Leveraging Corporate Skill Knowledge — From ProPer to OntoProPer

York Sure, Alexander Maedche, and Steffen Staab
Institute AIFB
University of Karlsruhe
76128 Karlsruhe
Germany
{sure, maedche, staab}@aifb.uni-karlsruhe.de

Abstract

Skill management systems serve as technical platforms for mostly, though not exclusively, corporate-internal market places for skills and know-how. The systems are typically built on top of a database that contains profiles of employees and applicants. Thus, the skills may be retrieved through database queries. However, these approaches incur two major problems, *viz.* the finding of *approximate matches* and the *maintenance* of skill data. In this paper we describe two systems that leverage corporate skill knowledge by offering advanced means for both. We present *ProPer* that uses means from decision theory to allow for compensate skill matching. Then, we describe *OntoProPer* that combines these methods with intelligent means for inferencing of skill data. For the latter an ontology provides background knowledge, *i.e.* conceptual structures and rules, which supplement the skill database with ground and inferred facts from secondary information, such as project documents. These supplement facts reduce maintenance efforts since much secondary information is gathered in the organizational memory through common working tasks.

Proc. of the Third Int. Conf. on Practical Aspects of Knowledge Management (PAKM2000) Basel, Switzerland, 30-31 Oct. 2000, (U. Reimer ed.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-34/

1 Introduction

Human Resource Management (HRM) is a key factor for success in knowledge-intensive companies. The basic and typically foremost problem is to find the right person for the job. Then, the HR management should recognize expertise gaps in the company and plan further HRM development in order to get the most leverage out of the company's human capital.

To support the HRM with keeping track of the company's intellectual assets, to help managers with finding people for their projects, and to facilitate that all employees find experts for particular problems, many knowledge-intensive companies have started to employ *skill management systems*. These systems are typically database applications that allow to search for people with particular skills or expertise using some form interface to database entries. Notwithstanding their benefits, this IT support for skill management is often plagued by two non-trivial problems.

First, while there are many uses for finding persons whose skills match exactly a particular database search, more often the problem that a HR manager or a project manager must face is to find someone who *roughly* matches some given requirements. Second, even when a process is established that ensures that the skills of all employees are found in the database, the maintenance of the database remains difficult. People work in new projects, they acquire new expertise, but they tend not to update the skill management database!

Both of these problems also occurred in a practical setting at a large IT consulting company (approximately 500 employees) that we had to deal with. The task of the first author was to develop and implement a skill management system that allowed the HR manager and the project managers, who have requirements

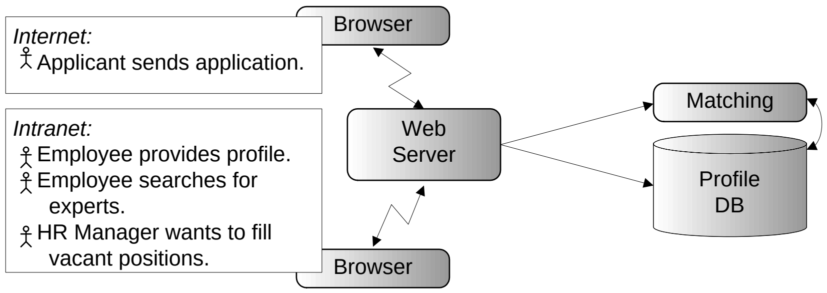

Internet:
Applicant sends application.
Intranet:
Employee provides profile.
Employee searches for experts.
HR Manager wants to fill vacant positions.
Browser
Web Server
Browser
Matching
Profile DB

Figure 1: Architecture of ProPer

for particular (project) positions, to retrieve "good" matches from a database of people's profiles. The profiles were about current employees, but they also came from people outside the company who filed their applications via the company's extranet.

We have developed a scheme that allowed to match between requirements and skill profiles using various methods from decision support systems, thus solving the first problem of *approximate matching*. This application, called **ProPer**, is in everyday use in the company right now and has proved very successful, especially with regard to the selection of applicants from outside the company.

Still, there has remained the problem of maintaining skill profiles. Hence, we have conceived complementary methods for provisioning up-to-date information to the skill database. The principal idea here was that employees produce a lot of *implicit knowledge* about themselves in their daily work and that this output may be an interesting source for deriving knowledge about the people's expertise. For instance, employees write project reports or they document ongoing projects in order to meet their customers' requirements, thus referring to the positions they had, the tools and technology they used and the companies they worked for. Exploiting *metadata* from these documents and using an *ontology* in order to structure document metadata, our prototype system, **OntoProPer**, draws inferences in order to derive or update the knowledge about individual's skills and integrates it with the skill matching capabilities of ProPer.

In the rest of this paper, we will first introduce the *architectures* of ProPer and OntoProPer (*cf.* Section 2). Then (Section 3), we describe the techniques used for skill matching in ProPer and provide a detailed example of how matching between profiles is performed. Section 4 shows the foundations of OntoProPer. We here explain the relationship between skills, ontology and metadata and show an example of how background knowledge is used to bring inferences on metadata to the skill matching component. After giving an overview of related work we conclude.

2 Architecture of our case study

In this Section we present the architectures of our application ProPer which corporates matching capabilities and our prototype OntoProPer, which aims at the integration of matching capabilities with means for easier and more comprehensive maintenance of skill data.

2.1 ProPer: Matching obvious skills

The architecture of ProPer is shown in Figure 1. It's general purpose was to introduce skill management to a particular IT company. Four different use cases exhibit requirements for the architecture of ProPer. They show different types of users involved in the skill management setting and problems to be solved by our system: two of the use cases show different requirements for *providing* skill profiles (1, 2) and the other two requirements for *accessing* stored skills (3, 4):

1. An applicant sends his application containing his personal skill profile through the internet (*i.e.* via an electronic job market).

2. An employee provides his skill profile through the intranet (*i.e.* via web forms).

3. An employee seeks an expert within the company who has certain skills.

4. A Human Resource Manager wants to fill vacant positions (*i.e.* empty or new jobs).

On the infrastructure level these requirements were fulfilled by storing skill profiles from applicants, employees and jobs in a *profile database* and by enabling *matching* between skill profiles on top of the database.

The general components of ProPer are widely known, hence we restrict our further explanation of ProPer to the matching component, which is described in detail in Section 3.

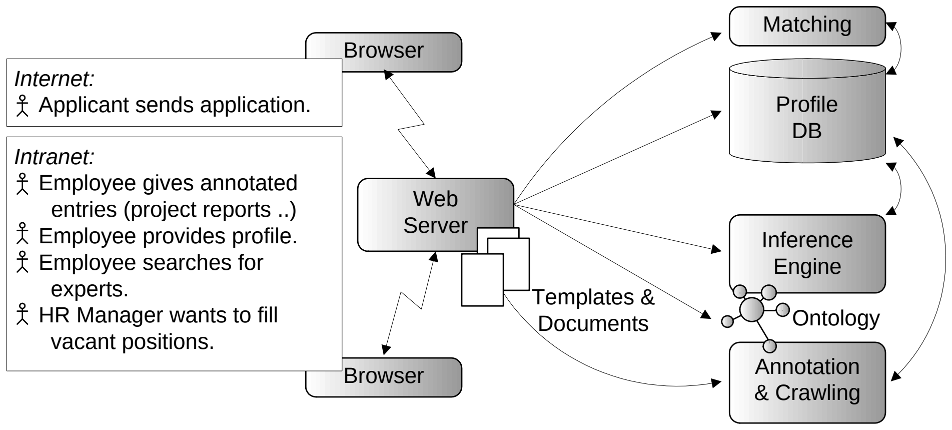

Browser
Internet:
Applicant sends application.
Intranet:
Employee gives annotated entries (project reports ..)
Employee provides profile.
Employee searches for experts.
HR Manager wants to fill vacant positions.
Web Server
Browser
Templates & Documents
Matching
Profile DB
Inference Engine
Ontology
Annotation & Crawling

Figure 2: Architecture of OntoProPer

2.2 OntoProPer: Reveal hidden skills

Our prototype OntoProPer extends the architecture of ProPer (*cf.* Figure 2). Maintaining profiles from employees manually is a time consuming task, therefore we use metadata that is structured according to an *ontology* and contained in *documents* (like *e.g.* project homepages and reports) to reveal additional information about skills from employees. The *crawled* metadata from various documents add value by constituting the foundation for inferences. Inferences are drawn exploiting metadata as well as conceptual structures (inheritance, associations) and rules from the ontology. Thus additional, automatically derived skill data supplements the skill data that is given explicitly in the database.

Metadata about documents is mainly generated from two sources: First, people work with templates. *E.g.* new project homepages are generated through templates that automatically annotate all given information (like employees working for the project, skills needed for that project etc.) while creating the homepage. Second, one may require that people annotate important documents. *E.g.* project reports documenting the ongoing project progress may contain valuable information about tools and technology used within that project.

The four principal use cases as well as the user interface from ProPer remain nearly unchanged: additionally an employee may here provide annotated entries such as project homepages and reports.

In Section 4 we explain more detailed how to reveal skills from annotated documents by crawling and inferencing.

3 A model for profiles

The model for profiles shown in this Section was implemented in the matching component on top of the profile database (*cf.* Section 2). First, we introduce the definition of profiles stored in our profile database (*cf.* Subsection 3.1). The following Subsection 3.2 explains the differences between non-compensatory and compensatory methods for matching. Then we give a definition for calculating match results and show how skills contained in profiles may be weighted according to their importance (*cf.* Subsection 3.3). Finally, we will provide an example how a compensatory matching between profiles is actually calculated (*cf.* Subsection 3.4).

3.1 Vector representation of profiles

Profiles consist of numerous values for different skills and may be represented as vectors. Our profile database P_{DB} (*cf.* Section 2) contains profiles from applicants (A) and employees (E) as well as requirement profiles (R) for jobs or project tasks:

$$\begin{aligned} &P_{DB} := A \cup E \cup R = \{p_i \mid i = 1 \dots m\} \\ &A \cap E = \emptyset \ \wedge \ A \cap R = \emptyset \ \wedge \ E \cap R = \emptyset \\ &p_i^T := (p_{i,1}, p_{i,2}, \dots, p_{i,n}) \end{aligned}$$

In our case study (especially in our example in Section 3.4) we used the integers “0” (no knowledge), “1” (beginner), “2” (intermediate) and “3” (expert) as skill values. If there is a need for more detailed measurement one may use an analogous, but more fine-grained scale.

3.2 Non-compensatory vs. compensatory methods

While looking for skillful employees or applicants you may have different requirements: you may look for a person that matches *exactly* or you may look for a person that matches *approximately* the given requirements. In the area of Multiple Attribute Decision Making (MADM) this is known as non-compensatory and compensatory methods [HK81].

Non-compensatory methods use cut-off vectors to make a binary decision. All skills from a persons profile have to equal or exceed their required skill — if only one skill is below the required ones, the profile is considered inadequate for the job. This method is particularly helpful for finding employees having exactly those skills that are required. This is mostly important for tasks requiring highly specialized skills such as maybe for biochemical gene analysis.

Compensatory methods allow to compensate "bad" skills (*viz.* skills below the requirements or skills that are simply missing) with "good" skills (*viz.* skills above the requirements). Instead of having a binary decision you may calculate a match result showing how well profiles match each other. This method gives you a good overview of potential persons for a job, so it is preferred *e.g.* for ranking incoming applicants. Also, it seems to be in general a good rule of thumb that you may compensate bad skills with good ones in the IT business — where you are happy these days to find any potentially skillful person at all. Therefore we preferred the compensatory method for matching in our application ProPer, which is shown more detailed in the following subsections.

3.3 Compensatory profile matching and weighting

In our scenario we have skill vectors of employees (in the following referred to as p_i) and requirement vectors (in the following referred to as p_j). To calculate compensatory match results M_C between two profiles as a percentage one multiplies both vectors (by using the dot product) and divides by the perfect match. If the profile of the employee is identical to the job requirements profile, the result indicates the good match by a value of 100%.

Simple skills may be of different importance for a given job and the importance may also vary for different jobs. *I.e.* the skill "Administration of Server X" is for a System Administrator more important than the skill "Programming in Y", but the reverse proposition may be the case for a Programmer. We take this into account by weighting each job profile with a corresponding weight matrix[1] (W) before actually calculating the dot product with a particular employee profile:

$$W := \begin{pmatrix} w_1 & 0 & .. & 0 \\ 0 & w_2 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & w_n \end{pmatrix}$$

[1]The weight matrix W is also stored in our profile database but kept separately from profiles.

$$M_C(p_i, p_j) := \frac{p_i^T \times (W * p_j)}{p_j^T \times (W * p_j)}; \; p_i, p_j \in P_{DB}$$

In our application we used the weights "1" (unimportant), "2" (important) and "3" (very important). Using "0" as a weight skips the weighted skill — so this skill is not relevant at all for the actual profile.

3.4 Example of matching process

Before we show in detail how to match profiles we give an example of an employee profile (*cf.* Figure 3) as it occurs in ProPer[2]. The profile consists of skill data which corresponds to domain-specific terms describing areas of expertise.

Let us consider an example scenario in which the Human Resource department tries to determine the most suitable person for a particular job. Assume that the employees provide data for the following array of skills (S):

$$S = (Windows\,2000,\ Java,\ JavaScript,\ Visual\,Basic)$$

For a new project the HR department is looking for a good programmer and creates a new job profile p_1 called "Programmer for project XYZ" from the given array of skills:

$$\left.\begin{array}{ll} Windows\,2000 & (beginner) \\ Java & (expert) \\ JavaScript & (intermediate) \\ Visual\,Basic & (beginner) \end{array}\right\} \; p_1 = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

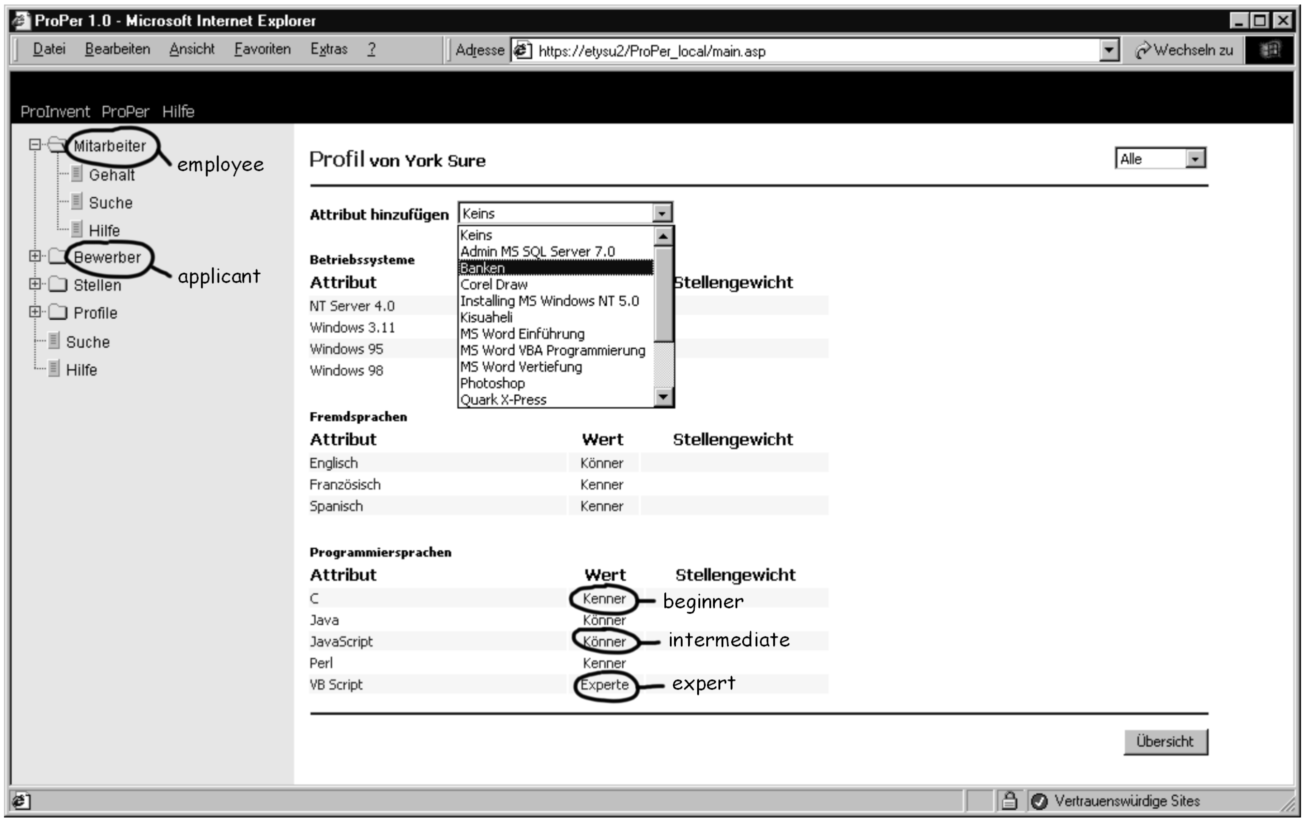
Thr HR department queries the profiles database and retrieves — beside others — the profile p_2 of Peter Perfect. For calculating the match result, the HR department now creates a weight matrix according to importance of the required skills:

$$\left.\begin{array}{ll} Windows\,2000 & (none) \\ Java & (intermediate) \\ JavaScript & (expert) \\ Visual\,Basic & (none) \end{array}\right\} \; p_2 = \begin{pmatrix} 0 \\ 2 \\ 3 \\ 0 \end{pmatrix}$$

$$\left.\begin{array}{ll} Windows\,2000 & (unimportant) \\ Java & (very\,important) \\ JavaScript & (important) \\ Visual\,Basic & (very\,important) \end{array}\right\} \; W$$

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

[2]The application is implemented in German.



Profil von York Sure

Attribut hinzufügen: Keins

Keins
Admin MS SQL Server 7.0
Banken
Corel Draw
Installing MS Windows NT 5.0
Kisuaheli
MS Word Einführung
MS Word VBA Programmierung
MS Word Vertiefung
Photoshop
Quark X-Press

Betriebssysteme

Attribut	Wert	Stellengewicht
NT Server 4.0		
Windows 3.11		
Windows 95		
Windows 98		

Fremdsprachen

Attribut	Wert	Stellengewicht
Englisch	Könner	
Französisch	Kenner	
Spanisch	Kenner	

Programmiersprachen

Attribut	Wert	Stellengewicht
C	Kenner	
Java	Könner	
JavaScript	Könner	
Perl	Kenner	
VB Script	Experte	

Figure 3: Example of a profile in ProPer

Now the match result is calculated:

$$M_C(p_2, p_1) = \frac{0*1*1+2*3*3+3*2*2+0*1*2}{1^2*1+3^2*3+2^2*2+1^2*2} = 79\%$$

Peter Perfect compensates very well his missing skills and his slightly too low skill "Java" with his strong capabilities in "JavaScript" which results in a match result of 79%.

4 Reveal Hidden Skills — Skill Inferencing

As already mentioned in the previous sections, the maintenance and the completion of data about employees' skills needs intelligent support. The idea of our implementation is that a lot of implicit knowledge is produced in the daily work, e.g. information contained in project reports. Our approach of skill inferencing is built on top of our previous work [DEFS99, SAD+00]. The system uses an ontology as conceptual and schematic backbone for structuring the domain, adding metadata to documents and for drawing inferences in order to derive or update the knowledge about individual's skills. The results of the skill inference process are integrated into the skill database, on which the skill matching functionality of ProPer works.

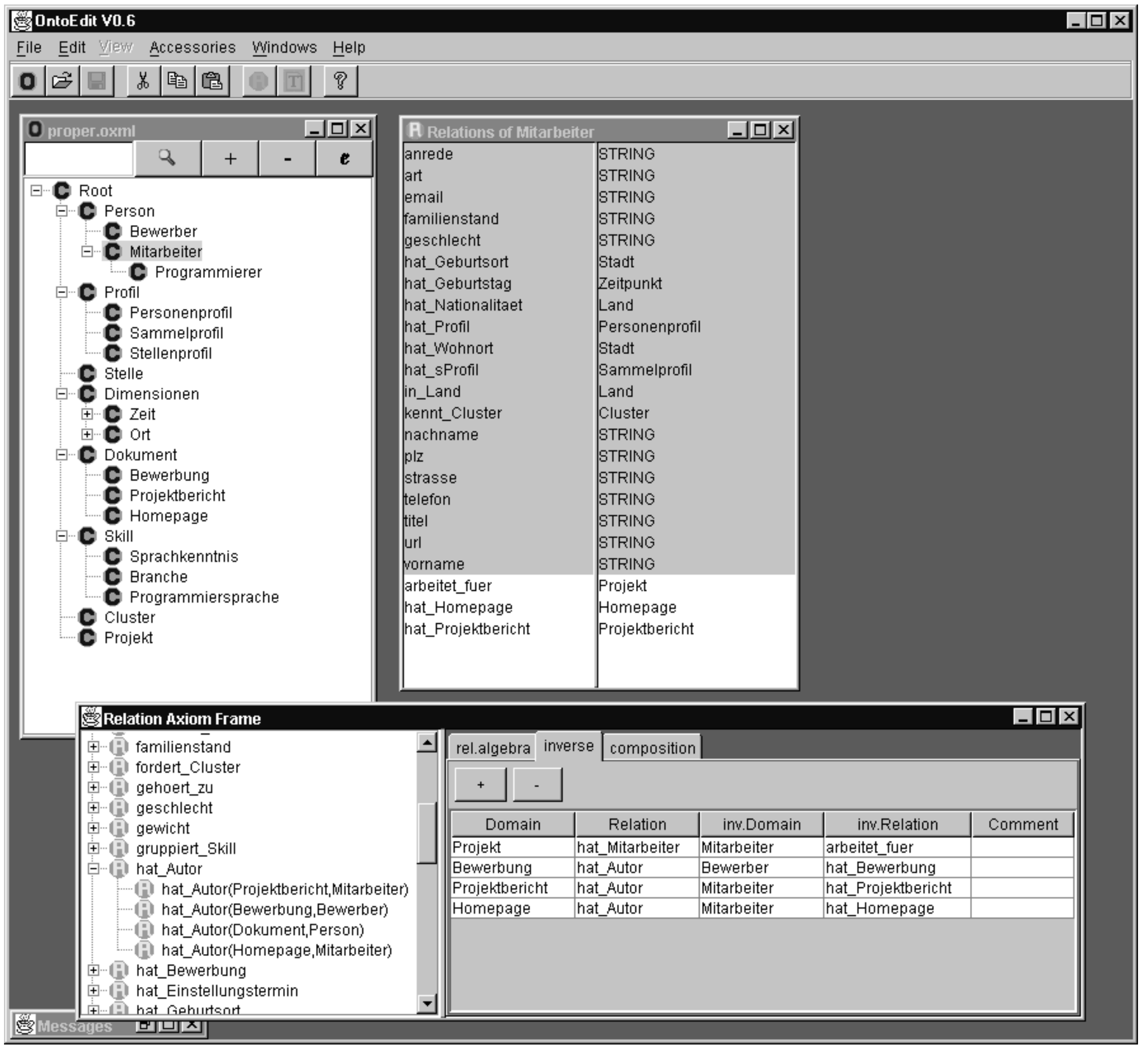
4.1 Ontology & Skills

An ontology for our skill management system has been manually derived based on the existing resources (*e.g.*, the skill database schemata, HR expert interviews) using standard knowledge acquisition techniques. The ontology engineering process has been supported by our Ontology Engineering Environment OntoEdit.[3] The role of an Ontology is to capture domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused and shared within communities or applications.

The skill management ontology for the OntoProPer system is partially depicted in Figure 4. It consists of *(i)* concepts, who are organized into a concept taxonomy, *(ii)* attributes of concepts and relations between concepts and *(iii)* rules allowing inferences. F-Logic[4] has been chosen to represent our skill management on-

[3]A comprehensive description of the ontology engineering system OntoEdit and the underlying methodology is given in [SM00].

[4]F-Logic is a frame-logic representation language conceived by [KLW95]. In the implementation by Angele and Decker that we use, F-Logic is a proper subset of first-order predicate logic. Concepts and relations are reified and, hence, may be treated as first-order objects over which quantification is possible. For efficient processing, F-Logic is translated into a datalog-style representation (*cf.* [LT84, Dec98]).



- Root
  - Person
    - Bewerber
    - Mitarbeiter
      - Programmierer
  - Profil
    - Personenprofil
    - Sammelprofil
    - Stellenprofil
  - Stelle
  - Dimensionen
    - Zeit
    - Ort
  - Dokument
    - Bewerbung
    - Projektbericht
    - Homepage
  - Skill
    - Sprachkenntnis
    - Branche
    - Programmiersprache
  - Cluster
  - Projekt

anrede	STRING
art	STRING
email	STRING
familienstand	STRING
geschlecht	STRING
hat_Geburtsort	Stadt
hat_Geburtstag	Zeitpunkt
hat_Nationalitaet	Land
hat_Profil	Personenprofil
hat_Wohnort	Stadt
hat_sProfil	Sammelprofil
in_Land	Land
kennt_Cluster	Cluster
nachname	STRING
plz	STRING
strasse	STRING
telefon	STRING
titel	STRING
url	STRING
vorname	STRING
arbeitet_fuer	Projekt
hat_Homepage	Homepage
hat_Projektbericht	Projektbericht

- familienstand
- fordert_Cluster
- gehoert_zu
- geschlecht
- gewicht
- gruppiert_Skill
- hat_Autor
  - hat_Autor(Projektbericht,Mitarbeiter)
  - hat_Autor(Bewerbung,Bewerber)
  - hat_Autor(Dokument,Person)
  - hat_Autor(Homepage,Mitarbeiter)
- hat_Bewerbung
- hat_Einstellungstermin
- hat_Geburtsort

Domain	Relation	inv.Domain	inv.Relation	Comment
Projekt	hat_Mitarbeiter	Mitarbeiter	arbeitet_fuer	
Bewerbung	hat_Autor	Bewerber	hat_Bewerbung	
Projektbericht	hat_Autor	Mitarbeiter	hat_Projektbericht	
Homepage	hat_Autor	Mitarbeiter	hat_Homepage	

Figure 4: Skill Management Ontology

tology. Relevant and important concepts of the human resource domain have been identified. In our scenario this included for instance the concepts Person, Profile, Document, Skill, Project, such as depicted in the right part of Figure 4. Relations have been used to describe the modeled concepts in further detail. For example the concept Employee ("Mitarbeiter") has 24 relations, where 21 are inherited from the more generic concept person and 3 are especially defined for the concept Employee.

This concept-relation-structure is further augmented by using rules. In the lower part of Figure 4, statements about relations are depicted. For example it is modeled, that the relation HAS_AUTHOR(ProjectReport,Employee) is inverse to the relation HAS_PROJECTREPORT(Employee, ProjectReport). A rule in F-Logic syntax describing a more complex example is given through

```
FORALL p,s,e e:Programmer[knows ->> s] <-
    e[worked_in ->> p]
    and p:Project
    and s:PrgLanguage[used_in ->> p].
```

This rule extends the data about programmers' programming skills. If a programmer worked for a project, in which a specific programming language has been used, than this programmer has at least some experience with the language. For each inferenced skill, we simply filled the profile database with the skill value "beginner". More fine-grained rules for determining the skill value may be used, if needed.

4.2 Adding Metadata to Documents

As already mentioned the ontology supports the generation of document metadata. Based on our previous work described in [EMSS00], project reports have been annotated using intranet templates and our annotation tool, ontological annotations have been added to the documents using an HTML extension, called HTML-A.

The annotation language HTML-A enriches HTML with primitives for tagging instances of concepts, for relating these instances, and for setting their properties, *i.e.* the ontology serves as a schema for semantic statements in these pages. For all these primitives the HTML anchor tag <A> has been extended with a special attribute `onto`. This decision implies that the original information sources remain nearly unchanged and still may provide semantically meaningful information. The semantic tags are embedded in the ordinary HTML text in such a way that standard browsers can process the HTML pages and, the knowledge crawler can extract the semantic annotations from them. Objects (instances of concepts) are uniquely identified by a URI, *i.e.* resources in the web are interpreted as surrogates for real objects like persons, organizations, and publications.

A small example annotation of a project report is given through:

```
<HTML>
  <BODY>
    <A onto="page:ProjectReport"/>
    <H1>
      Project Page
    </H1>
    <A onto="page[ProjectName=body]">
      Project VB
    </A>
    ...
    <A onto="page[usedMethod=body]">
      Visual Basic
    </A>
  </BODY>
</HTML>
```

In the schema `<A onto="`*O*`:`*C*`"></A>` of these expressions O represents the instance and C represents the concept. O can either be a global URI, a local part of a URI (that is expanded by the crawler to a global one), or one of the special keywords `page` or `body`. These special keywords represent resources relative to the current tag and the current web page, *e.g.* the keyword `page` represents the URI of the webpage of this statement. The keyword `body` refers to the content of the anchor tag. Thus, the actual information is rendered by a web browser and at the same time interpreted formally by the crawler. Including semantics in this way into HTML pages reduces redundancy and enhances maintainability, since changes in the prose part of the page are immediately reflected in the formal part, as well.

4.3 Example

Let us now continue our example from subsection 3.4 by applying our skill inferencing mechanisms. Peter Perfect's profile lacked two required skills: *Windows 2000* (which is *unimportant* according to the given weight matrix) and *Visual Basic* (which is *very important*). The HTML-A crawler has retrieved information about skills from employees contained in annotated documents such as project reports. Peter Perfect already worked for a "Project VB" where an application was implemented in *Visual Basic*. Applying one of the above described inference rules, the crawled facts "Peter Perfect worked for Project VB", "Visual Basic was used in Project VB", "Visual Basic is a programming skill" and "Peter Perfect is a programmer" result in "Peter Perfect knows Visual Basic". This information is added to his profile and a new match result is calculated:

$$M_C^{new}(p_2, p_1) = 88\%$$

The example shows that using our skill inference mechanisms supports maintenance and the completion of data about employees' skills. Using this intelligent support produces an added value, which is in our scenario measurable with better matching results.

5 Related Work

AIAI has been working on enterprise ontologies (*cf.* [UKMZ98]). Capability ontologies as part of enterprise ontologies describe capabilities that human beings or software agents may have. [SM99] restricts capabilities to those of human beings (also referred to as "skills") and presents well defined specifications for developing software systems based on capability ontologies. These systems help organizations to align skills of current and future employees with strategic business objectives. A prototype covering some of the given specifications has been implemented, but lacks *e.g.* of recruitment profiles.

Several organizations have developed systems who support the finding of hidden skill knowledge by using Artificial Intelligence (AI). A survey of existing AI-based systems is given in [BF00]: CONNEX is a people-finder developed by Hewlett-Packard. The company wanted to keep track of the business and technical skills of their employees. The goal was to build a guide to human knowledge by connecting a network of experts. They created profiles containing knowledge and skills as well as affiliations, who are updated by the employees themselves. Another system is SPUD, which was implemented by Microsoft. They have developed a structure of competency levels, defined competencies required for a particular job, included supervisors' rating of an employee's performance, implemented the competencies in an online system and linked this model to an automated providing of teaching courses. In difference to our approach, both systems do not rely on semi-structured documents nor do they make use of the inferencing capabilities of an inference engine to seek hidden structures. Furthermore, no matching between job profiles and people profiles is supported by CONNEX. The SAGE system is a knowledge management system for searching experts in Florida funded by NASA. In contrast to the above mentioned systems, SAGE relies on a keyword-based search engine instead of self-assessed profiles. Abstracts from researchers are searched for keywords — relying on the fact that this researcher has knowledge about a topic if he publishes about it. The system is designed to be as people-independent as possible and integrates numerous databases. Compared to our approach, the SAGE system does not provide intelligent inferencing of skills, nor does it integrate self-assessed profiles. Our strength is the integration of both methods to find knowledge combining skill matching and skill inferencing.

The Ontobroker system has been used for other approaches like *e.g.* a proactive inferencing agent for desk support [SS00] exploiting metadata and ontologies in a project planning scenario or knowledge management through ontologies [BFG98] making knowledge assets intelligently accessible to people in organizations. However, only OntoProPer combines the soft matching with the inferencing capabilities.

6 Conclusion

We have presented two systems for IT support of skill management. The first, **ProPer** employs a "soft matching component" in order to match people to position. The second, **OntoProPer** extends the first in order to provide more comprehensive knowledge about individuals' skills using background knowledge from an ontology and secondary information, such as from project documents. While ProPer is currently on duty and has proved very successful in a large IT consulting company, OntoProPer still has to be applied to the real world and evaluated.

Though we have mostly worked towards finding the right people for a particular job in this paper, we want to mention that a comprehensive, well-maintained skill database with intelligent techniques may give a leverage far beyond what we have explored in ProPer and OntoProPer.

For instance, given a comprehensive skill database HRM may analyze (*e.g.* with OLAP or with data mining techniques) whether the company suffers from a shortage of knowledge in a particular area or whether the company's expertise in a particular area is all built on one or two key persons and, hence, is lost when these persons leave the company. For the future, we plan to integrate data mining techniques that harvests new knowledge from skill databases.

Acknowledgements

The case study has been carried out together with Prodacta AG, Ettlingen, Germany. Part of the research described in this paper has been financially supported by EU in the IST project OnToKnowledge (EU-IST-1999-10132).

References

[BF00] Irma Becerra-Fernandez. The role of artificial intelligence technologies in the implementation of people-finder knowledge management systems. In Staab and O'Leary [SO00].

[BFG98] V.R. Benjamins, D. Fensel, and A. Gomez. Knowledge management through ontologies. In *PAKM 98 Practical Aspects of Knowledge Management — Proceedings of the Second International Conference*, 1998.

[Dec98] S. Decker. On domain-specific declarative knowledge representation and database languages. In A. Borgida, V. Chaudri, and M. Staudt, editors, *KRDB-98 — Proceedings of the 5th Workshop Knowledge Representation meets DataBases, Seattle, WA, 31-May-1998*, 1998.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.

[EMSS00] M. Erdmann, A. Maedche, H.-P. Schnurr, and Steffen Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *P. Buitelaar & K. Hasida (eds). Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, Luxembourg, August 2000.

[HK81] C.-L. Hwang and Y. Kwangsun. Multiple attribute decision making, methods and applications. In *Lecture Notes in Economics and Mathematical Systems*, number 186. Springer-Verlag, Berlin, Heidelberg, New York, 1981.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[LT84] J. W. Lloyd and R. W. Topor. Making Prolog more expressive. *Journal of Logic Programming*, 1(3), 1984.

[SAD+00] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, R. Studer, and Y. Sure. Semantic Community Web Portals. In *Proceedings of the 9th World Wide Web Conference (WWW-9), Amsterdam, Netherlands*, 2000.

[SM99] J. Stader and A. Macintosh. Capability modelling and knowledge management. In *Applications and Innovations in Expert Systems VII, Proceedings of ES 99 the 19th International Conference of the BCS Specialist Group on Knowledge-Based Systems and Applied Artificial Intelligence*, pages 33–50, Cambridge, December 1999. Springer-Verlag.

[SM00] S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *Proceedings of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods.* IOS Press, Amsterdam, 2000.

[SO00] S. Staab and D. O'Leary, editors. *Bringing Knowledge to Business Processes. Workshop in the AAAI Spring Symposium Series. Stanford, March 20-22, 2000*, Menlo Park, CA, 2000. AAAI.

[SS00] H.-P. Schnurr and S. Staab. A proactive inferencing agent for desk support. In Staab and O'Leary [SO00].

[UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13, 1998.