
</owl:SymmetricProperty>
<owl:FunctionalProperty rdf:about="#hasDegree">
  <rdfs:range rdf:resource="#Degree"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CV"/>
        <owl:Class rdf:about="#JobAd"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<Workplace rdf:ID="Objects_R_Us">
  <hasLocation>
    <Fylke rdf:ID="Møre_og_Romsdal">
      <isNeighbour>
        <Fylke rdf:ID="Sør-Trøndelag">
          <isNeighbour>
            <Fylke rdf:ID="Hedmark">
              <isNeighbour>
                <Fylke rdf:ID="Akershus">
                  <isNeighbour>

```

```

<Fylke rdf:ID="Oslo">
  <isNeighbour rdf:resource="#Akershus"/>
</Fylke>
</isNeighbour>
<isNeighbour>
  <Fylke rdf:ID="_Østfold">
    <isNeighbour rdf:resource="#Akershus"/>
    <isNeighbour rdf:resource="#Hedmark"/>
  </Fylke>
</isNeighbour>
<isNeighbour>
  <Fylke rdf:ID="Buskerud">
    <isNeighbour rdf:resource="#Akershus"/>
  <isNeighbour>
    <Fylke rdf:ID="Hordaland">
      <isNeighbour rdf:resource="#Buskerud"/>
    <isNeighbour>
      <Fylke rdf:ID="Sogn_og_Fjordane">
        <isNeighbour rdf:resource="#Buskerud"/>
      <isNeighbour>
        <Fylke rdf:ID="Oppland">
          <isNeighbour rdf:resource="#Hedmark"/>
          <isNeighbour rdf:resource="#Akershus"/>
          <isNeighbour rdf:resource="#Buskerud"/>
          <isNeighbour rdf:resource="#Møre_og_Romsdal"/>
          <isNeighbour rdf:resource="#Sør-Trøndelag"/>
          <isNeighbour rdf:resource="#Sogn_og_Fjordane"/>
        </Fylke>
      </isNeighbour>
    <isNeighbour rdf:resource="#Hordaland"/>
    <isNeighbour rdf:resource="#Møre_og_Romsdal"/>
  </Fylke>
</isNeighbour>
<isNeighbour>
  <Fylke rdf:ID="Rogaland">
    <isNeighbour>
      <Fylke rdf:ID="Aust-Agder">
        <isNeighbour>
          <Fylke rdf:ID="Vest-Agder">
            <isNeighbour rdf:resource="#Aust-Agder"/>
            <isNeighbour rdf:resource="#Rogaland"/>
          </Fylke>
        </isNeighbour>
      </Fylke>
    </isNeighbour>
  </Fylke>

```

```

</isNeighbour>
<isNeighbour rdf:resource="#Rogaland"/>
<isNeighbour>
  <Fylke rdf:ID="Telemark">
    <isNeighbour rdf:resource="#Rogaland"/>
    <isNeighbour rdf:resource="#Aust-Agder"/>
    <isNeighbour rdf:resource="#Buskerud"/>
    <isNeighbour rdf:resource="#Hordaland"/>
    <isNeighbour>
      <Fylke rdf:ID="Vestfold">
        <isNeighbour rdf:resource="#Aust-Agder"/>
        <isNeighbour rdf:resource="#Buskerud"/>
        <isNeighbour rdf:resource="#Telemark"/>
      </Fylke>
    </isNeighbour>
  </Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Vestfold"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Hordaland"/>
<isNeighbour rdf:resource="#Vest-Agder"/>
<isNeighbour rdf:resource="#Telemark"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Telemark"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Oppland"/>
<isNeighbour rdf:resource="#Sogn_og_Fjordane"/>
<isNeighbour rdf:resource="#Telemark"/>
<isNeighbour rdf:resource="#Vestfold"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Hedmark"/>
<isNeighbour rdf:resource="#Oppland"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#_Østfold"/>
<isNeighbour rdf:resource="#Sør-Trøndelag"/>
<isNeighbour rdf:resource="#Oppland"/>
</Fylke>

```

```

</isNeighbour>
<isNeighbour rdf:resource="#Møre_og_Romsdal"/>
<isNeighbour>
  <Fylke rdf:ID="Nord-Trøndelag">
    <isNeighbour>
      <Fylke rdf:ID="Nordland">
        <isNeighbour rdf:resource="#Nord-Trøndelag"/>
      <isNeighbour>
        <Fylke rdf:ID="Troms">
          <isNeighbour>
            <Fylke rdf:ID="Finnmark">
              <isNeighbour rdf:resource="#Troms"/>
            </Fylke>
          </isNeighbour>
        <isNeighbour rdf:resource="#Nordland"/>
      </Fylke>
    </isNeighbour>
  </Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Sør-Trøndelag"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Oppland"/>
</Fylke>
</isNeighbour>
<isNeighbour rdf:resource="#Oppland"/>
<isNeighbour rdf:resource="#Sogn_og_Fjordane"/>
</Fylke>
</hasLocation>
</Workplace>
<Workplace rdf:ID="Drift_R_us"/>
<Computer_Graphics rdf:ID="Photoshop_guru">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >8</hasCompetency_level>
</Computer_Graphics>
<CSS rdf:ID="CSS_5years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >5</hasCompetency_level>
</CSS>
<Webdesign_og_webmaster rdf:ID="Webdesigner">
  <hasCompetency rdf:resource="#Photoshop_guru"/>
  <hasCompetency>

```

```

<HTML rdf:ID="expert_web_coder">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</hasCompetency_level>
</HTML>
</hasCompetency>
<hasCompetency rdf:resource="#CSS_5years"/>
<hasDegree>
  <ToÅrig rdf:ID="toÅrig_degree">
    <owl:sameAs>
      <Fagbrev rdf:ID="fagbrev_degree">
        <owl:sameAs rdf:resource="#toÅrig_degree"/>
      </Fagbrev>
    </owl:sameAs>
  </ToÅrig>
</hasDegree>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Dine arbeidsoppgaver blir i utgangspunktet å estimere, utvikle og kvalitetssikre nettsider ved hjelp av CSS i vårt rammeverk for nettsidebygging i
publiseringsløsningen CorePublish.

```

Sentrale arbeidsoppgaver vil være:

\* For oss er design synonymt med brukeropplevelsen. Du brenner for web og ønsker å være blant de beste. Du er svært opptatt av design - både funksjonelt og visuelt. Dine kunnskaper innen fagområdene er høye

\* Du er vel bevandret i CSS, og er dyktig med dine verktøy.

\* Du kan Photoshop inn og ut.

Det er en fordel om du har kjennskap til Flash og actionscript.</rdfs:comment>

```

<hasLocation rdf:resource="#Vest-Agder"/>
<hasWorkplace>
  <Workplace rdf:ID="CoreTrek"/>
</hasWorkplace>
</Webdesign_og_webmaster>
<None rdf:ID="no_degree"/>
<owl:AllDifferent/>
<Workplace rdf:ID="Perl_R_Us">
  <hasLocation rdf:resource="#Oslo"/>
</Workplace>
<CV rdf:ID="Cplusplus_bachelor">
  <hasCompetency>
    <C_plusplus rdf:ID="C_plusplus_expert">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >8</hasCompetency_level>

```

```

</C_plusplus>
</hasCompetency>
<hasLocation rdf:resource="#Buskerud"/>
<hasDegree>
  <Bachelor rdf:ID="bachelor_degree">
    <owl:sameAs>
      <TreÅrig rdf:ID="treÅrig_degree">
        <owl:sameAs rdf:resource="#bachelor_degree"/>
      </TreÅrig>
    </owl:sameAs>
  </Bachelor>
</hasDegree>
</CV>
<Unix rdf:ID="Linux_5years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >5</hasCompetency_level>
</Unix>
<Database rdf:ID="SQL_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >3</hasCompetency_level>
</Database>
<CV rdf:ID="python_guru_w_master">
  <hasDegree>
    <Master rdf:ID="master_degree"/>
  </hasDegree>
  <hasCompetency>
    <Python rdf:ID="Python_guru">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >8</hasCompetency_level>
    </Python>
  </hasCompetency>
  <hasCompetency rdf:resource="#Linux_5years"/>
  <hasLocation rdf:resource="#Akershus"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Jeg heter Ronny Ponny og har tatt python-master ved NTNU.</rdfs:comment>
</CV>
<Nettverksadminstrasjon_og_sikkerhet rdf:ID="Nettverksadminstrasjon_og_sikkerhet_1">
  <hasCompetency>
    <Network rdf:ID="Network_3years">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3</hasCompetency_level>
    </Network>
  </hasCompetency>

```

```

</hasCompetency>
<hasCompetency>
  <Network rdf:ID="Active_Directory_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</hasCompetency_level>
  </Network>
</hasCompetency>
<hasCompetency>
  <Microsoft rdf:ID="Windows_2000_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</hasCompetency_level>
  </Microsoft>
</hasCompetency>
<hasLocation rdf:resource="#Akershus"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Pga. økt etterspørsel fra våre kunder etter IT-driftspersonell søker vi herved etter teknisk personell med erfaring fra 2. og 3.linje support/drift.

```

Sentrale arbeidsoppgaver vil være:

- \* Løse direkte brukerrorettede problemstillinger "on site"
- \* Installasjon/re-installasjon
- \* Konfigurasjon og backup
- \* Printer- og driverproblematikk
- \* Patching og flytting av brukere
- \* Feilsøking og feilretting av hard- og software
- \* Drift av Windows 2000 servere og Active directory

Ønskede kvalifikasjoner:

- \* 2-3 årig IT-utdanning med hovedvekt på drift/support
  - \* Kompetanse på Windows 2000/2003
  - \* Erfaring med Active directory
  - \* Utpregede samarbeidsevner og serviceinnstilling
  - \* Strukturert, ryddig og selvstendig
  - \* Gode kommunikasjonsevner, bidra sosialt til vekst i avdelingen og opptatt av "peek performance"
  - \* Gode engelskkunnskaper</rdfs:comment>
- ```

<hasWorkplace rdf:resource="#Drift_R_us"/>
<hasDegree rdf:resource="#treÅrig_degree"/>
</Nettverksadministrasjon_og_sikkerhet>
<java rdf:ID="Java_3years">
<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"

```

```

>3</hasCompetency_level>
</Java>
<Developement_IDE rdf:ID="Visual_Studio_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >3</hasCompetency_level>
</Developement_IDE>
<Unix rdf:ID="Mac_OSX_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >3</hasCompetency_level>
</Unix>
<CV rdf:ID="Skilled_Webdesigner">
  <hasLocation rdf:resource="#Oslo"/>
  <hasCompetency rdf:resource="#SQL_3years"/>
  <hasCompetency rdf:resource="#Photoshop_guru"/>
  <hasCompetency rdf:resource="#Visual_Studio_3years"/>
  <hasCompetency rdf:resource="#expert_web_coder"/>
  <hasCompetency rdf:resource="#CSS_5years"/>
  <hasDegree>
    <FemÅrig rdf:ID="femÅrig_degree"/>
  </hasDegree>
</CV>
<Helpdesk_og_teknisk_support rdf:ID="Helpdesk_og_teknisk_support_1">
  <hasCompetency>
    <Hardware rdf:ID="Computer_builder">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >6</hasCompetency_level>
    </Hardware>
  </hasCompetency>
  <hasCompetency rdf:resource="#Active_Directory_3years"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Xtra personell bemanner et teknisk kunde/supportsenter sentralt i Oslo, og vi har nå behov for nye medarbeidere på heltid.

```

Vi søker deg som ønsker å få relevant IT-erfaring og ser på dette som en god investering i din videre karriere. Supportsenteret støtter to av landets største aktører innen IT/Telecom, har 450 applikasjoner, og man vil få en bratt og rask teknisk læringskurve.

Teamet består av totalt 12 personer. Vi søker deg som har stor IT-interesse eller er nyutdannet og har evne til å lære raskt, og som vil komme inn i et faglig og sosialt meget godt miljø. Vi kan tilby svært gode utviklingsmuligheter for deg som vil.

Sentrale arbeidsoppgaver vil være:

- \* MS Office
- \* MS Win XP
- \* Active Directory
- \* Passord / tilganger



\* VPN

\* Mottak og håndtering av henvendelser vedr nettverksdrift, kabling, trådløst etc.

\* Supportering av både stasjonære og bærbare pc'er</rdfs:comment>

```
<hasCompetency>
  <Productivity rdf:ID="MS_Office_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3</hasCompetency_level>
  </Productivity>
</hasCompetency>
<hasLocation rdf:resource="#Oslo"/>
<hasWorkplace>
  <Workplace rdf:ID="Skoyen_TechSupport"/>
</hasWorkplace>
<hasCompetency rdf:resource="#Network_3years"/>
<hasCompetency>
  <Microsoft rdf:ID="Windows_XP_5years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >5</hasCompetency_level>
  </Microsoft>
</hasCompetency>
<hasDegree rdf:resource="#no_degree"/>
</Helpdesk_og_teknisk_support>
<System_og_applikasjonsutvikling rdf:ID="Game_developer">
  <hasLocation rdf:resource="#Oslo"/>
  <hasCompetency rdf:resource="#Linux_5years"/>
  <hasCompetency rdf:resource="#SQL_3years"/>
  <hasCompetency rdf:resource="#Visual_Studio_3years"/>
  <hasCompetency rdf:resource="#C_plusplus_expert"/>
  <hasWorkplace>
    <Workplace rdf:ID="FunCom">
      <hasLocation rdf:resource="#Oslo"/>
    </Workplace>
  </hasWorkplace>
</rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Game Programmer
```

If you are a skilled coder and want to be part of making the world's top selling next-generation titles, please get in contact right away.

Qualifications:

\* Service minded attitude

- \* Visual Studio 6/7
- \* C++
- \* RPG knowledge
- \* Ability to work closely with other professions (like designers and graphic artists)

Preferable knowledge:

- \* SQL
- \* Linux
- \* Game experience

Common for all positions, you need:

- \* to have good written and oral communication skills
- \* to have a passion for games
- \* to relocate to Oslo
- \* to have good English written and oral skills
- \* to be able to handle multiple tasks and work well under pressure
- \* to be able to work independently and to be self-directing

If you lack some of the above-mentioned qualifications, but consider yourself a highly skilled coder and a fast learner that Funcom can't be without, apply as well.

Please apply in English (email only). Attach your resume, samples of your work and source-code. Runtimes are not really needed, but is a nice add-on. Please state clearly in the application what kind of position you are applying for. Applications lacking relevant work samples (or link to a website with such material) will not be reviewed.

Please allow a few weeks processing time.</rdfs:comment>

```

<hasDegree rdf:resource="#no_degree"/>
</System_og_-applikasjonsutvikling>
<Phd rdf:ID="Phd_degree"/>
<CV rdf:ID="Database_guy">
<hasDegree rdf:resource="#bachelor_degree"/>
<hasCompetency>
<Security rdf:ID="Security_3years">
<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>3</hasCompetency_level>
</Security>
</hasCompetency>
<hasCompetency rdf:resource="#Network_3years"/>
<hasCompetency>
<Database rdf:ID="MySQL_expert">
<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>7</hasCompetency_level>
</Database>

```

```

</hasCompetency>
<hasCompetency rdf:resource="#SQL_3years"/>
<hasCompetency>
  <Database rdf:ID="Oracle_Designer_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3</hasCompetency_level>
  </Database>
</hasCompetency>
<hasLocation rdf:resource="#Oppland"/>
</CV>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Fylke rdf:about="#Oppland"/>
    <Fylke rdf:about="#Vest-Agder"/>
    <Fylke rdf:about="#Aust-Agder"/>
    <Fylke rdf:about="#Møre_og_Romsdal"/>
    <Fylke rdf:about="#Nord-Trøndelag"/>
    <Fylke rdf:about="#Sør-Trøndelag"/>
    <Fylke rdf:about="#Nordland"/>
    <Fylke rdf:about="#_Østfold"/>
    <Fylke rdf:about="#Vestfold"/>
    <Fylke rdf:about="#Oslo"/>
    <Fylke rdf:about="#Sogn_og_Fjordane"/>
    <Fylke rdf:about="#Telemark"/>
    <Fylke rdf:about="#Hordaland"/>
    <Fylke rdf:about="#Rogaland"/>
    <Fylke rdf:about="#Hedmark"/>
    <Fylke rdf:about="#Akershus"/>
    <Fylke rdf:about="#Buskerud"/>
    <Fylke rdf:about="#Troms"/>
    <Fylke rdf:about="#Finnmark"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Microsoft rdf:ID="Windows_NT_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</hasCompetency_level>
</Microsoft>
<Doctoral rdf:ID="doctoral_degree"/>
<IT-Arkitektur rdf:ID="IT-Architect">
  <hasWorkplace>
    <Workplace rdf:ID="Compass_Human_Resources_AS"/>
  </hasWorkplace>

```

```

<hasLocation rdf:resource="#Oslo"/>
<hasCompetency>
  <Leadership rdf:ID="Leadership_expert">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >7</hasCompetency_level>
  </Leadership>
</hasCompetency>
<hasCompetency rdf:resource="#Linux_5years"/>
<hasCompetency rdf:resource="#Security_3years"/>
<hasDegree rdf:resource="#femÅrig_degree"/>
<hasCompetency>
  <Perl rdf:ID="Perl_5years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >5</hasCompetency_level>
  </Perl>
</hasCompetency>
<hasCompetency rdf:resource="#Network_3years"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Sr Systemarkitekt UNIX

```

Beskrivelse:

SPENNENDE TEKNISKE STILLINGER I ET AV NORGES LEDENDE MILJØER INNEN DRIFT AV PLATTFORMER, APPLIKASJONER OG NETTVERK

Driftsmiljøet består i dag av over 200 medarbeidere som har ansvar for drift og vedlikehold av store plattformer, sammensatt infrastruktur og forretningskritiske applikasjoner basert på moderne teknologi. Våre medarbeidere har også sentrale roller i banebrytende teknologiutviklingsprosjekter.

PLATTFORMSDRIFT OSLO:

Seksjonen består i dag av 19 teknisk faglig tunge medarbeidere i Oslo og har basis driftsansvar for HW/OS samt databaseplattformer i Skandinavia.

Arbeidsoppgaver:

Vi har behov å toppe laget med nok en dyktig Senior Systemarkitekt/ System Administrator innen drift av Solaris- og Linux baserte løsninger. Du vil få muligheten til å bli med på IT-satsingen til et av verdens meste spennende teknologiskaper.

- \* Du vil være ansvarlig for leveranser og daglig drift, samt teknisk eskaleringspunkt ved utfall og kritiske hendelser.
- \* Du vil ha en rolle som fagekspert i prosjekter, samt mot store kunder og leverandører.
- \* Ansvar for arkitektur, installasjon, oppsett og drift av HW/OS, samt applikasjonsoptimalisering og leveransekoordinering med andre team
- \* Deltagelse i vaktordning vil bli kunne bli aktuelt.

Kvalifikasjoner:

- \* Teknisk utdannelse tilsv. ingeniør /sivilingeniør/Cand.scient.

- \* God generell teknisk forståelse.
- \* Det er ønskelig med minimum 5 års relevant yrkes erfaring.
- \* Erfaring med Perl og Shell scripting, samt C-programmering og kompilering av open source programvare
- \* Basis forståelse av systemsikkerhet, IP-nettverk, DNS og serverbaserte analyseverktøy

Personlige egenskaper:

- \* Selvstendig, kreativ og entusiastisk.
- \* Gode evner til å samarbeide med andre.
- \* Positiv innstilling, med godt humør.

Vi tilbyr:

- \* Utfordringer i et av Nordens største og mest teknisk avanserte IT/Datakommunikasjonsmiljøer
- \* Samarbeid/samhandling med internasjonale samarbeidspartnere og leverandører
- \* Kompetanseoppbygging/kurs innen fagområdene med individuell oppfølging og tilpasning
- \* Konkurransedyktige betingelser.
- \* Et uformelt og hektisk fagmiljø.
- \* Gode kollegaer som verdsetter kunnskapsdeling, fagdiskusjoner og et sosialt godt arbeidsmiljø.

Kvinner oppfordres til å søke.

For mer informasjon om stillingene, ta gjerne kontakt med Compass ved Bjørn Solum, tlf 901 75 692 eller Stein-Sverre Wold, tlf 906 88 270. For å søke på denne stillingen benytt det elektroniske søknadsskjemaet nederst på denne siden.</rdfs:comment>

```

<hasCompetency>
  <Shell rdf:ID="Shell_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3</hasCompetency_level>
  </Shell>
</hasCompetency>
</IT-Arkitektur>
<Workplace rdf:ID="Synergi"/>
<Service_oriented rdf:ID="Service_minded_experienced">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >4</hasCompetency_level>
</Service_oriented>
<Object_oriented rdf:ID="OO-programming_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</hasCompetency_level>
</Object_oriented>
<Databaseadministrasjon rdf:ID="Databaseadministrasjon_1">
  <hasLocation rdf:resource="#Rogaland"/>
  <hasWorkplace rdf:resource="#Synergi"/>

```

```

<hasCompetency rdf:resource="#Security_3years"/>
<hasCompetency rdf:resource="#SQL_3years"/>
<hasCompetency rdf:resource="#Oracle_Designer_3years"/>
<hasDegree rdf:resource="#master_degree"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Synergi Database Developer:

* Data modelling
* Development of new content, such as new fields, screens, reports and applications
* Migration of existing customer databases
* Technical administration related to nightly database builds for development and test environment
* General database administration
* Provide support related to upgrades, performance issues, and other support issues handles by R&D</rdfs:comment>
</Databaseadministrasjon>
<Perl rdf:ID="Perl_guru">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >8</hasCompetency_level>
</Perl>
<System_og_-applikasjonsutvikling rdf:ID="Perl_koder">
  <hasDegree rdf:resource="#bachelor_degree"/>
  <hasCompetency>
    <Perl rdf:ID="Perl_3years">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >3</hasCompetency_level>
    </Perl>
  </hasCompetency>
  <hasLocation rdf:resource="#Buskerud"/>
  <hasWorkplace rdf:resource="#Perl_R_Us"/>
</System_og_-applikasjonsutvikling>
<System_og_-applikasjonsutvikling rdf:ID="OO-programmer">
  <hasWorkplace rdf:resource="#Objects_R_Us"/>
  <hasDegree rdf:resource="#bachelor_degree"/>
  <hasLocation rdf:resource="#Hedmark"/>
  <hasCompetency rdf:resource="#OO-programming_3years"/>
</System_og_-applikasjonsutvikling>
<CV rdf:ID="Microsoft_guru">
  <hasLocation rdf:resource="#Akershus"/>
  <hasDegree rdf:resource="#bachelor_degree"/>
  <hasCompetency rdf:resource="#MS_Office_3years"/>
  <hasCompetency rdf:resource="#Windows_NT_3years"/>
  <hasCompetency rdf:resource="#Windows_2000_3years"/>
  <hasCompetency>

```

```

<Microsoft rdf:ID="Windows_Vista_2years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >2</hasCompetency_level>
</Microsoft>
</hasCompetency>
<hasCompetency rdf:resource="#Windows_XP_5years"/>
<hasCompetency>
  <Visual_Basic rdf:ID="Visual_Basic_3years">
    <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</hasCompetency_level>
  </Visual_Basic>
</hasCompetency>
</CV>
<CV rdf:ID="Bobs_CV">
  <hasLocation rdf:resource="#Oslo"/>
  <hasDegree rdf:resource="#bachelor_degree"/>
  <hasCompetency rdf:resource="#Java_3years"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Jeg heter BOB, og har besatt en javaprogrammerer for øyeblikket.</rdfs:comment>
</CV>
<CV rdf:ID="perl_guru_w_bachelor">
  <hasDegree rdf:resource="#bachelor_degree"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Jeg er en mann som har jobbet med Perl i 30 år, og jeg har bachelorgrad i kjole- og drakt-søm.</rdfs:comment>
  <hasCompetency rdf:resource="#Perl_guru"/>
  <hasLocation rdf:resource="#Oslo"/>
</CV>
<Workplace rdf:ID="Sun">
  <hasLocation rdf:resource="#Oppland"/>
</Workplace>
<owl:DataRange>
  <owl:oneOf rdf:parseType="Resource">
    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  </owl:oneOf>
</owl:DataRange>
<Development_IDE rdf:ID="Apache_Velocity_2years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >2</hasCompetency_level>
</Development_IDE>
<ActionScript rdf:ID="ActionScript_3years">

```

```

<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>3</hasCompetency_level>
</ActionScript>
<Delphi rdf:ID="Delphi_3years">
<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>3</hasCompetency_level>
</Delphi>
<JScript rdf:ID="JScript_2years">
<hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>2</hasCompetency_level>
</JScript>
<System_og_-applikasjonsutvikling rdf:ID="java_programmer">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Javautvikler

```

Hovedoppgaven vil være javautvikling. Både søkefront og salgssystem samt mot eksterne partnere.

Oppgaver:

Utvikling av web-applikasjoner basert på moderne Open Source-teknologi

Delta i teknologivalg og plattformarkitektur

Skreddersy søketeknologi mot eksterne partnere

Kvalifikasjoner:

Behersker godt JAVA og HTML

Gjerne erfaring med CSS, Velocity og Struts 2

Vi ønsker at du har 2 års erfaring med utvikling av disse verktøyene

Kan gjerne være self made - ingen formell utdanning kreves så lenge kandidaten er drivende dyktig og har rett erfaring

Komfortabel med kommandolinjer i Linux

Gjerne erfaring med SCRUM

Personlige egenskaper:

Inspirerende og positiv

Resultatorientert

Selvdreven</rdfs:comment>

```

<hasLocation rdf:resource="#Oslo"/>
<hasWorkplace rdf:resource="#Sun"/>
<hasDegree rdf:resource="#bachelor_degree"/>
<hasCompetency rdf:resource="#Java_3years"/>
<hasCompetency rdf:resource="#expert_web_coder"/>
<hasCompetency rdf:resource="#Apache_Velocity_2years"/>
<hasCompetency rdf:resource="#Linux_5years"/>
</System_og_-applikasjonsutvikling>

```



```

<JavaScript rdf:ID="JavaScript_3years">
  <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >3</hasCompetency_level>
</JavaScript>
<CV rdf:ID="Web_Coder">
  <hasLocation rdf:resource="#Aust-Agder"/>
  <hasCompetency rdf:resource="#Mac_OSX_3years"/>
  <hasCompetency rdf:resource="#JavaScript_3years"/>
  <hasCompetency rdf:resource="#ActionScript_3years"/>
  <hasCompetency>
    <Computer_Graphics rdf:ID="Macromedia_Flash_4years">
      <hasCompetency_level rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >4</hasCompetency_level>
    </Computer_Graphics>
  </hasCompetency>
  <hasDegree rdf:resource="#bachelor_degree"/>
</CV>
<CV rdf:ID="IT_Manager">
  <hasDegree rdf:resource="#master_degree"/>
  <hasLocation rdf:resource="#Rogaland"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Hei, jeg heter Adol Fhitler, og jeg har mastergrad i ledelse.</rdfs:comment>
  <hasCompetency rdf:resource="#Leadership_expert"/>
</CV>
<CV rdf:ID="PhD_Delphi">
  <hasDegree rdf:resource="#doctoral_degree"/>
  <hasLocation rdf:resource="#Finmark"/>
  <hasCompetency rdf:resource="#Delphi_3years"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Mitt navn er Doktor Drøvel, og jeg har programmert Delphi i syv år.</rdfs:comment>
</CV>
<owl:AllDifferent/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430) http://protege.stanford.edu -->

```

## Source code for jsp files

### Index.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<html>
  <head>
    <title>fredOnt</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <link href="fredstyle.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <table align="center">
      <jsp:include page="header.html" />
      <form name="queryform" method="post">
        <tr>
          <td align="center" colspan="10" class="searchtop">
            Find <select name="dropdown">
              <option value="JobAd">JobAds</option>
              <option value="CV">CVs</option>
            </select>
            with keyword <input type="text" name="keyword" size="20"></input>
          </td>
        </tr>
        <tr>
          <td align="center" colspan="10" class="searchbottom">
            <input type="reset" value="blank ut">
            <input type="submit" value="s&oslash;k">
          </td>
        </tr>
      </form>
      <jsp:include page="jena_query.jsp" flush="true" />
    </table>
  </body>
</html>
```

## jena\_query.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="com.hp.hpl.jena.ontology.*" %>
<%@ page import="com.hp.hpl.jena.rdf.model.*" %>
<%@ page import="com.hp.hpl.jena.util.*" %>
<%@ page import="com.hp.hpl.jena.query.*" %>
<%@ page import="com.hp.hpl.jena.n3.IRIResolver" %>
<%
String dataprfx = "prefix data: <http://folk.uio.no/fredrhal/ontology/Data_IT.owl#> ";
String rdfprfx = "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ";
String rdfspfx = "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ";
String vari = "";
String select = "";

/* backward support for v1
if(request.getParameter("sokestreng") != ""){
    String select = request.getParameter("sokestreng");
    inputQuery = select;
}
*/

session.setAttribute("category", request.getParameter("dropdown"));

if(request.getParameter("keyword") != ""){
    select = "select distinct ?result where { ";

    // stringtokenizer that splits keyword input on comma+space
    StringTokenizer st = new StringTokenizer(request.getParameter("keyword"), ", ");
    while(st.hasMoreTokens()){
        String keyword = st.nextToken();

        select += "{ ?result rdf:type data:" + request.getParameter("dropdown") + " . " +
            "filter regex(str(?result), '" + keyword + "', 'i') } union " +

            // competencies as keyword
            "{ ?result rdf:type data:" + request.getParameter("dropdown") + " . " +
            "?result data:hasCompetency ?competency . " +
            "filter regex(str(?competency), '" + keyword + "', 'i') } union " +
```

```

// class as keyword
"{ ?result rdf:type data:" + request.getParameter("dropdown") + " . " +
"?result data:hasCompetency ?competency . " +
"?competency rdf:type ?class . " +
"filter regex(str(?class), '" + keyword + "', 'i') } union " +

// superclass as keyword
"{ ?result rdf:type data:" + request.getParameter("dropdown") + " . " +
"?result data:hasCompetency ?competency . " +
"?competency rdf:type ?class . " +
"?class rdfs:subClassOf ?superclass . " +
"filter regex(str(?superclass), '" + keyword + "', 'i') } union " +

// location as keyword
"{ ?result rdf:type data:" + request.getParameter("dropdown") + " . " +
"?result data:hasLocation ?location . ";

if(st.hasMoreTokens()){
    select += "filter regex(str(?location), '" + keyword + "', 'i') } union ";
} else {
    select += "filter regex(str(?location), '" + keyword + "', 'i') ";
}
}
select += "};";
} else {
    select = "select distinct ?result where { ?result rdf:type data:" + request.getParameter("dropdown") + " } ";
}

```

```
String inputQuery = dataprfx + rdfrprfx + rdfsprfx + select;
```

```
long searchStart = System.currentTimeMillis();
```

```

try {
    OntModel m = ModelFactory.createOntologyModel();
    OntDocumentManager dm = m.getDocumentManager();
    dm.addAltEntry( "http://folk.uio.no/fredrhal/ontology/", "file:/usr/share/tomcat5.5-
webapps/ROOT/fredOnt/Data_IT.owl" );
    m.read( "http://folk.uio.no/fredrhal/ontology/" );

    Query query = QueryFactory.create(inputQuery);
    QueryExecution qexec = QueryExecutionFactory.create(query, m);

    try {
        ResultSet resultSet = qexec.execSelect();

```

```

List resultVars = resultSet.getResultVars();
int colCount = resultVars.size();

out.print("<tr>");
for (int i = 0; i < colCount; i++) {
out.print("<td class=\"header2\" colspan=\"4\">" + (String)resultVars.get(i) + "</td>");
}

out.print("</tr>");

while (resultSet.hasNext()) {
    out.print("<tr>");
    QuerySolution querySolution = resultSet.nextSolution();
    for (int i = 0; i < colCount; i++) {
        RDFNode node = querySolution.get((String)resultVars.get(i));
        out.print("<td class=\"sqltabell\">");
        if (node.isLiteral()) {
            Literal literal = (Literal)node;
            out.print(literal.getLexicalForm());
            vari = literal.getLexicalForm();
        } else if (node.isAnon()) {
            Resource resource = (Resource)node;
            out.print(resource.getId());
            vari = resource.getId().toString();
        } else if (node.isURIResource()) {
            Resource resource = (Resource)node;
            out.print(resource.getLocalName());
            vari = resource.getLocalName();
        }
    }
}

%>

</td><td class="sqltabell">
    <a href="competencies.jsp?comp=<%= vari %>">competencies</a>
</td><td class="sqltabell">
    <a href="description.jsp?desc=<%= vari %>">description</a>
</td>
<!--<td class="sqltabell">
    <a href="siblings.jsp?sibs=<%= vari %>">similar</a>
-->
</td>

%>
}
}

long searchTime = System.currentTimeMillis()-searchStart;

%>

```

```

        </tr>
        <tr>
            <td align="center" colspan="10" class="sqltabell">
                Search completed in : <%= searchTime %>ms
            </td>
        </tr>
    <%

```

```

        } finally {
            qexec.close();
        }
    } catch (Exception e) {
        out.print("<tr><td><xmp>" + e + "</xmp></td></tr>");
    }
}

```

```
%>
```

## flush.jsp

```

<html>
    <head>
        <title>fredOnt</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <link href="fredstyle.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <table align="center">
            <jsp:include page="header.html" />
            <tr>
                <% session.invalidate(); %>
                <td class="header2">Session has been flushed.</td>
            </tr>
        </table>
    </body>
</html>

```

## competencies.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="com.hp.hpl.jena.ontology.*" %>
<%@ page import="com.hp.hpl.jena.rdf.model.*" %>
<%@ page import="com.hp.hpl.jena.util.*" %>
<%@ page import="com.hp.hpl.jena.query.*" %>
<%@ page import="com.hp.hpl.jena.n3.IRIResolver" %>
<html>
    <head>
        <title>fredOnt</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <link href="fredstyle.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <table align="center">
            <jsp:include page="header.html" />

<%
String dataprfx = "prefix data: <http://folk.uio.no/fredrhal/ontology/Data_IT.owl#> ";
String rdfprfx = "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ";
String rdfsprfx = "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ";

String select = "select distinct ?competency ?category ?level ?supercat ?location where { " +
    "data:" + request.getParameter("comp") + " ?function ?competency . " +
    "?a data:hasCompetency ?competency . " +
    "?competency data:hasCompetency_level ?level . " +
    "?competency rdf:type ?category . " +
    "?category rdfs:subClassOf ?supercat . " +

    // bad triks, location kommer i hver competency-kolonne..
    "data:" + request.getParameter("comp") + " data:hasLocation ?location . " +
    "}";

String inputQuery = dataprfx + rdfprfx + rdfsprfx + select;

    try {
        OntModel m = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, null);
        OntDocumentManager dm = m.getDocumentManager();
        dm.addAltEntry( "http://folk.uio.no/fredrhal/ontology/", "file:/usr/share/tomcat5.5-
webapps/ROOT/fredOnt/Data_IT.owl" );
```

```

m.read( "http://folk.uio.no/fredrhal/ontology/" );

Query query = QueryFactory.create(inputQuery);
QueryExecution qexec = QueryExecutionFactory.create(query, m);

try {
    ResultSet resultSet = qexec.execSelect();
    List resultVars = resultSet.getResultVars();
    int colCount = resultVars.size();

    out.print("<tr>");
    for (int i = 0; i < colCount; i++) {
out.print("<td class=\"header2\">" + (String)resultVars.get(i) + "</td>");
    }

    out.print("</tr>");

    int y = 0;
    while (resultSet.hasNext()) {
        out.print("<tr>");
        QuerySolution querySolution = resultSet.nextSolution();
        for (int i = 0; i < colCount; i++) {
            RDFNode node = querySolution.get((String)resultVars.get(i));
            out.print("<td class=\"sqltabell\">");
            if (node.isURIResource()) {
                Resource resource = (Resource)node;
                out.print(resource.getLocalName());
                session.setAttribute("competency"+y, resource.getLocalName());
                y++;
            }
            else if (node.isLiteral()){
                Literal lit = (Literal)node;
                out.print(lit.getInt());
                session.setAttribute("compLevel"+y, lit.getInt());
                y++;
            }
            out.print("</td>");
        }
        out.print("</tr>");
    }
    session.setAttribute("items", y);
    if (session.getAttribute("category").toString().equals("JobAd")){
        out.print("<tr><td class=\"header3\" colspan=\"5\"><a href=\"matches.jsp\">Matching
CVs</a></td></tr>");
    }
}

```



```
        }
    } finally {
        qexec.close();
    }
} catch (Exception e) {
    out.print("<tr><td><xmp>" + e + "</xmp></td></tr>");
}
%>
</table>
</body>
</html>
```

## matching.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="com.hp.hpl.jena.ontology.*" %>
<%@ page import="com.hp.hpl.jena.rdf.model.*" %>
<%@ page import="com.hp.hpl.jena.util.*" %>
<%@ page import="com.hp.hpl.jena.query.*" %>
<%@ page import="com.hp.hpl.jena.n3.IRIResolver" %>
<html>
    <head>
        <title>fredOnt</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <link href="fredstyle.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <table align="center">
            <jsp:include page="header.html" />

<%
session.setAttribute("category", "CV");
int items = (Integer) session.getAttribute("items");

out.print("<tr><td class='\"header2\"' colspan='\"5\"'>Match criterias</td></tr><tr>");
for(int i=1; i<=items; i++){
    if(session.getAttribute("competency"+(i-1)) != null){
        out.print("<td class='\"sqltabell\"'>");
        out.print(session.getAttribute("competency"+(i-1)));
        out.print("</td>");
    }
    if(session.getAttribute("compLevel"+(i-1)) != null){
        out.print("<td class='\"sqltabell\"'>");
        out.print(session.getAttribute("compLevel"+(i-1)));
        out.print("</td>");
    }
    if(i%5 == 0){
        out.print("</tr><tr>");
    }
}

String datapfx = "prefix data: <http://folk.uio.no/fredrhal/ontology/Data_IT.owl#> ";
String rdfpfx = "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ";
```

```
String rdfsprfx = "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>";
```

```
String select = "select distinct ?klasse ?kompNavn ?cvNavn ?kompNivaa ?superklasse where {";
```

```
for(int i = 0; i < items-1; i++){
```

```
    if(session.getAttribute("competency"+i) != null){
```

```
        select += "{ ?cvNavn rdfs:subClassOf data:" + session.getAttribute("competency"+i) + " . " +
```

```
            "?kompNavn rdf:type ?cvNavn . " +
```

```
            "?klasse data:hasCompetency ?kompNavn . " +
```

```
            "?kompNavn data:hasCompetency_level ?kompNivaa . " +
```

```
            "?klasse rdf:type data:CV . " +
```

```
            "?cvNavn rdfs:subClassOf ?superklasse } union ";
```

```
    }
```

```
    select += "{ ?cvNavn data:hasCompetency_level data:" + session.getAttribute("compLevel"+i) + " } union ";
```

```
}
```

```
select += "{ ?cvNavn rdfs:subClassOf data:" + session.getAttribute("competency"+(items-1)) +
```

```
    " . ?kompNavn rdf:type ?cvNavn . " +
```

```
    "?klasse data:hasCompetency ?kompNavn . " +
```

```
    "?kompNavn data:hasCompetency_level ?kompNivaa . " +
```

```
    "?klasse rdf:type data:CV } " +
```

```
    "};
```

```
String inputQuery = dataprfx + rdfsprfx + rdfsprfx + select;
```

```
try {
```

```
    OntModel m = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, null);
```

```
    OntDocumentManager dm = m.getDocumentManager();
```

```
    dm.addAltEntry( "http://folk.uio.no/fredrhal/ontology/", "file:/usr/share/tomcat5.5-  
webapps/ROOT/fredOnt/Data_IT.owl" );
```

```
    m.read( "http://folk.uio.no/fredrhal/ontology/" );
```

```
    Query query = QueryFactory.create(inputQuery);
```

```
    QueryExecution qexec = QueryExecutionFactory.create(query, m);
```

```
try {
```

```
    ResultSet resultSet = qexec.execSelect();
```

```
    List resultVars = resultSet.getResultVars();
```

```
    int colCount = resultVars.size();
```

```
    int score;
```

```
    int res = 0;
```

```
    boolean notThere;
```

```
    while (resultSet.hasNext()) {
```

```

score = 0;
notThere = true;
QuerySolution querySolution = resultSet.nextSolution();

RDFNode cvInstanse = querySolution.get((String)resultVars.get(0));
RDFNode kompInstanse = querySolution.get((String)resultVars.get(1));
RDFNode klasse = querySolution.get((String)resultVars.get(2));
RDFNode kompLevel = querySolution.get((String)resultVars.get(3));
RDFNode superkl = querySolution.get((String)resultVars.get(4));

Resource cvNavn = (Resource)cvInstanse;
Resource kompNavn = (Resource)kompInstanse;
Resource klasNavn = (Resource)klasse;
Literal kompNiva = (Literal)kompLevel;
Resource supKlNav = (Resource)superkl;

// hvis navn finnes i session, hent inn navn og score
if(session.getAttribute(cvNavn.getLocalName()) != null ){
    score += (Integer)session.getAttribute(cvNavn.getLocalName());
    notThere = false;
}

// match score calculation
for(int ii=0; ii<items;ii++){
    if(session.getAttribute("competency"+ii) != null){
        if(supKlNav.getLocalName().equals(session.getAttribute("competency"+ii))){
            score += 5;
        }
        if(klasNavn.getLocalName().equals(session.getAttribute("competency"+ii))){
            score += 10;
            if(session.getAttribute("compLevel"+(ii+1)) != null){
                if(kompNiva.getInt() >=
(Integer)session.getAttribute("compLevel"+(ii+1))){
                    score += 2;
                }
                if(kompNiva.getInt() <
(Integer)session.getAttribute("compLevel"+(ii+1))){
                    score -= 2;
                }
            }
        }
    }
}
}

```

```

        // lagre i session
        session.setAttribute(cvNavn.getLocalName(), score);
        if(notThere){
            session.setAttribute("result"+res, cvNavn.getLocalName());
            res++;
        }
    }
}

%>
</tr><tr><td class="header2" colspan="5">Matching CVs</td></tr>
<tr><td class="header4" colspan="3">CV</td><td class="header4" colspan="2">Match score</td></tr>
<%

        // hente resultatene for listegenerering
        int tell = 0;
        String resNavn;
        int resScore;
        while(session.getAttribute("result"+tell) != null){
            resNavn = (String)session.getAttribute("result"+tell);
            resScore = (Integer)session.getAttribute(resNavn);
            out.print("<td class='sqltabell' " colspan='3'>");
            out.print("<a href='competencies.jsp?comp="+resNavn+" '>"+resNavn+"</a>");
            out.print("</td>");
            out.print("<td class='sqltabell' " colspan='3'>" + resScore + "</td>");
            out.print("</tr>");
            tell++;
        }
    } finally {
        qexec.close();
    }
} catch (Exception e) {
    out.print("<tr><td><xmp>" + e + "</xmp></td></tr>");
}

%>

</table>
</body>
</html>

```