Applied Intelligence manuscript No.
(will be inserted by the editor)

Embedding Semantics in Human Resources Management Automation via SQL

Eufemia Tinelli · Simona Colucci · Francesco M. Donini · Eugenio Di Sciascio · Silvia Giannini

Received: date / Accepted: date

Abstract Among enterprise business processes, those related to HR management are characterized by conflicting issues: on the one hand, the peculiarities of intellectual capital ask for rather expressive representation languages to convey as many facets as possible; on the other hand, such processes deal with huge amounts of resources to be managed.

For handling HR management tasks, our approach combines the representation power of a logical language with the information processing efficiency of a DBMS. It has been implemented in a fully functioning platform, `I.M.P.A.K.T.`, that we present here highlighting its peculiarities for three relevant business processes: skill matching, task/team composition and company core competence identification.

Keywords Skill Management · Knowledge Compilation · Description Logics · RDBMS · SQL

1 Introduction

The so-called network economy changed the way of conceiving knowledge management in organizations. Enterprise borders go far beyond its physical location

E. Tinelli
Politecnico di Bari, DEI, Via Orabona 4, 70125 Bari, Italy
E-mail: eufemia.tinelli@poliba.it

S. Colucci
Politecnico di Bari, DEI, Via Orabona 4, 70125 Bari, Italy
E-mail: simona.colucci@poliba.it

F. M. Donini
Università della Tuscia, DISUCOM, Via Santa Maria in Gradi 4, 01100 Viterbo, Italy
E-mail: donini@unitus.it

E. Di Sciascio
Politecnico di Bari, DEI, Via Orabona 4, 70125 Bari, Italy
E-mail: eugenio.disciascio@poliba.it

S. Giannini
Politecnico di Bari, DEI, Via Orabona 4, 70125 Bari, Italy
E-mail: silvia.giannini@poliba.it

and more and more knowledge-intensive resources become available across the network. Such a shared environment asks for unambiguous interpretation of information sources and makes knowledge-based technologies a key success factor. In particular, business processes automation can take a crucial chance by technologies supporting knowledge representation and management [21].

Among other business activities, human resources management can get a significant boost from the adoption of knowledge-based technologies aimed at business automation. In fact, intellectual capital is an asset with peculiarities: it is intangible, its description is subjective, and the way to describe such an asset is a key choice for its successful exploitation. Logic-based representation perfectly fits such peculiarities, and the representation language—if properly chosen—could support human resources management automation through suitable reasoning services.

Even though such a flexibility and informative potential is paid in terms of computational cost of reasoning, semantic-based enterprise systems may represent a good candidate to substitute traditionally employed solutions, mostly based on Relational Data Base Management Systems (RDBMSs). Despite recent emerging systems in the data storage field, such as NOSQL[1] technologies, we focus on RDBMS, as structured data model engine, and SQL, as declarative language for data manipulation and retrieval, since they still represent the most popular and well-known tools in enterprise scenario. Nevertheless, RDBMSs lack of a semantic characterization of resource descriptions and thus allow only for syntax-based information management, which is too restrictive for a domain as knowledge-intensive as human resources. Our main goal in this paper is aiming at filling such a gap.

For example, available e-recruitment tools[2] generally store information about employment, personal data, certifications and competence of candidates by exploiting RDBMSs with customized and structured templates. Then, some information is extracted through relational query languages—SQL or some variants. Now, an RDBMS is surely suitable for efficient storage and retrieval of data, yet SQL is usually not flexible enough to support a discovery process as complex as recruitment. Take, for instance, the problem of selecting the most appropriate candidate for some job; this is a process involving several preferences, some along orthogonal dimensions—*e.g.*, acquired skills vs. geographical location—while others somehow related—*e.g.*, learning gaps and expected salary. By means of SQL standard operators such as *aggregate functions*, `group by` and `order by` clauses, the user is able to retrieve the *best* tuples (*i.e.*, the best candidates for a specific task) according to her sorting criteria; yet such tuples still need a human-based post-processing phase, where incompatible or unsatisfied preferences (not directly representable by standard SQL) are evaluated and traded off.

More generally, classical DB-based techniques show their limits in managing complex domains: (i) search processes can be very time-consuming but often unsatisfactory because underlying frameworks basically rely only on keyword-based approaches; (ii) a user can express only mandatory requirements (it is not possible to select features according to some negotiable constraints). On the one hand, such systems usually do not return arranged outcomes (a priori excluding results summarily deemed as not relevant) and, above all, they do not provide any re-

[1] *e.g.*, `http://nosql-database.org/`

[2] *e.g.*, `http://www.monster.com/`, `http://www.careerbuilder.com`

sults explanation. On the other hand, heavy computational capabilities prevent a widespread usage of semantic approaches, and as soon as real data sets have to be faced in common applications, response times become often unacceptable.

We propose to reconcile such a dichotomy by combining the richness in informative content, typical of semantic-based approaches, with efficient data management and scalability, characterizing RDBMS-based ones. To draw an analogy with Programming Languages, our approach allows a user to express information in a high-level knowledge representation language, and then it "compiles" and stores it away as tuples in a Relational Data Base (RDB). Then, when the user expresses queries regarding such data—still in the high-level representation language—they are compiled to SQL queries that match the previous tuples. In this way, our system retains both the expressiveness of a knowledge representation language and the efficiency of an RDBMS.

Our approach follows the so-called Knowledge Compilation scheme [10]. Knowledge Compilation has been employed to make computationally easier to reason over the information contained in a Knowledge Base (KB), by splitting the reasoning process in two phases: (i) the KB is pre-processed, thus parsing it in a proper data structure (*off-line reasoning*); (ii) the query is answered exploiting the structure coming out of the first phase (*on-line reasoning*). To this purpose, the proposed skill engine leverages KB pre-processing to reduce on-line reasoning overhead.

The main contributions of this article may be summarized as follows:

– we present a novel skill management approach, which adopts a Knowledge Compilation scheme over a KB formalized in Description Logics (DLs) [2], that makes the proposed approach different from our previous work [17]. At the same time, it preserves computational efficiency, which is quite low in fully logic-based approaches.
– we present `I.M.P.A.K.T.` (**I**nformation **M**anagement and **P**rocessing with the **A**id of **K**nowledge-based **T**echnologies)[3], as an innovative system implementing the approach above. `I.M.P.A.K.T.` is an integrated system for human resources management by supporting the execution of three relevant business processes: i) retrieval of ranked referral lists, ii) working team composition, iii) automated extraction of strategical enterprise competence[4]. The performance of such business processes exploits an inference engine solving non-standard reasoning services, by means of a flexible query strategy in standard SQL. This strategy results from the design of the on-line reasoning component of our Knowledge Compilation approach.

For a matter of readability, we here report only on the modeling approach allowing to translate the KB into the reference relational database. We do not delve into details about the design of SQL queries solving the reasoning services at the basis of provided HRM services, only reporting on the design of the off-line reasoning phase of the Knowledge Compilation approach.

The rest of the paper is organized as follows. The next section reports on relevant professional tools and research proposals related to our work. Section

[3] An embryonic `I.M.P.A.K.T.` version, including only retrieval of ranked referral lists of candidates, has been presented in [53].

[4] A short introduction of the three services in `I.M.P.A.K.T.` has been given in some previous works [55, 54, 18].

3 introduces the off-line reasoning component of the proposed knowledge compilation approach (the reader not familiar with Description Logics may need to read through Appendix A before). In Section 4 we first show efficacy of the proposed non-standard reasoning services through a comprehensive example and then present a numerical experimental evaluation of the system. Conclusions close the paper.

2 Related Work

In order to motivate the proposal of `I.M.P.A.K.T.`, we compare it with relevant related work. In particular, to show how `I.M.P.A.K.T.` combines advantages of both DB-based and logic-based approaches to automated recruitment, we report on related work pertaining to both categories, proposed in professional and academic products.

2.1 Professional solutions for Recruitment

Most professional tools for talent management[5] and e-recruitment are enterprise suites supporting human resources management, including solutions that, even though improving the recruitment process by means of innovative media and tools, do not introduce a significant novelty charge. Available solutions in fact exploit databases to store candidate personal and employment information, and perform recruitment on the basis of exact match, in absence of a logic-based structure.

On the contrary, the inadequacy of exact match of resumes in job search has been recognized several times over the last years, and motivates the proposal of some professional solutions.

To the best of our knowledge, one of the first logic-based tools for recruitment and referral process is RESUMIX[6], which has been employed at US Army and Navy Department till 2012, and is still at the basis of NASA recruitment process. As far as we can see, due to the privacy restrictions required by its employment in military institutions, RESUMIX is a staffing tool using techniques mining the context of words to overcome limits of keyword-based search. It is semi-automated (the last resume review is left to humans) and asks applicants for writing resumes according to a specific format (guided by a web interface), and apply for open positions. RESUMIX distinguishes skills in *required* and *desired* ones in the description of open positions: only required skills must be matched by the retrieved candidate, while *desired* ones are only used to rank candidates. No explanation is returned for the proposed solutions.

Sovren[7] offers different valuable solutions to semantic-based matching of profiles. Notably, the *Sovren Resume Parser* deals with the problem of converting resumes from several text formats to the standard HR-XML schema[8]. This fully

[5] `http://www.attract-hr.com/cm/about,`
`https://www.oracle.com/it/applications/human-capital-management/talent-management/acquisition/index.html`

[6] `http://nasajobs.nasa.gov/NASAStars/about_NASA_STARS/what_is_resumix.htm`

[7] `http://www.sovren.com`

[8] `http://www.hropenstandards.org/`

distinguishes Sovren from other providers, which ask candidates to model CVs in a specific format. The fully-automated tool *Sovren Semantic Matching Engine* (SSME) is able to match resumes (in HR-XML) to job offers properly formatted thanks to the *Sovren Job Order Parser* embedded in SSME (the parser translates also job offers in plain text). SSME allows for weighting search criteria (the granularity level of the weighting mechanism is not provided) and returns a ranked list of candidates, which seems to exclude false-positives (it is not clear if the process produces false-negatives, because the matching strategy is not revealed in full details). The full automation makes the matching process completely hidden to the recruiter, who does not receive any explanation about retrieved results. SSME provides also a service for finding candidates (jobs) who are similar to other candidates (jobs, respectively), called candidate (job) clone.

Around 2013, *Monster.com*$^{(R)}$, the leading Web job-matching engine, introduced the service *Monster Power Resume Search*$^{(TM)}$[9] to improve the relevance of candidates recruitment over Monster database. The product relies on the semantic 6*Sense*$^{(TM)}$ search technology, patented by Monster Worldwide, Inc. itself. Thanks to such a search technology, the novel engine is able to mine the content and the context of each term in the request. A guided interface supports recruiters in specifying all the features of the request, together with the related levels and years of experience. *Monster Power Resume Search*$^{(TM)}$ retrieves a ranked list of candidates, whose top skills are shown in the results interface and easy to compare. The service apparently lacks of a mechanism for weighting job requirements and of an explanation of ranked results.

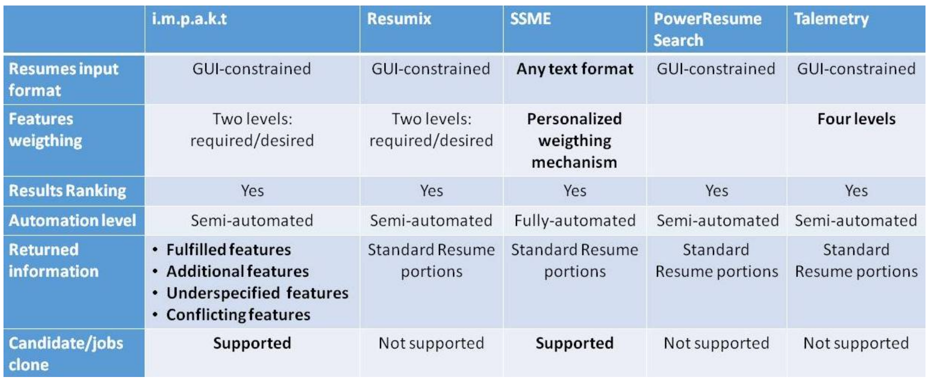
Talemetry$^{(R)}$ Source & CRM offers an easily searchable interface over a single database combining internal and external candidate sources. The interface allows for specifying the relevance of each feature in the request. The match process returns a ranked list of candidates, together with a brief description of their main skills and job experiences. The matching criterion adopts both ontological categorization and semantic analysis, but is not revealed in details. No explanation of results is provided.

In Figure 1 we classify the four tools introduced above and `I.M.P.A.K.T.`, according the six dimensions emerged in their analysis: input format of resumes, weighting mechanism of search features, ranking of results, level of automation, information returned with ranked results, and candidate/job clone service.

We immediately remark that none of the analyzed tools deals with team composition and core competence extraction, which are therefore outside the analysis, but still represent services which `I.M.P.A.K.T.` has more than the others. We also remark that such services ground on the same ontology, E-R model, profile storing and reasoning primitives in SQL. This makes `I.M.P.A.K.T.` an integrated system for human resources management and not only a recruitment service.

As for skill matching, we report in bold font the distinguishing peculiarities of each tool. Notably, all the tools return a ranked list of candidates (see Row 3 in Figure 1) but most of them show only a standard portion of the resume beside each returned candidate. Only `I.M.P.A.K.T.` returns explanation of results, in the form of fulfilled, additional, underspecified and conflicting features (see Section 4.1 for details on such features).

[9] `http://hiring.monster.com/recruitment/Resume-Search-Database.aspx`



	i.m.p.a.k.t	Resumix	SSME	PowerResume Search	Talemetry
Resumes input format	GUI-constrained	GUI-constrained	**Any text format**	GUI-constrained	GUI-constrained
Features weigthing	Two levels: required/desired	Two levels: required/desired	**Personalized weigthing mechanism**		**Four levels**
Results Ranking	Yes	Yes	Yes	Yes	Yes
Automation level	Semi-automated	Semi-automated	Fully-automated	Semi-automated	Semi-automated
Returned information	• **Fulfilled features** • **Additional features** • **Underspecified features** • **Conflicting features**	Standard Resume portions	Standard Resume portions	Standard Resume portions	Standard Resume portions
Candidate/jobs clone	**Supported**	Not supported	**Supported**	Not supported	Not supported

Fig. 1 Comparison of semantic-based recruitment systems (distinguishing peculiarities are reported in bold font)

Even more importantly, I.M.P.A.K.T. is the only one *using* explanation to rank results. In particular, it extends the search to non-exact matches in an Open World Assumption, going further a merely subsumption-based match. The match strategy of the other tools is not revealed in full details, and, for some of them, it is not clear if an ontology is employed to mine the context and the content of keywords. In the worst case, no ontological structure is used and neither subsumption between candidate and job description holds. In the best case, an ontology is used but the tool does not go further a subsumption-based match. This may cause an unaffordable number of false negatives, which, in turn, reduces the tool utility.

As for the first two rows in Figure 1, a service for CV translation from plain text and an improved mechanism for weighting features are under investigation. Their implementation will be part of our future work.

We did not report in bold font the automation level of SSME, because the adoption of full automation in skill matching is controversial. Many systems (see RESUMIX, as an example) leave the final choice up to humans and claim semi-automation as a noteworthy possibility.

2.2 Review of related literature

The knowledge compilation process underlying I.M.P.A.K.T. makes it able to share some features with DB-based approaches to skill management and some others with logic-based ones. Therefore, we first report on the DB-based solutions to resource matching related to ours. Then, we compare the framework underlying I.M.P.A.K.T. with logic-based research proposals.

A feature not new in database querying and provided by our system is the possibility to take into account both user preferences and strict requirements in the overall retrieval procedure. In database querying, infact, the need to provide users with a set of answers taking into account the presence of preferences has been recognized. Two competing approaches emerged so far, thanks to the work by Chomicki [12], though not specifically applied to skill management. The first one—defined as *quantitative*—models preferences by means of utility functions [9,44], whereas the second *qualitative* one uses logical formulas [12,32]. In other words, in the qualitative approach, the preferences between the tuples in the result set of a query are directly specified, usually by using binary preference relations. In the quantitative approach, preferences are instead indirectly expressed by using scoring functions that associate a numeric score to each tuple in the result set. Then, a tuple t_1 is preferred to a tuple t_2 iff the score of t_1 is higher than the score of t_2.

Differently from both the above introduced approaches, we do not define a specific query language for preference management. In fact, we are interested in providing a powerful tool exploiting only standard SQL, which is both well-known in enterprise environments and commonly employed in resource management frameworks implementation. Moreover, in preference-based approaches, tuples represent classical structured information. On the other hand, several works have been presented [40, 30] dealing with non-structured information, *i.e.*, textual descriptions representing informative needs. On the contrary, we manage semantic-based information, properly modelled and stored in tuples of a database, such that reasoning tasks can be performed over them.

Several approaches have been presented, where databases allow users and applications to access both ontologies and other structured data in a seamless way.

An approach aimed at classification in SQL databases of ontologies formalized in $\mathcal{ELH}$ has been presented in [19]. Furthermore, [56] shows a translation of $\mathcal{ALN}$-concepts into pairs (database, query) such that to decide Subsumption $C \sqsubseteq D$ one evaluates the query from D over the database from C. In this way, subsumption between the original concepts can be decided as query answering over a database. These approaches are devoted only to knowledge representation, since they solve subsumption, and not to non-standard reasoning services like Concept Abduction and Contraction (actually such services do not even have a formalization in $\mathcal{ELH}$). Another possible optimization is to cache the classification hierarchy in the database and to provide tables maintaining all the subsumption relationships between primitive concepts. This happens for example in *Instance Store* (***iS***) [5], a system for reasoning over OWL KBs specifically adopted in biomedical-informatics domains. ***iS*** is also able—by means of a hybrid reasoner/database approach—to reply to instance retrieval queries w.r.t. an ontology, given a set of axioms asserting class-instance relationships. A comparison between ***iS*** and our approach shows that the former reduces instance retrieval to pure TBox reasoning and it is able to return only exact matches, whilst we use an enriched relational schema storing only the ABox (*i.e.*, facts) in order to provide a logic-based ranked list of results.

Das *et al* [13] developed a system that stores OWL-Lite and OWL-DL ontologies in Oracle RDBMSs, and provides a set of SQL operators for ontology-based matching. *Jena 2 Ontology Stores* [63], *Sesame* [8] and *Oracle RDF Store* use a three columns relational table $\langle Subject, Property, Object \rangle$ to memorize RDF triples whereas other ontology storage systems, such as *DLDB* [45] and *Sesame on PostgreSQL* [8], adopt binary tables. The most popular and recent OWL storage is OWLIM [33]. It is a Sesame plug-in able to add a robust support for the semantics of RDFS, OWL Horst and OWL2 RL [1]. Another system exploiting DBMS techniques to deal with reasoning tasks (*i.e.*, subsumption check and instance retrieval) is OWLDB[10]. It defines a methodology for translating $\mathcal{SHOIN}$ inference rules into relational database queries, with benefits in scalability and performance.

[10] `http://owldb.sourceforge.net/`

SHER [20] is a highly-scalable OWL reasoner performing both membership and conjunctive query answering over large relational datasets using ontologies modeled in a subset of OWL-DL without nominals. It relies on an indexing technique summarizing database instances into a compact representation used for reasoning. It works by selectively uncompressing portions of the summarized representation relevant for the query, in a process called refinement. Internally, SHER uses Pellet to reason over the summarized data and obtain justifications for data inconsistency. SHER allows for getting fast query answering, but does not provide a ranked list of results. PelletDB[11] provides an OWL 2 reasoning system specifically built for enterprise semantic applications. It combines Pellet's OWL capabilities and scalable native reasoning of Oracle Database 11g so ensuring performance improvements w.r.t. to the use of such technologies separately. Differently from the previous approaches, one of the most widespread DL-reasoners, *i.e.*, KAON2[12], does not implement the tableaux calculus, but it reduces a $\mathcal{SHIQ}(\mathcal{D})$ knowledge base to a disjunctive datalog program. An inference engine for answering conjunctive queries has been so developed applying well-known deductive database techniques.

All the cited systems, although using languages more expressive than `I.M.P.A.K.T.`, are only able to return either exact matches (*i.e.*, instance retrieval) or general query answering. Instead, we use an enriched relational schema to deal with several non-standard inferences, to provide effective value-added services. This peculiarity is due to the logic-based nature of `I.M.P.A.K.T.`: the knowledge compilation process is able to embed an informative content otherwise hardly extractable with classical DB-based approaches.

The benefits of semantic technologies in enterprise knowledge management have been pointed out since 1990s and continue to be widely recognized—even in very recent works [28,31,60]. In particular, several approaches propose ontologies as knowledge repositories, to provide a common vocabulary and to model general Knowledge Management procedures [27] and tools [11].

A relevant issue arises in using ontologies once they have been built, *i.e.*, reasoners and reasoning services must be designed and implemented to take full advantage from the effort placed in structuring an ontology. Also, an intense use of inference services is required [42,43] to justify the computational cost of their performance.

On the contrary, although several semantic facilitators have been proposed in literature for several scenarios and techniques [48,50,57,64] often they do not fully leverage the ontological structure, limiting their inferences to simple subsumption matching. The work by Colucci *et al* [14] gathers several semantic-based approaches to retrieval, based on specifically devised non-standard services in Description Logics. Thanks to them, the matchmaking process may be extended w.r.t. exact match. An approach seemingly similar in overcoming exact matches has also been proposed [6]. It extends the one for measuring similarities in ontologies by Ehrig *et al* [22] for combining the advantages of similarity-based search with those of ontology-based systems. Nevertheless, such an approach does not allow to provide explanation of results in case of non-exact matches.

The approach to skill matching by Hefke *et al* [26] may appear close to the above-mentioned one [14]. It is based on a technique presented by Stojanivic *et al*

[11] `http://clarkparsia.com/pellet/`

[12] `http://kaon2.semanticweb.org/`

[49] for ranking query results. The relevance of query outcomes is computed taking into account the structure of the underlying domain (using the Knowledge Base content) and the inference process characteristics. The ranking, though providing a useful support to the choice between the returned profiles, only classifies answers w.r.t. queries formalized with a well-defined structure. Such an approach lacks expressiveness in the querying process. Moreover, the Open World Assumption is not made, because only answers explicitly providing characteristics required by the query are considered. Finally, no explanations are provided about the rationale in case of absence of match.

Mochol *et al* [38] address the role of semantic techniques in improving the accuracy of job search. The authors propose to describe candidates and jobs according to so-called thematic clusters, which represent specific portions of CVs and job postings. The similarity of each pair of corresponding clusters is then computed on an ontological basis and contributes to the calculus of total similarity between the candidate and the job. Although sustaining semantic-based search, the authors raise important outstanding problems, like the impossibility to express the duration of a particular experience (*e.g.*, 3 years experience in Java programming) and the lost of job applications which do not fit 100% to the defined requirements but are still acceptable for the employer (*e.g.*, 3 years instead of 5 years industry experience). To address such needs, the authors propose a query relaxation approach supporting the raking of results and reducing the number of false negatives. Notably, `I.M.P.A.K.T.` overcomes both the problems raised by Mochol *et al*: it supports the specification of years of experience related to each profile features and performs an extended matchmaking, which not only returns imperfect matches but also explains the reason for such an imperfection.

Fazel *et al* [23] propose the formalization of job announcements and resumes in a given template in DLs. The template distinguishes skills in requirements and desires, and performs match only on requirements, using desires only for ranking matching candidates. `I.M.P.A.K.T.` allows for such a distinction, but returns also candidates slightly conflicting the job posting, together with the reason for the mismatch. Moreover, the approach by Fazel *et al* does not investigate and motivate the adopted DL and does not exploit its reasoning services. The match is based on measures for semantic similarity and seems not to take much advantage from the proposed formalization in DLs. In other words, it is not clear how the logic-based characterization of employed terms affects the ranking process. On the contrary, `I.M.P.A.K.T.` performs the match (and all the other supported processes) by implementing well-investigated inferences in DL, in a feasible and unambiguous fashion. The choice of the DL is therefore motivated by the feasibility needs and made clear to the reader.

Very recently, an approach for reasoning under uncertainty and vagueness, adopting a fuzzy Description Logics, has been applied to the job market [29]. The fuzzy extension copes with the need to manage vague and uncertain information, like the salary, the age and the daily working hours, so to give flexibility to the query process. In particular, Dempster-Shafer Framework and Dempster's rule of Combination are used to reach an agreement for matchmaking between a job seeker and a recruiter. This theory is suited for preference fusion situations. The matchmaking process implements a set of Semantic Web Rule Language (SWRL) [41] rules, executed to define Offer and Seeker constraint factors. As for the approach by Fazel *et al* [23], the underlying ontology models concepts in a two-fold fashion: the same entity (*e.g.*, Degree) has one conceptualization as requirement (*e.g.*, RequiredDegree) and one other as preference (*e.g.*, PreferredDegree). Duplex conceptualizations are also set to differentiate offered features (*e.g.*, SeekerPreferences) from searched ones (*e.g.*, SkillPreferences). This is due to the choice of managing the *JobSeeker* and the *JobOffer* in different templates, to be logically compared by applying above mentioned rules. Thanks to knowledge compilation, `I.M.P.A.K.T.` is able to return and explain also *almost*-exact matches, only through standard SQL queries translating well-defined reasoning services. Notably, `I.M.P.A.K.T.` ontology allows for modeling both resumes and job offers with the same template, without splitting the same entity in different concepts.

The design of an adaptive approach based on automatic matching learning has been proposed for the mediation between recruiters and candidates [36]. This approach should be able to calculate the transformation cost of a given profile into a requested job offer, so that profiles with higher transformation cost should rank worse than those with lower cost. The approach is just sketched at design level, but the authors outline the importance of: i) using knowledge bases to avoid multiple conceptualizations of the same entity; ii) having a weighting mechanism for the formalization of requests; iii) augmenting the number of matches through a mediation process. The design of `I.M.P.A.K.T.` shares this vision, so that these three issues are addressed by the system.

The research by Rácz *et al* [46] extends the results of an exact subsumption-based matching, by taking into consideration similarities between different skills that are not related by the subsumption relation. In particular, the approach computes the probability of having some required skills on the basis of the explicitly hold ones. Then, a probabilistic matching based on the maximum entropy model is analyzed: the match value of a job offer and an application is the result of a probabilistic query. The number of false positives deriving from this approach is not investigated and no explanations is given for retrieved matches.

From the literature reviewed so far, it emerges that:

- all the approaches propose a semantic-based representation of human resources domain;
- logic-based matching (possibly based on subsumption) of candidates and offers is the ideal one, but often returns a too small number of candidates;
- a mechanism for weighting the importance of features required/offered is provided in each approach;
- each approach adopts a different approximation strategy to gain matching chances.

In the following, we show how `I.M.P.A.K.T.` follows the principles itemized above, but presents several distinguishing features w.r.t. existing approaches. In particular, it is a semantic-based system, which incorporates in a unique modeling framework knowledge management solutions related to different organizational needs, rarely—if ever—faced from an integrated perspective. To the best of our knowledge, it is among the few systems fully exploiting the knowledge representation effort spent in modeling informative resources. `I.M.P.A.K.T.` proposes, in fact, significant explained solutions double-tied with some Description Logics inferences, specifically developed for explanation purpose. This ensures the originality of system functionalities.

In all the revised approaches, the approximation strategy is not made clear to the user (either the recruiter or the candidate), while `I.M.P.A.K.T.` makes all the information used for ranking candidates explicit and available for possible mediation processes.

Moreover, we stress that `I.M.P.A.K.T.` completely differs from the system we presented in our past research [17], which: i) is fully logic-based and therefore has retrieval times not feasible in a real scenario; ii) lacks of a mechanism for distinguishing requirements and preferences in the query process; iii) grounds on a different approximation strategy and then returns less matches, with the explanation of only missing and conflicting features; iv) does not offer the possibility to mediate match starting form the provided results.

On the contrary, the modeling effort spent in knowledge compilation makes `I.M.P.A.K.T.` significantly original. It is in fact able to perform in feasible times tasks related to knowledge representation and reasoning by only reverting to a DBMS.

3 Modeling the Knowledge Base

In the following, we assume the reader familiar with basic DLs formalism and reasoning services. Nevertheless, a short introduction to DLs may be found in Appendix A for interested readers.

`I.M.P.A.K.T.` framework aims at properly storing profiles (*i.e.*, structured CVs, see Definition 2 for its formalization), to efficiently perform reasoning services only via SQL standard queries over a relational database fully mapping the KB. Provided services are related to three categories of human resources management business processes and rely on a unified framework for knowledge representation. `I.M.P.A.K.T.` takes all the information needed to understand and manage the domain of human resources from a specifically developed modular ontology. The ontology currently includes nearly 5000 concepts modeling both the technical and the complementary skills a candidate may hold. In particular, by technical skills we mean the candidate background knowledge about specific technologies and tools, while by complementary skills we mean personal and social abilities. In order to give an idea of the modeling effort, we note its development took one year with a working team composed of three knowledge engineers and one domain expert.

An upper level sketch of the modular KB structure, denoted by $\mathcal{K}$ is shown in Figure 2, with reference to both the intensional (TBox $\mathcal{T}$) and the extensional (ABox $\mathcal{A}$) components defined in Appendix A. Of course, the ontology modularity allows for extending it whenever a new category of work-related features is identified. Such an extension would change the knowledge compilation schema, whose construction process is able to support the incremental introduction of new modules in the ontology, as shown in Section 3.1. Hereafter, the content modeled in each ontology module is briefly described:

– `Knowledge` models the hierarchy of possible candidate competence and technical tools usage abilities, including classes like `FunctionalProgramming`, `Database`, `PHP` and `ZendFramework`, to name a few; moreover, the module provides a property to specify, for each competence, the related experience role `type` (*e.g.*, developer, administrator, and so on) and a predicate to convey the experience level, expressed in work `years`;

– `JobTitle` models the hierarchy of possible job positions, such as `TeacherAssistant` or `DatabaseAdministrator`; the module also provides a predicate to specify the job experience level (`years`).
– `Industry` models the hierarchy of industrial sectors where a candidate may have worked, such as `InstitutionalAreas`, `ResearchLaboratories`, `CompanyDepartments`; a module predicate allows also for defining `years` of work experience in a specific industry.
– `ComplementarySkill` models the class hierarchy about complementary attitudes (a.k.a. soft skills) such as `Cooperation`, `StressTolerance`, `Leadership`, `ProblemSolving`, which complete CV technical skills and competence and often help evaluating a candidate profile.
– `Level` models the hierarchy of candidate education and training levels: from basic education to `MasterDegree`, the whole candidate qualifications can be specified including specific `Certifications` she gained.
– `Language` models the hierarchy of possible languages known by the candidate and provides three concrete features to further classify language knowledge in `verbalLevel`, `readingLevel` and `writingLevel` and assign a reference level (from 1—basic—to 3—excellent) to such language skills.

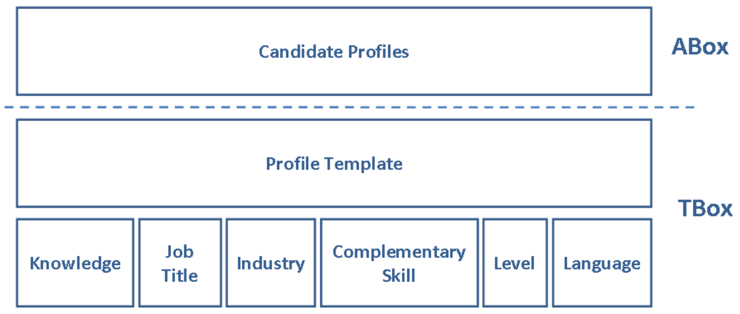


Fig. 2 Skill KB overview

In the following, we formally denote by $\mathcal{T} = \{M_i | 0 \leqslant i \leqslant 6\}$ the whole skills ontology adopted by the current `I.M.P.A.K.T.` implementation. All the classes modeling technical and complementary skills, briefly described above, belong to one of the ontology submodules M_i, with $i > 0$, shown in the lower layer in Figure 2. The component *Profile Template* in Figure 2 is the main ontology module, M_0. It directly imports all the previous modules and models all of the properties needed for describing the candidate profile through the above detailed classes. We define such properties by the name *entry points*. In particular, *Profile Template* includes one entry point for each imported sub-module: as an example, *hasComplementarySkill* (or alternatively *hasIndustry*) is the entry point which allows for specifying features of the candidate profile related to the category `ComplementarySkill` (respectively `Industry`).

Formally, an entry point is defined as follows:

Definition 1 (Entry Point) Given the skill ontology $\mathcal{T}$, an **Entry Point** R_j^0 with $1 \leqslant j \leqslant 6$, is a property defined in the module M_0 such that $(R_j^0)^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times M_j^{\mathcal{I}}$.

In order to fully represent the features of Human Resources management, real-life examples suggest that at least the following constructors are needed: conjunction, universal and existential quantification, and concrete features (see *e.g.*, *Robert* profile described at the end of this section). We did not find evidence that atomic negation is needed, at least when modeling a knowledge profile; in fact, profiles never require that a candidate *does not know* something. Also for describing a CV, negative information is not provided (no one starts a list of things s/he does not know). However, the interplay of existential and universal quantification leads to reasoning problems that are not computable in polynomial time [2, Ch.3], and such computational complexity hampers the translation into SQL of our problems (see Appendix B for proofs and a discussion). Therefore, `I.M.P.A.K.T.` adopts a CV representation (see Definition 2) allowing for reasoning only on $\mathcal{FL}_0(\mathrm{D})$ concepts which represent knowledge about our domain. Such a DL does not allow for atomic negation, coherently with the domain assumption motivated above, and drops existential quantification. As a consequence, each ontology module M_i, with $i > 0$, is modeled according to the formalism of $\mathcal{FL}_0(\mathrm{D})$.

Thanks to the knowledge modeling outlined so far, it is possible to model *Candidate Profiles* in the ABox (see Figure 2). We notice that all personal information in the CV, which are structured data by definition (*i.e.*, first/second name, address, telephone, car availability, etc.) are not considered for the logic formalization, given that they can be more properly represented straightly in the relational schema. The CV classification approach we propose is based on a role-free ABox, which then includes only concept assertions of the form $P(a)$, stating that the candidate a (*i.e.*, her CV description) offers profile features P.

The effectiveness and the efficiency of `I.M.P.A.K.T.` framework is intuitively double-tied to the design of the Entity-Relationship (E-R) model: only a properly designed storing of both ABox instances and TBox axioms may make the further reasoning stages work. Before starting with the description of the employed E-R model, we need to provide readers with some crucial definitions.

Definition 2 (Profile) Given the skill ontology $\mathcal{T}$, a profile P is an $\mathcal{ALE}(\mathrm{D})$ concept defined as a conjunction of existential quantifications, $P = \sqcap(\exists R_j^0.C)$, with $1 \leqslant j \leqslant 6$, where R_j^0 is an entry point and C is a concept in $\mathcal{FL}_0(\mathrm{D})$ modeled in the ontology module M_j.

We notice that the description of a profile would need the alphabet of $\mathcal{ALE}(\mathrm{D})$, but its structure is constrained by Definition 2; we cannot therefore exploit the full $\mathcal{ALE}(\mathrm{D})$ expressiveness to model the knowledge at hand. We point out, also, that the profile structure given in Definition 2 allows for six types of conjuncts modeling CV features, but the number of features is not constrained: the six entry points are just a way to identify CV sections and describe all items belonging to them.

The currently implemented list of possible conjuncts of a profile P is reported in Table 6 in Appendix C.

3.1 Knowledge Compilation schema

Our knowledge compilation problem aims at translating the skill knowledge base into a relational model, without loss of information and expressiveness, in order to reduce on-line reasoning time. So, relational schema modeling is the most crucial design issue and it is strongly dependent on both knowledge expressiveness to be stored and reasoning to be provided over such a knowledge.

Notice that all non-standard reasoning services performed by `I.M.P.A.K.T.` process the atomic information making up the knowledge descriptions to be stored rather then the concept as a whole. For this reason, the availability of a finite normal form for such descriptions turns out to be very useful and effective. We recall that $\mathcal{FL}_0$(D) concepts can be normalized according to the *Concept-Centered Normal Form* (CCNF), [2, Ch.2], through the recursive application of the formulas in Figure 3, until no rule is applicable at every nesting level.

TBox reduction			Concept reduction		
A	$\rightarrow$	$A \sqcap C$	$\forall R.(D \sqcap E)$	$\rightarrow$	$\forall R.D \sqcap \forall R.E$
		if $A \sqsubseteq C \in \mathcal{T}$			
A	$\rightarrow$	C			
		if $A = C \in \mathcal{T}$			

Fig. 3 Rules for $\mathcal{FL}_0$(D) CCNF.

A finite normal form is instead not available for $\mathcal{ALE}$(D) concepts and this motivates our choice to model CVs according to Definition 2. `I.M.P.A.K.T.` exploits all the informative content needed for the on-line reasoning phase by extracting $\mathcal{FL}_0$(D) components (namely each C in a Profile—see Definition 2) from modeled CVs.

In the following, we provide Definition 3 to denote what we mean by CCNF of a concept F—CCNF(F)—in the rest of the paper.

Definition 3 (CCNF(F)) Given the skill ontology $\mathcal{T}$:

1. if F is a concept description in $\mathcal{FL}_0$(D) modeled in the ontology module M_j, with $1 \leqslant j \leqslant 6$, then $CCNF(F)$ is the concept-centered normal form of F, computed according to rules in Figure 3.
2. if F is a conjunct in Definition 2, $F = \exists R_j^0.C$, we denote by CCNF(F) the concept description $CCNF(F) = \exists R_j^0.CCNF(C)$.
3. if F is a Profile according to Definition 2, $F = \sqcap(CJ)$, with $CJ = \exists R_j^0.C$, we denote by CCNF(F) the conjunction of all CCNF(CJ).

The E-R model resulting from knowledge compilation is sketched in Figure 4. In particular, the schema in Figure 4(a) shows all the entities needed to represent the Tbox $\mathcal{T}$ axioms and is obtained according to the following design rules:

1. a table Concept is created to store all the atomic information managed by the system: i) concept and role names; ii) the CCNF atoms of all the $\mathcal{FL}_0$(D) concepts defined in modules M_j, with $1 \leqslant j \leqslant 6$. Among attributes of Concept table, a specific relevance is assumed by `level`: it indicates the *depth level* of the concept name in the ontology (taxonomy).

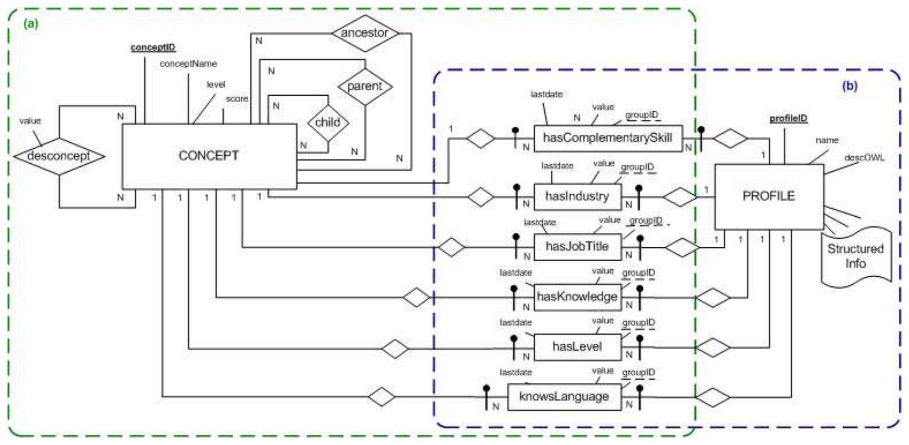


Fig. 4 Database relational schema

2. three tables mapping recursive relationships over the table Concept—namely Parent, Ancestor and Desconcept—are created to store, respectively, for a concept C, information about its parents, its ancestors and the atoms of its CCNF—CCNF(C)—in case C is a defined concept.
3. a table $R_j(X)$ is created for each entry point R_j^0 (see Definition 1), where X is a set of attributes $X = \{profileID, groupID, conceptID, value, lastdate\}$ detailed in the following (see Figure 6 for a toy example).

Notice that, thanks to the third rule, our model can be easily extended. If module M_0 in $\mathcal{T}$ is enhanced by a new entry point in order to capture a novel aspect of candidate CVs, then the schema may be enriched by adding the corresponding table $R_j(X)$ to it.

The schema in Figure 4(b) models instead the entities needed to store all features of candidate profiles $P(a)$.

Notice that Figure 4(a) and Figure 4(b) share the same tables $R_j(X)$, but from different levels of abstraction. In fact, the ontology modules in $\mathcal{T}$define the *schema level* by determining both the number and the structure of tables $R_j(X)$, whereas ontological features of candidate profiles $P(a)$ in $\mathcal{A}$ represent the *instance level* and populate such $R_j(X)$ tables.

Thus, all features modeled in profile descriptions according to Definition 2 are stored in tables $R_j(X)$ related to the involved entry points, while Profile table includes the so called *structured information*: extra-ontological content, such as personal data (*e.g.*, last and first name, birth date) and work-related information (*e.g.*, preferred working hours, car availability). Intuitively, each individual a is involved in one $P(a)$ assertion only, while the same feature P could be offered by more than one candidate.

Once the CCNF(P) of a profile $P = \sqcap(\exists R_j^0.C)$—with $1 \leqslant j \leqslant 6$—has been computed according to Definition 3, the assertion $P(a)$ is stored in the database. In particular, `I.M.P.A.K.T.` produces a unique identifier for the candidate a, assigned to attribute $profileID$ in Table Profile, and, for each conjunct $\exists R_j^0.C$ belonging to $P(a)$, it adds one tuple for each atom of the $CCNF(C)$ to the related table $R_j(X)$.

MasterDegree $\sqsubseteq$ Level

EngineeringDegree $\sqsubseteq$ MasterDegree

ComputerScienceEngineeringDegree $\sqsubseteq$ EngineeringDegree

English $\sqsubseteq$ Language

Programming $\sqsubseteq$ KnowledgeType

EngineeringAndTechnologies $\sqsubseteq$ Knowledge

Java $\sqsubseteq$ OOP

C++ $\sqsubseteq$ OOP

VisualBasic $\sqsubseteq$ OOP

OOP $\sqsubseteq$ ProgrammingLanguage $\sqsubseteq$ SWDevelopment $\sqsubseteq$ ComputerScienceSkill $\sqsubseteq$ EngineeringAndTechnologies

PostgreSQL $\sqsubseteq$ RDBMS $\sqsubseteq$ OpenSourceDBMS $\sqsubseteq$ DBMS $\sqsubseteq$ InformationSystem $\sqsubseteq$ ComputerScienceSkill

InternetTechnologies $\sqsubseteq$ ComputerScienceSkill

ArtificialIntelligence $\sqsubseteq$ ComputerScienceSkill

Fig. 5 The ontology sketch used as reference in examples

In order to better clarify the role of tables $R_j(X)$, consider the informative content in hasKnowledge (*i.e.*, $R_j = hasKnowledge$). In such a table, `I.M.P.A.K.T.` archives all technical skills and competence held by $P(a)$, where each conjunct $\exists hasKnowledge.C$ describes a single skill of $P(a)$. Thus, the entry point *hasKnowledge* identifies the table for storing the profile competence, but the information actually modeling the held skill is described by C—or equivalently by $CCNF(C)$. As a consequence, to fully convey all the competence and technical knowledge in a given profile (identified by `profileID`), the full set of tuples in hasKnowledge table, related to that `profileID`, is needed. We notice that the same candidate profile $P(a)$ may include more than one conjunct involving the same entry point. For example, both $\exists hasKnowledge.C$ and $\exists hasKnowledge.D$ could belong to the same $P(a)$. In accordance with the relational schema introduced so far, the atoms of both CCNF(C) and CCNF(D) are stored in the same table hasKnowledge, but two different *groupID* values are assigned to C and D atoms, respectively (see Figure 6 for a toy example).

To further clarify both usefulness and effectiveness of the previous E-R modeling w.r.t. skill and talent management domain, a toy profile description is proposed hereafter.

Let us suppose *Robert* is a candidate; his profile can be described as: *"Robert speaks English with a scholar level whereas he is doing better with written English. He has a degree in Computer Science Engineering, with mark equal to 110 (out of 110), an excellent experience in Java programming (5 years until December 10, 2010) and he is two years experienced in PostgreSQL DBMS (until July, 2011), ...". Robert* profile can be represented according to the following features:

– ***hasLevel*** - Computer Science Engineering Degree (final mark = 110)
– ***hasKnowledge*** - Java (knowledge type = programming, years of experience = 5, last update = 2010-12-10); PostgreSQL (years of experience = 2, last update = 2011-07-31)
– ***knowsLanguage*** - English (verbalLevel=1, writingLevel=2).

Robert profile is modeled according to Definition 2 and Table 6 (see Appendix C) in a concept P as in the following:

Concept		
conceptID	*name*	*level*
1	mark	null
2	year	null
3	skillType	null
4	lastdate	null
5	skillType.Programming	2
6	skillType.KnowledgeType	1
7	Knowledge	1
8	EngineeringAndTechnologies	2
9	ComputerScienceSkill	3
10	SWDevelopment	4
11	ProgrammingLanguage	5
12	OOP	6
13	Java	7
14	PostgreSQL	8
15	OpenDBMS	7
...	...	...

hasKnowledge				
profileID	*groupID*	*conceptID*	*value*	*lastdate*
100	1	13	null	null
100	1	12	null	null
100	1	11	null	null
100	1	10	null	null
100	1	9	null	null
100	1	8	null	null
100	1	7	null	null
100	1	6	null	null
100	1	5	null	null
100	1	2	5	2010-12-10
...	...	...	...	...
100	2	14	null	null
100	2	15	null	null
...	...	...	...	...

Profile							
profileID	*first_name*	*second_name*	*birth_date*	...	*working_hours*	*car*	...
100	Robert	Wane	1959-07-15	...	9-14	true	...
...	...	...	...	...	...	...	...

Fig. 6 Tables filled to store one feature of Robert profile ($profileID = 100$)

$\exists hasLevel.(ComputerScienceEngineeringDegre\sqcap =_{110} mark) \sqcap$
$\exists hasKnowledge.(Java \sqcap \forall skillType.Programming\sqcap =_5 years\sqcap =_{2010-12-10} lastdate)\sqcap$
$\exists hasKnowledge.(PostgreSQL\sqcap =_2 years\sqcap =_{2011-07-31} lastdate)\sqcap$
$\exists knowslanguage.(English \sqcap =_1 verbalLevel\sqcap =_2 writingLevel).$
`I.M.P.A.K.T.` normalizes and splits P in components then stored in the three tables: hasKnowledge, hasLevel and knowsLanguage.

With reference to the model in Figure 4 and ontology sketch in Figure 5, tuples storing only the conjunct $\exists hasKnowledge.(Java\sqcap\forall skillType.Programming\sqcap =_5 years\sqcap =_{2010-12-10} lastdate)$ are reported in Figure 6. The reader can notice that the conjunct:
$\exists hasKnowledge.(PostgreSQL\sqcap =_2 years\sqcap =_{2011-07-31} lastdate)$
has a $groupID$ value different from 1 in Table HasKnowledge, but the same `profileID` value.

In order to improve readability, in Figure 4 we do not represent tables needed to store intermediate results for computing the final reasoning task. Such tuples are materialized in the proper tables and views, created at runtime according to user requirements. Finally, the presented modeling approach translates a profile assertion $P(a)$ of n conjuncts into more than n database tuples (see Figure 4, as an example). It therefore increases the storage size, almost linearly. Nevertheless, such a drawback is largely repaid in terms of flexible match classes management and quick logic-based ranking and explanation of results as reported in the following.

4 I.M.P.A.K.T. in action

I.M.P.A.K.T. is a Java web services application. It uses Jena API[13] to access the underlying ontology model and Pellet[14] reasoner to classify ontologies in the preprocessing phase. It is noteworthy that, every time there are no implicit axioms in the reference ontology, it is possible to give up the reasoner by disabling its service, thus improving system performance. I.M.P.A.K.T. has been developed upon the open source PostgreSQL 9.3 DBMS and uses: **(1)** auxiliary tables and views to store the intermediate results; and **(2)** stored procedures and b-tree indexes on proper attributes to reduce retrieval times. Moreover, the compliance with standard SQL makes I.M.P.A.K.T. available for a broad variety of platforms.

A real data set originated from three different employment agencies—one having the headquarter in Italy and the other two in France—was initially created by collecting approximately 180 CVs of candidates specifically skilled in the ICT domain, so to simulate the scenario of an actual company in the ICT industry. Such a dataset has also been exploited for an iterative refinement phase of both the Skill Ontology development and the setting of the *Skill Matching* parameters (*i.e.*, entry points levels and weights in scoring strategy).

In the next subsections, we show I.M.P.A.K.T. working mode through an extended example covering, respectively, three services for human resource management: i) skill matching for the retrieval of ranked referral lists, ii) working team composition, iii) automated extraction of strategical enterprise competence. The subset of ten candidate profiles (out of 180) used throughout the example is given in Appendix D. Profiles have been chosen in order to bring out the special features of our approach during the presentation of solution processes. To this aim, we also consider in the following up-to-date all the features tied to experience years specification in each candidate profile. We outline that actual semantic profiles can be imported by editing I.M.P.A.K.T. Graphical User Iinterface (GUI), specifically built for inserting both structured information and ontologically one by means of ontology browsing.

4.1 Skill Matching

When dealing with the search for the right candidate to assign to a job, recruiters are often in front of cases of non-perfect match: the availability of knowledge profiles fully satisfying a job request, even though desirable, is quite a rare event. In most cases, therefore, the search for a candidate profile more specific than the knowledge request at hand fails and needs to be followed by the search for profiles only approximately matching the request.

Obviously, in order to evaluate the matching degree between a job request and a candidate profile, it is necessary that both of them share the knowledge base used for representation. Thus, the job requests submitted to I.M.P.A.K.T. need to be represented according to the syntax detailed in Definition 2, that is the same formalism employed for candidate profiles representation. According to the same

[13] http://jena.apache.org/

[14] http://pellet.owldl.org/

profile data structure, both required and provided knowledge profiles are modeled and each conjunct $\exists R_j^0.C$ represents either a feature to search or to store.

The formal approach adopted by `I.M.P.A.K.T.` to perform the *Skill Matching* process has been presented in our past research [55]. We here just recall that, for a candidate profile $P(a)$ to fully satisfy a job request $\mathcal{F}$, both formalized as a DL concept description, there should exist a subsumption relation between P and $\mathcal{F}$, formally $P \sqsubseteq \mathcal{F}$. If, instead, subsumption does not hold, specific non-standard inferences can be exploited both to retrieve a ranked referral list of candidates only approximately matching the request, and to explain the reasons for the absence of a perfect match in terms of missing (through a Concept Abduction Problem) or conflicting (through a Concept Contraction Problem) features.

Moreover, a job opening may have some features strictly required, which have to be necessarily fulfilled by selected candidates, and some other characteristics managed as preferences. Thus, both groups of user requirements (preferences $\mathcal{FP}$ and strict constraints $\mathcal{FS}$) compose a job request $\mathcal{F}$. In the most general case of job request $\mathcal{F}$ containing both $\mathcal{FS}$ and $\mathcal{FP}$, `I.M.P.A.K.T.` performs a two-step matchmaking approach, namely *Matchmaking*, which starts with *Strict Match* process, computing a set of profiles fully satisfying strict requirements (*i.e.*, we can retrieve candidate profiles $P(a)$ more specific than $\mathcal{FS}$) and then proceeds with *Soft Match* process trying to approximately match preferences with profiles belonging to the set returned by *Strict Match*. The *Soft Match* is devoted to finding candidates satisfying to some extent the preferences $\mathcal{FP}$ in the job request $\mathcal{F}$ by implementing the above mentioned approach to approximate matching. In particular, the search has to revert also to candidates having some missing features and/or having features *slightly conflicting* w.r.t. $\mathcal{FP}$.

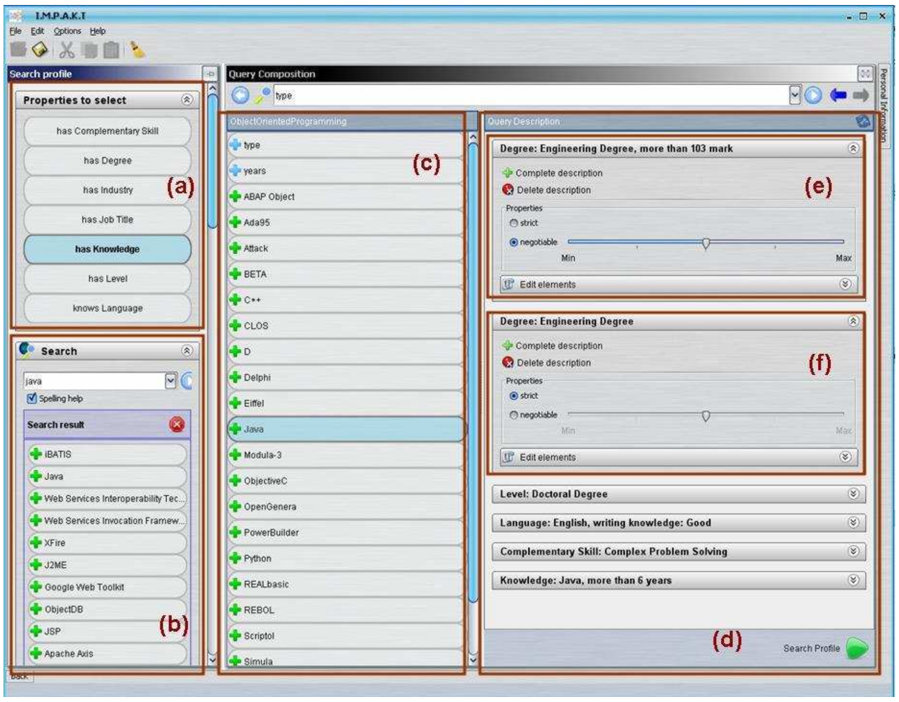
In order to clarify *Skill Matching* behavior, we start from a typical request of a recruiter:

I'm looking for a candidate having an Engineering Degree (preferably in Computer Science with a final mark equal or higher than 103 (out of 110)). A Doctoral Degree is welcome. A good familiarity with written English could be great. Furthermore s/he should be at least six years experienced in Java and s/he should possibly have complex problem solving capabilities.

`I.M.P.A.K.T.` provides a GUI to compose recruiter's requests: Figure 7 shows such a GUI w.r.t. the previous example request, which we refer to with **Q2** in the following. Observe that `I.M.P.A.K.T.` provides exactly the same interface used for defining/updating the candidate profile, coherently with our approach that defines the same template for CVs and queries description.

By looking at Figure 7, we shortly introduce all highlighted panels in the following:

- Panel (a) provides a GUI section for each of the six entry points (see Definition 1) described in Section 3, with an additional *Degree* section only for recruiter convenience;
- Panel (b) allows users to perform a keyword-based search aimed at identifying ontology concepts modeling the requirements they have in mind (such a search process involves concept names, labels and comments); it allows one also to search candidates profiles comparable to given ones by editing first and last name of the known candidate;



Degree: Engineering Degree, more than 103 mark

Degree: Engineering Degree

Level: Doctoral Degree

Language: English, writing knowledge: Good

Complementary Skill: Complex Problem Solving

Knowledge: Java, more than 6 years

Fig. 7 Query composition GUI

– Panel (c) enables the user to explore both taxonomy and properties of a concept selected to specify a given entry point;
– Panel (d) shows all of the features required in the query;
– Panels (e)-(f): for each item in Panel (d), the GUI allows users: **(1)** to specify whether the feature is a strict one—Panel (f)—or if it is a preference—Panel (e); **(2)** to delete the whole feature; **(3)** to complete the description showing all the elements (concepts, roles and concrete features) that could be added to the selected feature; **(4)** to edit either feature atoms or existing feature values.

Q2 request shown in Figure 7 can be summarized as:

1. **strict requirements**: Engineering Degree;
2. **preferences**: Computer Science Engineering Degree and final mark >= 103; Doctoral Degree; Java with experience >= 6 years; Complex Problem Solving capabilities; Good Written English.

In the following, we show how the approach we propose can provide an answer to **Q2**, including both strict requirements and preferences, w.r.t. to the data set in Appendix D. Part (a) of Figure 8 presents the ranked list of returned candidates (9 out of 10 available ones) with the related score. The ranked list can be explained as follows:

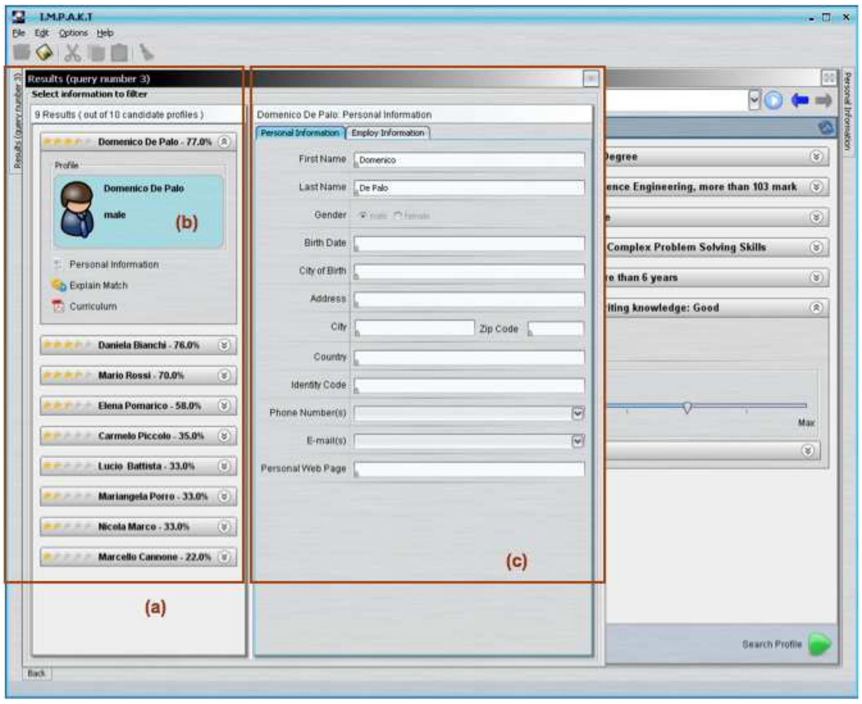
– only 9 out of 10 selected profiles are returned by the *Strict Match*. In fact, the candidate *Carla Buono* does not fulfill strict requirements specified in **Q2** and then she will not be part of the final result set;

– a small subset of candidate sample (*i.e.*, *Mario Rossi*, *Daniela Bianchi* and *Elena Pomarico*) is made up by people with similar profiles: their CVs only differ by experience years associated to either job titles, enterprise working or exploitation of a given competence. In particular, years of experience are never specified in *Elena Pomarico*'s profile. Such profiles allow us to make clear how these differences, even slight, cause profiles to be differently ranked. Hence, *Daniela Bianchi* is the best result among them because she fully satisfies the Java experience requested by the recruiter together with the strict requirements;
– the best results are *Domenico De Palo* and *Daniela Bianchi*. The former satisfies several preferences but he is 6 years experienced in object oriented programming (a direct ancestor of Java programming in the reference ontology) whereas the latter is 6 years experienced in Java, as required. Observe that *hasKnowledge* entry point, exploited to represent the user requirements of Java programming, belongs to the main relevance level in the ranking strategy;
– two profiles (*i.e.*, *Mario Rossi* and *Carmelo Piccolo*) include features slightly conflicting (according to the *Soft Match* definition) with query preferences;
– several candidates (*i.e.*, *Mariangela Porro*, *Marcello Cannone*, *Carmelo Piccolo*, *Lucio Battista* and *Nicola Marco*) satisfy only a few characteristics other than strict requirements. The scoring mechanism ranks them lower than all the other profiles, better filling query preferences;
– many selected profiles have additional features w.r.t. the query, although they do not affect the ranking mechanism.

With reference to Figure 8, for each of the ranked results, the recruiter can ask for: **(1)** viewing the CV (Panel (b)); **(2)** analysing the employment and personal information (Panel (c)); and **(3)** executing the match explanation procedure. In Figure 9 match explanation outcomes of *Mario Rossi* candidate are presented. In particular, in Panels (a)-(b) an overview of the request is shown (differentiating strict constraints—Panel (a)—from preferences—Panel (b)), whereas Panels (c)-(d)-(e)-(f) show the following information:

– Panel (c) shows *fulfilled features*, *i.e.*, features required by the query and either perfectly matched by the candidate or slightly conflicting in her profile;
– Panel (d) provides *additional features*, *i.e.*, technical skills in the candidate CV not required at all by the query or more specific than the required ones;
– Panel (e) gathers *underspecified features*, *i.e.*, parts of the query not explicitly specified in the retrieved CV;
– Panel (f) provides *conflicting features* as specified in the retrieved CV.

Let us consider feature 6 in Panel (b) of Figure 9: "*at least six years of Java experience*". By looking at Panel (c), it can be noticed that *Mario Rossi* is 5 years experienced in Java and Object Oriented Programming. Such a conflict is highlighted in the GUI as shown in Panel (f) so making the recruiter aware of profile features slightly conflicting with the query. Notice that the same preference (*i.e.*, the one identified by $ID = 6$) generates two pieces of information in Panel (c): the candidate CV includes both Java and some Object Oriented Programming 5 years experienced knowledge. Such a duplication in fulfilled features does not introduce redundancy and it is instead exploited to show in Panel (f) CV conflicting features related both to Object Oriented Programming (lack of one more year of working



Results (query number 3)
Select information to filter
9 Results (out of 10 candidate profiles)

- Domenico De Palo - 77.0%
  - Profile: Domenico De Palo, male
  - Personal Information
  - Explain Match
  - Curriculum
- Daniela Bianchi - 76.0%
- Mario Rossi - 70.0%
- Elena Pomarico - 58.0%
- Carmelo Piccolo - 35.0%
- Lucio Battista - 33.0%
- Mariangela Porro - 33.0%
- Nicola Marco - 33.0%
- Marcello Cannone - 22.0%

Domenico De Palo: Personal Information
Personal Information | Employ Information

First Name	Domenico
Last Name	De Palo
Gender	male
Birth Date	
City of Birth	
Address	
City	
Zip Code	
Country	
Identity Code	
Phone Number(s)	
E-mail(s)	
Personal Web Page	

egree
ence Engineering, more than 103 mark
e
Complex Problem Solving Skills
e than 6 years
iting knowledge: Good

Fig. 8 *Skill Matching* results GUI after $Q2$ execution

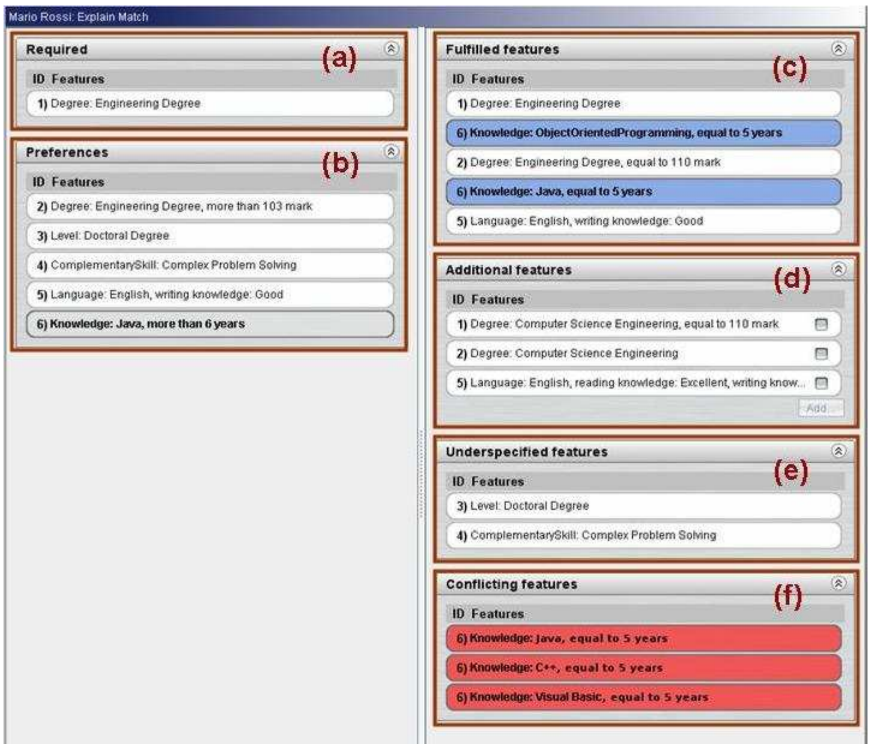
experience in C++ and Visual Basic) and to specific Java knowledge. Besides the conflicting features, *Mario Rossi* has also some underspecified ones (see Panel (e)) and then, he cannot fully satisfy the recruiter request.

4.2 Team Composition

`I.M.P.A.K.T.` *Team Composition* service aims at supporting the process of assigning more than one task to different groups of available individuals: a process we denote by *many to many* skill matching. The formal approach adopted by `I.M.P.A.K.T.` to perform such a process has been presented in previous work [54].

Such service supports a Project Manager by automatically providing a set of possible working teams for each task or project activity. In particular, a project manager request is composed by the descriptions of all the project tasks to be covered. In turn, each project activity (or task) description, PA_i, is composed by three entities: (i) a description K_i of the knowledge required for the task; (ii) a set of temporal constraints D_i and (iii) a number of required team members m_i. Formally, $PA_i = \langle K_i, D_i, m_i \rangle$ where K_i holds a crucial role in the selection of candidates profiles set at the basis of the *Team Composition* process.

Coherently with the strategy adopted by the previously detailed *Matchmaking* process, team composition takes full advantage from the ontology modeling effort and adopts the OWA. In fact, it allows one not only to find a team that, based on provided skills descriptions, fully covers K_i, but also teams only approximating


Mario Rossi: Explain Match

Required (a)

ID	Features
1)	Degree: Engineering Degree

Preferences (b)

ID	Features
2)	Degree: Engineering Degree, more than 103 mark
3)	Level: Doctoral Degree
4)	ComplementarySkill: Complex Problem Solving
5)	Language: English, writing knowledge: Good
6)	Knowledge: Java, more than 6 years

Fulfilled features (c)

ID	Features
1)	Degree: Engineering Degree
6)	Knowledge: ObjectOrientedProgramming, equal to 5 years
2)	Degree: Engineering Degree, equal to 110 mark
6)	Knowledge: Java, equal to 5 years
5)	Language: English, writing knowledge: Good

Additional features (d)

ID	Features
1)	Degree: Computer Science Engineering, equal to 110 mark
2)	Degree: Computer Science Engineering
5)	Language: English, reading knowledge: Excellent, writing know...

Add

Underspecified features (e)

ID	Features
3)	Level: Doctoral Degree
4)	ComplementarySkill: Complex Problem Solving

Conflicting features (f)

ID	Features
6)	Knowledge: Java, equal to 5 years
6)	Knowledge: C++, equal to 5 years
6)	Knowledge: Visual Basic, equal to 5 years

Fig. 9 *Mario Rossi* match explanation with conflicting features

such a full cover. Of course the search reverts to such a partial cover only when a completely satisfactory group cannot be retrieved due to lack of requested skills or temporal unavailability of candidates. Moreover, the *Team Composition* process considers equivalent all possible solutions—several combinations of candidates allocation may exist—and leaves the selection of the most proper set of assignments up to the Project Manager, given the high level of subjectivity of the task.

We notice that, in order to retrieve the candidate profiles satisfying as much as possible skills K_i of task PA_i, both K_i and P have to be described according to Definition 2. In particular, K_i conjuncts of the form $\exists R_j^0.C$ employ only the entry point $R_j^0 = hasKnowledge$: *TeamComposition* involves currently only technical skills in the search process.

Given a set $\mathcal{PA}$ of project activities $PA_i = \langle K_i, D_i, m_i \rangle$, *Team Composition* process is basically performed according to the following steps:

1. the *Matchmaking* process in Section 4.1 is performed by taking as input each conjunct of a Project Manager request, that is a job request $\mathcal{F}$ including only *hasKnowledge* entry point, in accordance with the above introduced considerations on K_i;
2. for each PA_i the candidates' availability is checked by joining the set of profiles returned by the previous step with tuples in a table—AVAILABILITY—specifically defined in the DB to store candidates temporal constraints (even

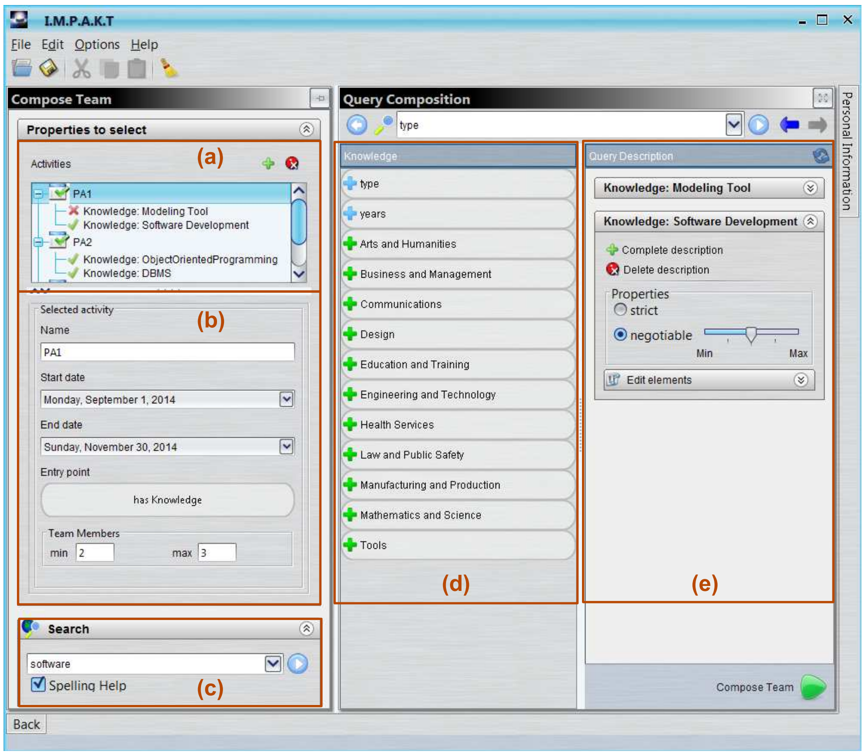
though not represented in Figure 4 for the sake on synthesis). Only candidate profiles satisfying the constraints in D_i are returned;

3. the set of all candidate teams, $Assignments = \{Assignments(PA_i)|1 \leqslant i \leqslant N\}$, where each $Assignments(PA_i)$ is a set of all possible teams covering PA_i, is computed without taking into account the need for executing $PA_j \in \mathcal{PA}$ with $i \neq j$. In turn, each team in $Assignments(PA_i)$ is computed by taking into account both the need for covering as much as possible required skills K_i and the required number of team members m_i;
4. a Constraint Satisfaction Problem (CSP)[58] solver computes the set of all possible solution teams solving the whole set of project activities $\mathcal{PA}$. Such solutions are obtained by considering the elements in the set *Assignments* returned at *step 3* as variables and temporal information in D_i as constraints, for each activity PA_i: the final goal is obviously returning an assignment set such that concurrent activities never involve the same profile.

`I.M.P.A.K.T.` provides a GUI specifically built to compose the inputs to *Team Composition.* According to the above definition of project activities, let us consider a *project manager* searching for a work team to employ in a new project composed by 3 activities (see Panels (a)-(b) of Figure 10):

1. **PA1 – Architecture Design**
   Start Date: 2014-09-01
   End date: 2014-11-30
   Team: from 2 to 3 candidates
   Skill: Modeling tool (preference), Software Development (preference)
2. **PA2 – Data Layer**
   Start Date: 2014-10-13
   End date: 2014-12-20
   Team: 2 candidates
   Skill: Object Oriented Programming (preference), DBMS (preference)
3. **PA3 – Implementation Layer**
   Start Date: 2014-12-15
   End date: 2015-04-30
   Team: from 2 to 3 candidates
   Skill: J2EE (preference), Hibernate (preference)

Observe that, by using `I.M.P.A.K.T.` GUI in Figure 10, the project manager can describe each activity by means of technical knowledge needed for solving it. It is also possible to set such required competence as strict requirement or preference. For each project activity, Panel (d) in Figure 10 enables ontology browsing only for defining required knowledge. In particular, Panels (c), (d) and (e) provide the same functionalities (*i.e.*, "global" keyword-based search on ontology entities, ontology classes and properties browsing, and features editing, respectively) showed for the query composition GUI of *Skill Matching* service (see Section 4.1 for details). Panel (b) is introduced in order to edit all the activity features: name, start date, end date, number of team members. Moreover, Panel (a) shows an overview of all the selected skills for each activity. It is noteworthy that, by adding a required knowledge in Panel (e), `I.M.P.A.K.T.` automatically performs the matchmaking process, and then it highlights by means of a red icon those skills not covered by any profiles. In our example, Panel (a) shows a red icon for the knowledge of a *Modeling Tool* to indicate that it cannot be satisfied. This happens when no candidate experienced in the required knowledge exists. In other words, before starting the composition process, the system acknowledges the user whether each of the required task competencies is covered or not. In this way, he can decide how to proceed: choosing another competence and/or deleting the non-covered competence, given that it does not have correspondence in candidates retrieval nor in results composition.



PA1
- Knowledge: Modeling Tool
- Knowledge: Software Development

PA2
- Knowledge: ObjectOrientedProgramming
- Knowledge: DBMS

Selected activity
Name: PA1
Start date: Monday, September 1, 2014
End date: Sunday, November 30, 2014
Entry point: has Knowledge
Team Members: min 2 max 3

Search: software
[✓] Spelling Help

Query: type

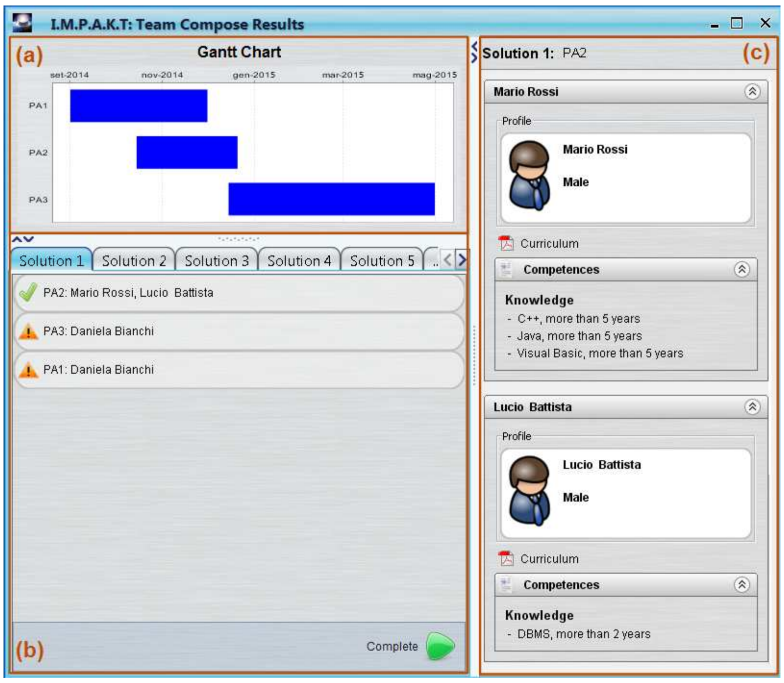
Knowledge: Modeling Tool
Knowledge: Software Development
Properties: strict [✓ negotiable] Min Max

Fig. 10 Activities definition GUI

With reference to profiles in Appendix D and to the activities introduced before, team solutions are presented in Panel (b) of Figure 11. In particular, the figure shows team members assigned by *Solution 1*. For example, *Mario Rossi* and *Lucio Battista* are the selected team members for activity **PA2**, thanks to their experience, respectively, in object oriented programming (*i.e.*, Java, C++, and Visual Basic) and DBMS, as showed by the "'Competence"' list in Panel (c) of Figure 11. We point out that, the system is able to assign the same Human Resource to different activities when these one are not overlapped (*e.g., Daniela Bianchi* is assigned to both activities **PA1** and **PA3**).

For each activity, the previous panel presents candidate lists to solve the task, and `I.M.P.A.K.T.` GUI also supports team completion process in case of selection of teams smaller than required (*i.e.*, teams indicated with a "warning" icon in the same figure's panel). The team completion process is performed using again the CSP solver having as input the same temporal constraints but different variables.

In fact, `I.M.P.A.K.T.` now considers a smaller profiles set (candidates already selected have not to be considered anymore). An assignment is incomplete when no other candidate covering both the required skills and the selected temporal view exists. Hence, in order to select candidates among the available ones, several strategies can be tested. `I.M.P.A.K.T.` currently favours candidates with the highest rank value and, among equally ranked ones, it chooses candidates with the highest number of additional features w.r.t. the required ones.
Additionally, Panel (c) of Figure 11 shows candidates description (*i.e.*, the relative CV and competence list), whereas a temporal scheduling of work activities is represented through a Gantt chart in Panel (a) of the same figure.



Solution 1: PA2

PA2: Mario Rossi, Lucio Battista
PA3: Daniela Bianchi
PA1: Daniela Bianchi

Mario Rossi
Mario Rossi
Male
Knowledge
- C++, more than 5 years
- Java, more than 5 years
- Visual Basic, more than 5 years

Lucio Battista
Lucio Battista
Male
Knowledge
- DBMS, more than 2 years

Fig. 11 Team Composition results GUI

4.3 Core Competence Identification

In recent enterprise solutions research literature emphasis has been given, in particular, to the identification of capabilities leading companies to business success: several approaches to strategic management have been proposed and classified according to the perspective they take on the problem [24].

Many research contributions sustain the *resource-based theory* of the company [61,62] suggesting to search for competitive advantage in unique company capabilities [3,4,34,37].

Other proposals focus on the achievement of competitive advantage through the deployment and exploitation of capabilities embedded in business processes:

such a *dynamic capabilities approach* asks for continuous reshaping of firms assets [52].

Alternative approaches [7,47,51] take the so-called *competence-based perspective*, which identifies in the *Core Competence* of the company as a whole a source of competitive advantage more crucial than the its single, discrete, assets. The notion of Core Competence was firstly defined [25] as a sort of capability providing customer benefits, hard to be imitated from competitors and possessing leverage potential. Further definitions of Core Competence have been proposed in the literature in the attempt of finding methods for detecting such a collective knowledge [35,39].

Our proposal takes the competence-based perspective and in particular shares with it the interpretation of company strategic competence as a collective asset, resulting from the synergy of human resources.

To this aim, `I.M.P.A.K.T.` services rely on the computation of partial Common Subsumers, formally defined in our past research [15]. In that paper it is proposed a *Common Subsumer Enumeration* algorithm determining the sets of common subsumers of a collection $\{C_1, \ldots, C_p\}$. The rationale of the algorithm is that of extracting from the set of profiles at hand, the knowledge components shared by a significant number of individuals in the set, with such a significance level to be set as a threshold value by the people in charge for strategic analysis. The algorithm works by taking as input a concept collection in the form of a *Subsumers Matrix* and the above introduced threshold value.

`I.M.P.A.K.T.` implements the above recalled service, but redefines the *Subsumers Matrix* as a a *Profiles Subsumers Matrix*, in order to cope with the features of the concept collection at hand. The formal approach adopted by `I.M.P.A.K.T.` has been presented in previous work [18].

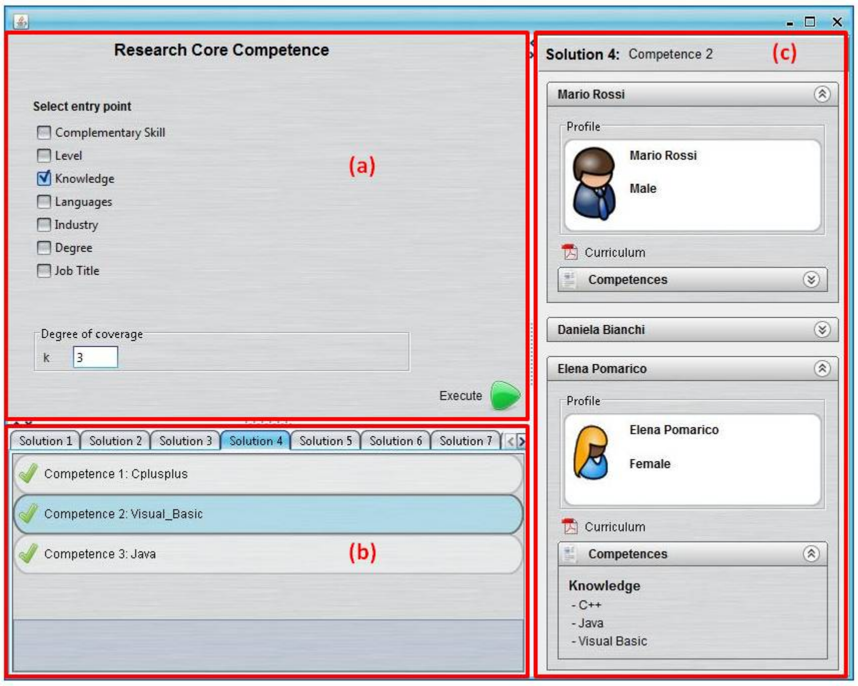
In order to understand the rationale of Core Competence Identification, we show how `I.M.P.A.K.T.` works with reference to the example CVs shown in D. In particular, in order for the problem representation to be more compact, we take—w.l.o.g.—the following assumptions: i) only CV information related to technical knowledge is taken into account; ii) we consider a subset P of CVs in D such that the modeled technical knowledge involves only concepts represented in Figure 5 and thus related to Computer Science domain: $P = \{1, 2, 3, 4, 5, 9, 10\}$.

In Figure 12, the `I.M.P.A.K.T.` GUI for the Core Competence Identification process is shown. Panel (a) provides the input user interface for choosing the degree of coverage k and the desired entry points to be considered in the extraction process. Panel (b) lists all possible pieces of company Core Competence, providing the user with the possibility to visualize (in Panel (c)) the personnel holding such a strategic asset.

5 Experimental evaluation

Section 4 shows the efficacy of the proposed approach, with reference to its three main services, as performed by `I.M.P.A.K.T.`.

Assuming such an efficacy, we here focus only on performance tests, in order to evaluate *data complexity* and *expression complexity* of our knowledge compilation approach.



Research Core Competence

Select entry point

- [ ] Complementary Skill
- [ ] Level
- [x] Knowledge
- [ ] Languages
- [ ] Industry
- [ ] Degree
- [ ] Job Title

Degree of coverage

k 3

Execute

Solution 1 | Solution 2 | Solution 3 | Solution 4 | Solution 5 | Solution 6 | Solution 7

✓ Competence 1: Cplusplus

✓ Competence 2: Visual_Basic

✓ Competence 3: Java

Solution 4: Competence 2

Mario Rossi

Profile

Mario Rossi

Male

Curriculum

Competences

Daniela Bianchi

Elena Pomarico

Profile

Elena Pomarico

Female

Curriculum

Competences

Knowledge
- C++
- Java
- Visual Basic

Fig. 12 Core Competence Identification GUI

We recall that both *Team Composition* and *Core Competence Identification* services rely on the performance of the matching service introduced in Section 4.1. For this reason, we start (see Section 5.1) by testing the performance of *Skill Matching* process and then refer to the achieved results for evaluating the execution times of the other two processes (see Section 5.2 and Section 5.3).

Execution times are retrieved by running `I.M.P.A.K.T.` on an Intel Dual Core server, equipped with a 2.26 GHz processor and 4 GB RAM.

All tests measure the average time over ten repetitions of the same request for each service.

Datasets specifically suited for testing each service have been designed and described in the following subsections. In particular, different datasets are selected among the ones possibly returned by a specifically developed synthetic KB instances generator. The generator is able to automatically create satisfiable profiles according to Definition 2, according to a generation criterion which may be set on the basis of testing needs. For example, one may choose the features format (*i.e.*, number of features for each entry point/relevance level, number of numeric restrictions, minimum number of specified technical skills, etc.). Indipendently on the service, we built datasets in which profiles have a number of features for candidate comparable to the average value evaluated in the real data set discussed in Section 4.

5.1 Skill Matching

As hinted before, we tested each service with reference to datasets specifically selected for their suitability to the solving approach to experiment.

In particular, for testing *Skill Matching*, we generated five data sets, namely DS_1, DS_2, DS_3, DS_4, DS_5 including respectively 500, 1000, 2000, 3500 amd 5500 profiles. The bigger datasets are supersets of the smaller ones. Furthermore, the format of generated profiles is set to include 30 features for *hasKnowledge* entry point, 2 features for *hasLevel* and *knowsLanguage* entry points, and 3 features for *hasJobTitle*, *hasIndustry* and *hasComplementarySkill* entry points.

In this specific service, also queries to be performed need to be properly selected for testing pourpose. In particular, we refer to nine queries, which we consider significantly different in expressiveness and which we classify into the following three groups:

A queries including only strict requirements, described by either generic concepts (Q_4) or more specific ones (Q_5);
B queries including only preferences, also represented by either generic concepts (Q_2) or more specific ones (Q_3);
C queries combining features in items A and B (Q_1,Q_6,Q_7,Q_8,Q_9).

We also notice that:

1. query Q_1 is the formal profile (see Definition 2) translating a real job request returned by `http://www.monster.co.uk/`, exploited keywords: *SQL, SSAS, OLAP Cube, C#*; Q_1 thus includes 3 strict and 6 soft requirements for the entry point *hasKnowledge*, 1 strict and 3 soft requirements for the entry point *hasComplementarySkill*, 1 and 3 soft requirements for the entry *hasIndustry* and *hasJobTitle*, respectively.
2. queries from Q_2 to Q_7 include one feature per entry point.
3. $Q_6 = Q_2 \cup Q_4$ and $Q_7 = Q_3 \cup Q_5$.
4. Q_8 involves only three entry points, *i.e.*, *hasKnowledge*, *knowsLanguage* and *hasLevel*.
5. Q_9 involves several features for each entry point.

Table 1 shows the retrieval times and the number of retrieved profiles ($\#p$) for each data set and request discussed above.

Reported times refer to three distinguished phases of the matching process: query normalization (retrieval times denoted by t_n in Table 1), which is dataset-independent; *Strict Match* (retrieval times denoted by t_{st} in Table 1) and *Soft Match* (retrieval times denoted by t_{sf} in Table 1), which also includes the final ranking process.

As for data complexity, the reader may notice that, not surprisingly, retrieval times of both match processes increase almost linearly with datasets size (*e.g.* see Q_5). Furthermore, *Strict Match* performance is dramatically affected by $\#p$ (see results for DS_5 in Table 1), whereas the *Soft Match* retrieval times seem to grow more slowly with $\#p$. We observe that the more significant impact of the number of retrieved profiles on *Strict Match* execution is due to the fact that such a process involves by construction the SQL intersection of several queries. Moreover, results of Q_5 are dataset-independent: the *Strict Match* process always returns the same profile (*i.e.*, no other profile satisfying strict requirements exists in the datasets).

Table 1 Normalization times (t_n) and retrieval times for strict (t_{st}) and soft (t_{sf}) match, and number of retrieved profiles ($\#p$) for datasets DS_1, DS_2, DS_3, DS_4 and DS_5 of, respectively, 500, 1000, 2000, 3500 and 5500 profiles. Times are expressed in milliseconds.

		DS_1			DS_2			DS_3			DS_4			DS_5		
	t_n	t_{st}	t_{sf}	$\#p$	t_{st}	t_{sf}	$\#p$	t_{st}	t_{sf}	$\#p$	t_{st}	t_{sf}	$\#p$	t_{st}	t_{sf}	$\#p$
Q_1	724.4	124.2	210.6	4	246.6	240.8	10	545.6	382.5	20	784.5	402.2	28	2334.8	551.8	144
Q_2	335.8	0	305.7	461	0	456.8	927	0	563.6	1829	0	756.2	3202	0	1158.8	5029
Q_3	474.8	0	440.5	396	0	578.2	740	0	782.2	1560	0	1624.8	2729	0	2775	4270
Q_4	225.9	71.2	0	10	110.4	0	13	212.8	0	23	336.4	0	35	423	0	52
Q_5	224.4	74.1	0	1	115.2	0	1	218	0	1	342	0	1	441.4	0	1
Q_6	240.6	96.7	103.8	10	147.4	128.4	13	227.4	139.4	23	344.4	173	35	485.5	180.4	52
Q_7	538.8	84.8	97.8	1	119.8	133.4	1	219.2	179.6	1	343.8	193.8	1	473.2	208	1
Q_8	347	228.6	96.6	17	456.6	113	44	927	125.2	79	1277.4	132.4	131	2593.4	196.8	226
Q_9	317.8	136.8	163	3	244.2	166.5	3	385.6	168.6	4	671.2	180	5	1245.8	252	7

As concerns expressiveness complexity, we observe that: (i) the retrieval times of queries classified in items A and B increase with the query expressiveness; (ii) queries classified in item C, for which an execution of *Strict Match* preliminary to *Soft Match* is needed, show times for *Soft Match* notably reduced, so confirming the theoretical complexity results. Moreover, for a number of retrieved profiles greater than 3000 (see t_{sf} in Q_2 and Q_3 on DS_4, DS_5), the more the data set is large, the more the expressiveness of soft requirements impacts on retrieval times. Noteworthy, the real query Q_1 generates, for each dataset, retrieval times comparable to all queries classified in item C, considering also the $\#p$ value. As a general remark, the query expressiveness does not significantly affect retrieval times in the whole matchmaking process involving both strict requirements and preferences.

Finally, for giving the reader a hint on the effectiveness of the presented approach in real enterprise scenario, where commercial DBMS are often adopted, we also conducted some tests on Oracle XE. We compared retrieval times obtained by executing the same queries on both PostgreSQL database and Oracle XE instance. Results confirm the performance limits of exploiting open-source DBMS, revealing retrieval times reduction of 90 percent for the execution over Oracle XE vs. PostgreSQL for a dataset of 50000 profiles.

5.2 Team Composition

Coherently with the above evaluation guidelines, we first introduce the datasets and the queries adopted for the evaluation of *Team Composition* process. As concerns the first design choice, we here use the same datasets DS_1, DS_2, DS_3 as in Section 5.1. The cardinality of such datasets (500, 1000 and 2000 profiles, respectively) may be considered comparable to the size of the real-world companies in the need for automatically composing multidisciplinary teams.

The queries adopted to evaluate how much the *Matchmaking* process affects the retrieval times of *Team Composition* service have been designed as described below. We recall that each project activity description, PA_i, is composed by three entities: (i) a description K_i of the knowledge required for the task (K_i is described, according to Definition 2, as conjunction of features of the form $\exists R_j^0.C$, where $R_j^0 = hasKnowledge$); (ii) a set of temporal constraints D_i and (iii) a number of required team members m_i. In particular, each query is composed by combining three heterogeneous project manager requests (PA_i, with $i \in \{1, 2, 3\}$), which differ from each other in the level of expressiveness. Namely, the categories of project activities in the following have been conceived:

- PA_1 involves only rather generic knowledge (*e.g.*, *DBMS*) in its component K_1;
- PA_2 involves in its component K_2 also features with an higher specificity (*e.g.*, *PostgreSQL*) than PA_1;
- PA_3 involves in its component K_3 very specific knowledge, described by the most specific concepts in the Skill Ontology taxonomy.

For each PA_i we always ask for three team members ($m_i = 3$) and set *Start date* and *End Date* as in the project description of Section 4.2.

Table 2 `I.M.P.A.K.T.` Team Composition times (in ms).

		No. profiles								
		500	**1000**	**2000**	**500**	**1000**	**2000**	**500**	**1000**	**2000**
		P1			**P2**			**P3**		
PA_1	t_{st}	35	102	114	0	0	0	38	85	122
	t_{sf}	0	0	0	516	898	1869	0	0	0
	t_{ca}	*75*	*77*	*81*	*90*	*98*	*91*	*73*	*74*	*80*
	$\#n$	10	33	56	24	41	84	10	33	56
PA_2	t_{st}	62	79	148	0	0	0	36	64	104
	t_{sf}	0	0	0	217	352	809	236	172	355
	t_{ca}	*76*	*81*	*79*	*79*	*80*	*90*	*75*	*78*	*90*
	$\#n$	53	104	196	93	187	377	71	146	270
PA_3	t_{st}	29	57	96	0	0	0	0	0	0
	t_{sf}	0	0	0	158	204	285	192	203	244
	t_{ca}	*48*	*60*	*69*	*78*	*71*	*89*	*71*	*85*	*83*
	$\#n$	2	8	22	11	23	59	11	23	59
P	$\boldsymbol{t_{DB}}$	*199*	*218*	*229*	*247*	*249*	*270*	*219*	*237*	*253*
	$\boldsymbol{t_{CSP}}$	*661*	*1043*	*1156*	*1260*	*2017*	*1601*	*1631*	*1284*	*1257*
	$\#sol$	4	11	19	14	20	54	12	16	23
	$\boldsymbol{t_{tc}}$	***860***	***1261***	***1385***	***1507***	***2266***	***1871***	***1850***	***1521***	***1510***

Each query asks for three project activities, such that each activity belongs to one category above. Moreover, we identified three classes of queries, P_i, with $i \in \{1,2,3\}$, such that:

- P_1 is a class of queries including only strict requirements and project activities made up by at least three conjuncts in K_i;
- P_2 is a class of queries rewriting the same queries as in P_1 by managing all features as preferences;
- P_3 is a class of queries combining both strict requirements and preferences and including project activities made up by at least five conjuncts in K_i.

Times resulting from queries execution are reported in Table 2. In order to better investigate on the *Team Composition* performance, for each project activity PA_i we show retrieval times w.r.t to the service steps recalled in Section 4.2. In particular, in Table 2 , we denote by $t_{st}$the time for Strict Matching, by t_{sf} the time for Soft Matching, by t_{ca} the time for checking temporal constraints and by t_{CSP} the time for executing CSP solver. Moreover, we refer to the number of profiles assigned to each activity PA_i by $\#n$ and to the number of project teams returned by the CSP solver by $\#sol$.

We recall that, by construction, *Strict* and *Soft Match* are automatically performed during the composition phase of each project activity. Then, the availability check is executed for each profile returned by the matching process. Eventually, all candidate profiles are combined in different possible team solutions through the CSP solver. According to this, we here do not comment on retrieval times for the matchmaking process which, by the way, confirm results shown in Section 5.1.

The time for team composition is therefore formally defined as follows: $t_{tc} = t_{DB} + t_{CSP}$, where $t_{DB} = \sum_{i=1}^{3} t_{ca_{PA_i}}$ (intuitively, $t_{ca_{PA_i}}$ is the time need for checking availability of candidate profiles matching component K_i of PA_i). In particular, t_{DB} measures the time for checking temporal constraints for retrieved candidates through SQL queries executed on the PostgreSQL database, whereas t_{CSP} measures computation time for the adopted CSP solver.

As shown in Table 2, the time for checking availability seems not to be much affected by the number of profiles to be analyzed. On the other hand, the number of possible final solutions affects the computation time for the CSP solver execution, that represents also the most time consuming phase—if compared to both skill match and availability checks phases. We adopted a simple Java API for CSP solution, because the proposed experimental evaluation is preliminary and we are interested in proving only the feasibility of the team composition. Performed tests suggest instead to search for more suitable APIs for CSP computation, able to implement advanced mechanisms for CSP problem optimization.

5.3 Core Competence Identification

As in the previous subsections, we are here interested in the evaluation of *data complexity* and *expression complexity* of our knowledge compilation approach to Core Competence extraction. To this aim, we first selected the most suitable datasets to perform the following two test campaigns: 1) the performance of our implementation is compared to a fully logic-based one ([17]) and expression complexity is evaluated; 2) data complexity of our approach is evaluated.

A different pair of datasets is adopted in the two test categories. More specifically, the first category works on two datasets, DS_1 and DS_2 such that DS_2 is more specific than DS_1, *i.e.*, it is characterized—for the same number of profiles—by a bigger set of resulting profile concept components. The second test category adopts instead other two datasets, DS_3 and DS_4, also such that DS_4 is more specific than DS_3.

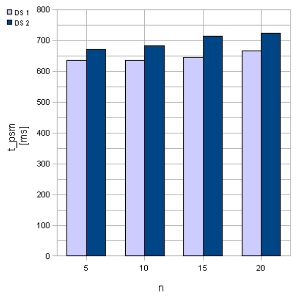
Moreover, in generating the datasets, profiles features have been set by taking into account only CV information related to technical knowledge (*i.e.*, profiles include only conjuncts of the form $\exists R_j^0.C$, where the entry point $R_j^0 = hasKnowledge$).

The first test campaign is aimed at evaluating expressiveness complexity. In the evaluation of execution times, we considered that Core Competence Identification is made up by two main extraction steps: the *Profile Subsumers Matrix* (PSM) computation, and the *Common Subsumer Enumeration* (CSE) algorithm execution (we recall that such algorithm takes the PSM as input). From now on, we call t_{psm} the average time for computing a PSM, t_{cse} the time for performing CSE algorithm and n the number of profiles.

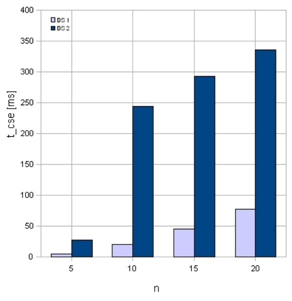
Figure 13 and Figure 14 show the performance results with reference to subsets of 5, 10, 15 and 20 profiles in DS_1 and DS_2 (such subsets are characterized by the same cardinality as those evaluated in the fully logic-based solution [17]).

Adopting subsets of different cardinality allows for investigating on the impact of the number of analyzed profiles on execution time. Figure 13 shows, in fact, t_{psm} (Figure 13(a)) and t_{cse} (Figure 13(b)), both in milliseconds, vs. n.

We recall that the computation of PSM asks for the reduction of the input profiles in so-called *Profile Concept Components* [18]. Intuitively, the more specific is the profile, the bigger is the number of components making it up. This motivates the adoptions of data sets of different specificity: DS_1 and DS_2: the objective of evaluation is investigating the relation between the number of profile concept components resulting from the data set and the execution time, when the number of profiles is given. Figure 14 presents, in fact, t_{psm} and t_{cse} vs. the profile concept components number. We notice that both Figure 14(a) and Figure 14(b) refer to the profile concept components deriving from DS_1 and DS_2: for each value of n, the smaller computation time value refers to DS_1 and the bigger to DS_2. In both experiments, $k = 0.3 \times n$.

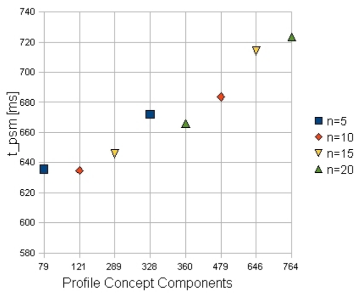


(a)

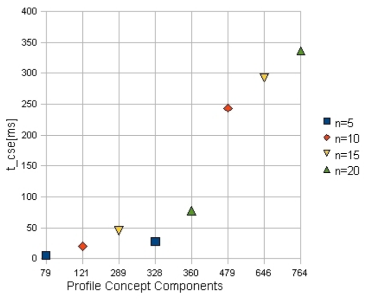


(b)

Fig. 13 PSM (a) and CSE (b) computation time vs. number of profiles. Times are expressed in milliseconds.



(a)



(b)

Fig. 14 PSM (a) and CSE(b) computation time vs. profile concept components

By looking at results, it can be noticed that the matrix creation is the most computationally expensive process: t_{psm} is in general bigger than t_{cse}. Moreover, the number of profiles affects more the common subsumer enumeration process, than profile subsumers matrix computation.

It is worth noticing how the adopted knowledge compilation approach dramatically improves process performance w.r.t. the fully logic-based one [17]: as an example, matrix computation time for 20 profiles has changed from 180 seconds to about 660 milliseconds. Such a significant improvement encourages the adoption of the approach in large real-world scenarios.

The second test campaign is aimed at evaluating the data complexity of the approach.

The KB instances generator was adopted to randomly create the data set DS_3 of 500 profiles (each with only 3 technical skills) and to extend it to 1000 and 2000 profiles. DS_4 was analogously generated, starting from 500 profiles, including a higher average number of technical skills (30 instead of 3). Intuitively, the resulting profile concept components number increases (in DS_3 components arise up to 11643 for 2000 profiles, while in DS_4 to 28177). DS_3 has been also extended to 1000 and 2000 profiles. The execution time for the two main steps of Core Competence extraction process are shown in Table 3, w.r.t. DS_3 and DS_4 and $k = 0.3 \times n$.

		Datasets Cardinality		
		500	1000	2000
t_{psm}	DS_3	0.87	1.1	1.74
	DS_4	3.91	9.24	27.29
t_{cse}	DS_3	0.21	0.46	1.4
	DS_4	66.06	235.81	912.57

Table 3 Core Competence extraction times (in seconds)

With reference to DS_3, the Profile Subsumers Matrix creation is still the most computationally expensive process. Conversely, the Core Competence Enumeration execution time dramatically raises in presence of significantly complex profiles—and consequently of a huge number of deriving concept components (see values of t_{cse} referred to DS_4 and Figure 14(b)). Performed tests suggest that there should be a critical value of the number of concept components, after which the most time-consuming phase switches from the matrix computation to the common subsumers sets identification.

6 Conclusions

Motivated by the need of efficiently managing large quantities of information in a human resources management system while still benefiting from novel non-standard reasoning services typical of knowledge representation and reasoning, we introduced a knowledge compilation approach in an originally designed relational schema, and devised solutions to execute inference services—both standard and non-standard ones—through standard-SQL queries only. We exploited such services referring to three relevant business processes typical of recruitment and human resources management, presenting them in the framework of the `I.M.P.A.K.T.` system. We reported an effective comparison with existing tools and research solutions and showed the effectiveness of our approach also from a computational point of view. Implementation of optimization techniques, such as table partitioning in our PostgreSQL database, are under development. As expected, first results show an improvement in the `I.M.P.A.K.T.` performance (*e.g.*, for the skill matching execution over a dataset of 10000 profiles, we obtain a reduction of 30 percent on the retrieval time). Moreover, we are currently studying the peculiarities of the proposed design method for database modeling and management, with the aim of generalizing it to a framework fully independent from the underlying ontology.

Future work aims at testing further devised strategies for score calculation and at designing a service for CV translation from plain text according to our skill ontology. Moreover, in order to deal with specific business application requirements, *e.g.*, the need to deploy `I.M.P.A.K.T.` in a more scalable environment, we are investigating the possibility of exploiting Big Data technologies for KB modeling and querying.

References

1. Owl 2 web ontology language structural specification and functional-style syntax (2009). Www.w3.org/TR/2009/REC-owl2-syntax-20091027/
2. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook – 2nd edition. Cambridge University Press (2007)
3. Barney, J.B.: Strategic Factor Markets: Expectations, Luck and Business Strategy. Management Science **32**, 1231–1241 (1986)
4. Barney, J.B.: Firm Resources and Sustained Competitive Advantage. Journal of Management **17**(1), 99–120 (1991)
5. Bechhofer, S., Horrocks, I., Turi, D.: The OWL instance store: System description. In: Proceedings of the 20th International Conference on Automated Deduction(CADE '05). Tallinn, Estonia (2005)
6. Biesalski, E., Abecker, A.: Similarity measures for skill-profile matching in enterprise knowledge management. In: Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration (ICEIS 2006), Paphos, Cyprus, pp. 11–16 (2006)
7. Bogner, W., H.Thomas: Core competencies and competitive advantage: a model and illustrative evidence from the pharmaceutical industry. In: G. Hamel, A. Heene (eds.) Competencies-based Competition, pp. 111–144. Wiley, New York (1994)
8. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: The First International Semantic Web Conference (ISWC '02), pp. 54–68. Springer-Verlag (2002)
9. C. Li, K. C.-C. Chang, Ihab F. Ilyas and S. Song: RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In: ACM SIGMOD, pp. 131–142. ACM (2005)
10. Cadoli, M., Donini, F.M.: A survey on knowledge compilation. AI Commun. **10**(3-4), 137–150 (1997)
11. Carloni, O., Leclère, M., Mugnier, M.L.: Introducing reasoning into an industrial knowledge management tool. Applied Intelligence **31**(3), 211–224 (2007). DOI 10.1007/s10489-007-0103-x. URL `http://dx.doi.org/10.1007/s10489-007-0103-x`
12. Chomicki, J.: Querying with Intrinsic Preferences. In: Advances in Database Technology - EDBT 2002, pp. 34–51. Springer-Verlag (2002)
13. Chong, E.I., Das, S., Eadon, G., Srinivasan, J.: An Efficient SQL-based RDF Querying Scheme. In: The 31th International Conference on Very Large Data Bases (VLDB'05), pp. 1216–1227. VLDB Endowment (2005)
14. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: Semantic-based skill management for automated task assignment and courseware composition. Journal of Universal Computer Science **13**(9), 1184–1212 (2007)
15. Colucci, S., Di Sciascio, E., Donini, F.M.: A knowledge-based solution for core competence evaluation in human-capital intensive companies. In: Proceedings of 8th International Conference on Knowledge Management(I-KNOW-08), pp. 259–266 (2008)
16. Colucci, S., Di Sciascio, E., Donini, F.M., Tinelli, E.: Finding informative commonalities in concept collections. In: CIKM 2008, Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA, October 26-30, pp. 807–817. ACM (2008)
17. Colucci, S., Tinelli, E., Donini, F.M., Di Sciascio, E.: Automating competence management through non-standard reasoning. Engineering Applications of Artificial Intelligence (2011). Doi:10.1016/j.engappai.2011.05.015
18. Colucci, S., Tinelli, E., Giannini, S., Di Sciascio, E., Donini, F.M.: Knowledge compilation for core competence extraction in organizations. In: 16th International Conference on Business Information Systems, Lecture Notes in Business Information Processing. Springer (2013)

19. Delaitre, V., Kazakov, Y.: Classifying ELH ontologies in SQL databases. In: OWL: Experiences and Directions 2009 (OWLED 2009), *CEUR Workshop Proceedings*, vol. 529. CEUR-WS.org, Chantilly, VA, United States (2009)
20. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007) (2007)
21. Draganidis, F., Mentzas, G.: Competency based management: A review of systems and approaches. Information Management and Computer Security **14**(1), 51–64 (2006)
22. Ehrig, M., Haase, P., Stojanovic, N., Hefke, M.: Similarity for ontologies — a comprehensive framework. In: D. Bartman, F. Rajola, J. Kallinikos, D. Avison, R. Winter, P. Ein-Dor, J. Becker, F. Bodendorf, C. Weinhardt (eds.) Information Systems in a Rapidly Changing Economy: Proceedings of the 13th European Conference on Information Systems (ECIS 2005), May 26-28, 2005, Regensburg, Germany (2005)
23. Fazel-Zarandi, M., Fox, M.S.: Semantic matchmaking for job recruitment: an ontology-based hybrid approach. In: Proceedings of the 8th International Semantic Web Conference (2009)
24. Hafeez, K., Zhang, Y., Malak, N.: Core competence for sustainable competitive advantage: a structured methodology for identifying core competence. Engineering Management, IEEE Transactions on **49**(1), 28 –35 (2002)
25. Hamel, G., Prahalad, C.K.: The core competence of the corporation. Harvard Business Review **May-June**, 79–91 (1990)
26. Hefke, M., Stojanovic, L.: An ontology-based approach for competence bundling and composition of ad-hoc teams in an organization. In: Proceedings of 4th International Conference on Knowledge Management (2004)
27. Holsapple, C.W., Joshi, K.D.: A formal knowledge management ontology: Conduct, activities, resources, and influences: Research articles. Journal of the American Society for Information Science and Technology **55**(7), 593–612 (2004)
28. Kadiri, S.E., Grabot, B., Thoben, K.D., Hribernik, K., Emmanouilidis, C., von Cieminski, G., Kiritsis, D.: Current trends on {ICT} technologies for enterprise information systems. Computers in Industry **79**, 14 – 33 (2016). DOI http://dx.doi.org/10.1016/j.compind.2015.06.008. URL `http://www.sciencedirect.com/science/article/pii/S0166361515300142`. Special Issue on Future Perspectives On Next Generation Enterprise Information Systems
29. Karanikola, L., Karali, I.: A fuzzy logic approach for reasoning under uncertainty and vagueness - a matchmaking case study. In: 2016 2nd International Conference on Information Management (ICIM), pp. 52–56 (2016). DOI 10.1109/INFOMAN.2016.7477533
30. Kessler, R., Bchet, N., Roche, M., Torres-Moreno, J.M., El-Bze, M.: A hybrid approach to managing job offers and candidates. Information Processing & Management **48**(6), 1124 – 1135 (2012). DOI http://dx.doi.org/10.1016/j.ipm.2012.03.002
31. Khilwani, N., Harding, J.A.: Managing corporate memory on the semantic web. Journal of Intelligent Manufacturing **27**(1), 101–118 (2016). DOI 10.1007/s10845-013-0865-4. URL `http://dx.doi.org/10.1007/s10845-013-0865-4`
32. Kießling, W.: Foundations of Preferences in Database Systems. In: The 28th international conference on Very Large Data Bases (VLDB'02), pp. 311–322. Morgan Kaufmann, Los Altos (2002)
33. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM - A Pragmatic Semantic Repository for OWL. In: WISE, vol. 3807, pp. 182–192. Springer (2005)
34. L. Halawi, J.A., McCarthy, R.: Resource-Based View of Knowledge Management for Competitive Advantage. The Electronic Journal of Knowledge Management **3**(2), 75–86 (2005)
35. Markides, C.C., Williamson, P.J.: Related diversification, core competences and corporate performance. Strategic Management Journal **15**, 49–65 (1994)
36. Martinez-Gil, J., Paoletti, A.L., Schewe, K.D.: A smart approach for matching, learning and querying information from the human resources domain. In: M. Ivanović, B. Thalheim, B. Catania, K.D. Schewe, M. Kirikova, P. Šaloun, A. Dahanayake, T. Cerquitelli, E. Baralis, P. Michiardi (eds.) New Trends in Databases and Information Systems: ADBIS 2016 Short Papers and Workshops, BigDap, DCSA, DC, Prague, Czech Republic, August 28-31, 2016, Proceedings, pp. 157–167. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-44066-8_17. URL `http://dx.doi.org/10.1007/978-3-319-44066-8_17`
37. Meso, P., Smith, R.: A Resource-Based View of Organizational Knowledge Management Systems. Journal of Knowledge Management **4**(3), 224–231 (2000)
38. Mochol, M., Wache, H., Nixon, L.: Improving the accuracy of job search with semantic techniques. In: W. Abramowicz (ed.) Business Information Systems: 10th International Conference, BIS 2007, Poznan, Poland, April 25-27, 2007. Proceedings, pp. 301–313.

Springer Berlin Heidelberg, Berlin, Heidelberg (2007). DOI 10.1007/978-3-540-72035-5_23. URL `http://dx.doi.org/10.1007/978-3-540-72035-5_23`
39. Nelson, R.R.: Why do firms differ, and how does it matter? Strategic Management Journal **12**, 61–74 (1991)
40. Neshati, M., Beigy, H., Hiemstra, D.: Expert group formation using facility location analysis. Information Processing & Management **50**(2), 361 – 383 (2014). DOI http://dx.doi.org/10.1016/j.ipm.2013.10.001
41. O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., Musen, M.: Supporting rule system interoperability on the semantic web with swrl. In: Y. Gil, E. Motta, V.R. Benjamins, M.A. Musen (eds.) The Semantic Web – ISWC 2005: 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005. Proceedings, pp. 974–986. Springer Berlin Heidelberg, Berlin, Heidelberg (2005). DOI 10.1007/11574620_69. URL `http://dx.doi.org/10.1007/11574620_69`
42. OLeary, D.: Enterprise knowledge management. IEEE Computer **31**(3), 54–61 (1998)
43. OLeary, D.: Using AI in knowledge management: Knowledge bases and ontologies. IEEE Intelligent Systems **13**(3), 34–39 (1998)
44. P. Bosc and O. Pivert: SQLf: a relational database language for fuzzy querying. IEEE Transactions on Fuzzy Systems **3**(1), 1–17 (1995)
45. Pan, Z., Heflin, J.: DLDB: Extending Relational Databases to Support Semantic Web Queries. In: The First International Workshop on Practical and Scalable Semantic Systems (PSSS1), vol. 89, pp. 109–113. CEUR Workshop Proceedings (2003)
46. Rácz, G., Sali, A., Schewe, K.D.: Semantic matching strategies for job recruitment: A comparison of new and known approaches. In: International Symposium on Foundations of Information and Knowledge Systems, pp. 149–168. Springer (2016)
47. Sanchez, R., Heene, A.: Reinventing strategic management: New theory and practice for competence-based competition. European Management Journal **15**(3), 303 – 317 (1997). DOI DOI:10.1016/S0263-2373(97)00010-8. URL `http://www.sciencedirect.com/science/article/B6V9T-3SWXWWW-9/2/e4ae2d056e3ef23222e396122a08559b`
48. Staab, S., Schnurr, H., Studer, R., Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems **16**(1) (2001)
49. Stojanovic, N., Studer, R., Stojanovic, L.: An approach for the ranking of query results in the semantic web. In: Proceedings of the Second International Semantic Web Conference (2003)
50. Sure, Y., Maedche, A., Staab, S.: Leveraging corporate skill knowledge - from proper to ontoproper. In: Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (2000)
51. Tampoe, M.: Exploiting the core competences of your organization. Long Range Planning **27**(4), 66 – 77 (1994). DOI DOI:10.1016/0024-6301(94)90057-4. URL `http://www.sciencedirect.com/science/article/B6V6K-45K4F3J-7/2/bbb97f402644fd446a6dfb0cb7a3db61`
52. Teece, D.J., Pisano, G., Shuen, A.: Dynamic capabilities and strategic management. Strategic Management Journal **18**(7), 509–533 (1997)
53. Tinelli, E., Cascone, A., Ruta, M., Di Noia, T., Di Sciascio, E., Donini, F.M.: I.M.P.A.K.T.: An Innovative Semantic-based Skill Management System Exploiting Standard SQL. In: Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009), pp. 224–229. Milan, Italy, May 6-10 (2009)
54. Tinelli, E., Colucci, S., Di Sciascio, E., Donini, F.M.: Knowledge compilation for automated team composition exploiting standard sql. In: Proceedings of the 27th Annual ACM (SIGAPP) Symposium on Applied Computing. ACM Press (2012)
55. Tinelli, E., Colucci, S., Giannini, S., Di Sciascio, E., Donini, F.M.: Large scale skill matching through knowledge compilation. In: 20th International Symposium on Methodologies for Intelligent Systems (ISMIS12), Lecture Notes on Computer Science. Springer (2012)
56. Tinelli, E., Donini, F.M., Sciascio, E.D.: Compiling subsumption to relational databases. Intelligenza Artificiale **7**(1), 19–29 (2013)
57. Trastour, D., Bartolini, C., Priest, C.: Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In: Proceedings of the Eleventh International World Wide Web Conference, pp. 89–98. ACM (2002)
58. Tsang, E.P.K.: Foundations of constraint satisfaction. Computation in cognitive science. Academic Press (1993)
59. Vardi, M.: The complexity of relational query languages. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing (STOC '82), pp. 137–146. ACM New York, NY, USA (1982)

60. Vas, R.: Studio: Ontology-centric knowledge-based system. In: A. Gábor, A. Kő (eds.) Corporate Knowledge Discovery and Organizational Learning: The Role, Importance, and Application of Semantic Business Process Management, pp. 83–103. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-28917-5_4. URL `http://dx.doi.org/10.1007/978-3-319-28917-5_4`
61. Wernerfelt, B.: A Resource-Based View of the Firm. Strategic Management Journal **5**(2), 171–180 (1984)
62. Wernerfelt, B.: The Resource-Based View of the Firm: Ten Years After. Strategic Management Journal **16**(3), 171–174 (1995)
63. Wilkinson, K., Sayers, C., Kuno, H.A., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. In: The first International Workshop on Semantic Web and Databases (SWDB'03), pp. 131–150 (2003)
64. Zhang, F., Ma, Z.M.: Representing and reasoning about xml with ontologies. Applied Intelligence **40**(1), 74–106 (2014). DOI 10.1007/s10489-013-0446-4. URL `http://dx.doi.org/10.1007/s10489-013-0446-4`

A Basic Description Logics

Description Logics are a family of formalisms and reasoning services widely employed for knowledge representation, in a decidable fragment of First Order Logic. We give here a limited introduction to make this paper self-contained, referring the interested reader to more comprehensive introductions [2, Ch.2].

The alphabet of each DL is made up by unary and binary predicates, known as **Concept Names** $A_1, A_2, A_3, \ldots$ and **Role Names** $r_1, r_2, r_3, \ldots$, respectively. Complex **Concept Descriptions**—which we denote with the symbols C, D—are built (recursively) from concept and role names composed by *constructors* like, for example, conjunction of concepts $A_1 \sqcap A_2$, minimum number of role fillers $(\geqslant\ n\ r)$, and many others. Intuitively, concepts represent classes of individuals of the domain of interest, and roles represent binary relations between them. Each choice of constructors defines a different DL, and characterizes such DL both in terms of expressiveness and computational complexity of reasoning tasks. In fact, it is well established that the more a DL is expressive, the harder is inferring new knowledge from its descriptions [2, Ch.3].

The expressiveness of a DL may be also enriched by the introduction of *concrete features* $f_1, f_2, f_3, \ldots$, which are binary predicates whose second argument belongs to a *concrete domain* D (*e.g.*, integers, reals, strings, dates). Each domain comes along its set of unary predicates $p_1, p_2, p_3, \ldots$, and new classes can be constructed by requiring that an individual satisfying a predicate—for instance, a feature `years` representing years of experience of an individual, could be used with a predicate $=_3$ to form the class $=_3$ (`years`) of individuals having exactly three years of experience. There is also the possibility of having n-ary predicates over n concrete domains, but we are not going to use them here. Given a DL $\mathcal{L}$, its enrichment with concrete features is usually denoted by $\mathcal{L}$(D).

The semantics of concept descriptions is conveyed through an **Interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of $\mathcal{I}$—a nonempty set—and $\cdot^{\mathcal{I}}$ is an *interpretation function* such that, conforming to the above intuition about concepts and roles,

- $\cdot^{\mathcal{I}}$ maps each concept name A in a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- $\cdot^{\mathcal{I}}$ maps each role name r in a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- if concrete features with some domain D are used, $\cdot^{\mathcal{I}}$ maps each feature name f in a binary relation $f^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathrm{D}$, and each predicate p to a subset $p^{\mathcal{I}} \subseteq \mathrm{D}$.

The DL constructors we use or mention in this paper, along with their semantics, are shown in Table 4. In the last four columns, an "x" in the cell of row c (the constructor) and column $\mathcal{L}$ (the DL) means that c is used in $\mathcal{L}$, except for the last row that names $\mathcal{L}$ with concrete domain D as $\mathcal{L}$(D).

Statements about classes in the domain of interest are divided into **Concept Definitions** and **Concept Inclusions**. Definitions (denoted by $A \equiv C$) state—in the form of a complex concept C—the necessary and sufficient conditions for an individual to belong to the concept A. For instance, $A_3 \equiv A_1 \sqcap A_2$ states that an individual belongs to A_3 *if and only if* it belongs to both A_1 and A_2. Inclusions (denoted by $A \sqsubseteq C$) state in C only the necessary conditions for membership in A. For instance, $A_4 \sqsubseteq A_5$ states that an individual belongs to A_4 *only if* it belongs to A_5. Each concept name A can appear on the left-hand side of at most one of such definitions or inclusions—if any. Concept names are divided into *Defined Concepts*, appearing on the left-hand side of some concept definition, and *Primitive Concepts*, which do not appear on the left-hand side of any definition (but can appear on the left-hand side of an inclusion). Intuitively, an individual belongs to a primitive concept A only if this membership is explicitly stated (we define later on how this can be done), while membership can be implicit for defined concepts (and reasoning can be necessary to derive it).

The set of inclusions and definitions yield a formal representation of the intensional knowledge of the domain of interest, known as **TBox** in DL systems, and **Ontology** in the generic knowledge representation framework. TBoxes containing recursive concept definitions are called *cyclic* (*acyclic* otherwise). In this paper we use only acyclic TBoxes.

An interpretation $\mathcal{I}$ is a **model** for a TBox $\mathcal{T}$ if it satisfies all concept definitions and inclusions in $\mathcal{T}$.

A DL system usually allows one to make statements about named individuals $a_1, a_2, a_3, \ldots$. This part of a DL-knowledge base is known as **ABox**, and statements have one of the following two forms:

- **Concept assertions**: $C(a)$ states that an individual a belongs to the concept C

Table 4 DLs Set of Adopted Constructors

Constructor Name	**Syntax**	**Semantics**	$\mathcal{SHIN}$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{FL}_0$
top-concept	$\top$	$\Delta^{\mathcal{I}}$	x	x	x	x
bottom-concept	$\bot$	$\emptyset$	x	x	x	x
full negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$	x		x	
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$	x	x	x	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x	x	x
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	x		x	
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \vert \forall y : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	x	x	x	x
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \vert \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	x	x	x	
at-least restriction	$(\geqslant\ n\ r)$	$\{x \in \Delta^{\mathcal{I}} \vert \sharp\{y \in \Delta^{\mathcal{I}} \vert (x, y) \in r^{\mathcal{I}}\} \geqslant n\}$	x			
at-most restriction	$(\leqslant\ m\ r)$	$\{x \in \Delta^{\mathcal{I}} \vert \sharp\{y \in \Delta^{\mathcal{I}} \vert (x, y) \in r^{\mathcal{I}}\} \leqslant m\}$	x			
concrete feature	$p(f)$	$\{x \in \Delta^{\mathcal{I}} \vert \exists y : (x, y) \in f^{\mathcal{I}} \wedge y \in p^{\mathcal{I}}\}$	$\mathcal{SHIN}$(D)	$\mathcal{ALE}$(D)	$\mathcal{ALC}$(D)	$\mathcal{FL}_0$(D)

- **Role assertions**: $r(a, b)$ states that individual a relates to the individual b through role r.

An interpretation $\mathcal{I}$ assigns an element $a^{\mathcal{I}} \in \Delta$ to each individual a, and is a **model** for an ABox $\mathcal{A}$ if it satisfies $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all role assertions $r(a, b) \in \mathcal{A}$ and $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all concept assertions $C(a) \in \mathcal{A}$.

Table 5 The expressiveness of $\mathcal{FL}_0$(D) (adopted by `I.M.P.A.K.T.`) explained with examples.

Expression	**Example**	**Intuitive Explanation**
top-concept	$\top$	whole domain
bottom-concept	$\bot$	empty set
conjunction	$\texttt{Java} \sqcap \forall\texttt{skillType.Programming}$	knowledge about Java and programming experience
value restriction	$\forall\texttt{knowsLanguage.English}$	elements of the domain knowing only English as foreign language
concrete features	$=_3$ (`years`)	elements of the domain endowed with exactly 3 years of working experience
concept definition	$\texttt{EnglishSpeaker} \equiv \forall\texttt{knowsLanguage.English} \sqcap =_3 (\texttt{verbalLevel})$	represents someone skilled at English conversation
concept inclusion	$\texttt{Java} \sqsubseteq \texttt{OOP}$	Java knowledge is more specific than Object Oriented Programming knowledge

`I.M.P.A.K.T.` adopts a CV representation (see Definition 2) allowing for reasoning only on $\mathcal{FL}_0$(D) concepts which represent knowledge about our domain.The full expressiveness of the adopted $\mathcal{FL}_0$(D) subset is explained, with the aid of constructors usage examples, in Table 5.

A.1 Standard Inference Services

The most important service characterizing reasoning in DL checks for specificity hierarchies, by determining whether a concept description is more specific than another one or, formally, if there is a *subsumption* relation between them.

Definition 4 (Subsumption) Given two concept descriptions C and D and a TBox $\mathcal{T}$in a DL $\mathcal{L}$, we say that D subsumes C w.r.t. $\mathcal{T}$if for every model of $\mathcal{T}$, $C^{\mathcal{I}} \subset D^{\mathcal{I}}$. We write $C \sqsubseteq_{\mathcal{T}} D$, or simply $C \sqsubseteq D$ if we assume an empty TBox.

For example, consider the following concept descriptions, referred to a required task and a personnel profile, respectively:

– $T_1 = \forall\texttt{hasKnowledge.ProgrammingLanguage} \sqcap \geqslant_3 (\texttt{years})$
– $P_1 = \forall\texttt{hasKnowledge.Java} \sqcap =_5 (\texttt{years}) \sqcap \forall\texttt{hasLevel.ComputerScience}$

Considering a TBox with the two following concept inclusions $\texttt{Java} \sqsubseteq \texttt{OOP}$ and $\texttt{OOP} \sqsubseteq \texttt{ProgrammingLanguage}$, knowledge expressed by P_1 is more specific than the one required by T_1: according to the previous definition T_1 subsumes P_1.

Several widely used services may be reduced to subsumption, like concept equivalence and concept satisfiability (intuitively, a concept description is satisfiable if it can be somehow interpreted in the knowledge domain).

A.2 Non-standard Inference Services

Although very useful in many knowledge management settings, both subsumption and satisfiability return a yes/no answer. The first category of services provided by `I.M.P.A.K.T.` is instead aimed at returning referral lists of job candidates, ranked according to their ability to fulfill the job request initiating the recruiting process. In such a scenario, both explanation and belief revision turn out to be useful to cope with cases in which no perfect match exists between job request and candidates. The process performed by `I.M.P.A.K.T.` to return referral lists of candidates conceptually originates from the extended matchmaking approach originally introduced in our past research [17], based on Concept Abduction and Concept Contraction.

Concept Contraction is useful when $C \sqcap D$ is unsatisfiable in the ontology $\mathcal{T}$, *i.e.*, the task and the profile are not compatible with each other. In this case, as in a belief revision process, we want to retract some requirements G (for *Give up*) in D, to obtain a new contracted task request K (for *Keep*) which is compatible with D. In other words, such that $K \sqcap D$ is satisfiable in $\mathcal{T}$.

Concept Abduction is instead useful when C and D are satisfiable w.r.t. each other (the task and the profile do not contain conflicting information) but subsumption does not hold (*i.e.*, a full match is unavailable). In this case the objective is hypothesizing some explanation on which are the causes of this result.

The third `I.M.P.A.K.T.` service category aims at determining the strategic competence of a company, denoted by Core Competence in knowledge management literature [25]. The objective of the implementation of services for automatic Core Competence extraction is identifying a common know-how in a significant portion of company personnel, with a degree of coverage to be set by the management. To this aim, `I.M.P.A.K.T.` framework follows the conceptual line by Colucci *et al* [16], based on Least Common Subsumer (LCS) computation.

B Theoretical Limits of Knowledge Compilation in SQL

The translation into SQL of a problem stated in Description Logics is subject to some theoretical limitations regarding efficiency. It is well known [59] that deciding whether a relational query Q over a database db retrieves an individual a—a problem that we denote by $db \models Q(a)$ in the following—is PSPACE-complete in *expression complexity* (fixed db, varying Q), and LOGSPACE-complete in *data complexity* (fixed Q, varying db). We now analyze the consequences of these facts for the size and complexity of our translation. We denote by $|db|$ the size of a database, which is proportional to the number of tuples assuming relations of bounded arity.

Given a CV C and a profile P expressed in some DL, the standard and non-standard services offered by our system imply—as a special case—deciding Subsumption between C and P, denoted by $C \sqsubseteq P$. Our approach translates:

– a CV C into a database, which we denote $\nu(C)$, with a special individual a representing the person having that CV, and
– a profile P into an SQL query $\pi(P)$.

Subsumption between C and P holds iff $\nu(C) \models \pi(P)(a)$, that is, iff a is retrieved from the database $\nu(C)$ by the query $\pi(P)$. Hence, subsumption in a given DL could be solved by first applying the translation, and then answering the corresponding query. Rephrasing the complexity results, for a fixed query $\pi(P)(a)$, the problem $\nu(C) \models \pi(P)(a)$ is solvable in LOGSPACE considering $|\nu(C)|$ as input.

Now let C, P be expressed in a DL whose subsumption problem $C \sqsubseteq P$ is EXPTIME-complete, such as $\mathcal{SHIN}(D)$, which is equivalent to `OWL1-DL`. We observe that $C \sqsubseteq P$ iff $(C \sqcap \neg P) \sqsubseteq \bot$, where $C \sqcap \neg P$ is a concept which is still in $\mathcal{SHIN}(D)$, and the same is true for every DL which is closed under concept negation. If transformation $\nu(\cdot)$ could be performed in polynomial time, then its output $\nu(C \sqcap \neg P)$ has size polynomial in $|C \sqcap \neg P|$. Since $\nu(C \sqcap \neg P) \models \pi(\bot)(a)$ can be decided in space logarithmic (and hence time polynomial) in $|\nu(C \sqcap \neg P)|$, then the EXPTIME-complete problem $C \sqsubseteq P$ could be solved in polynomial time by first transforming C, P into $\nu(C \sqcap \neg P)$, then $\bot$ into $\pi(\bot)$—a constant since $\bot$ is fixed—and then deciding $\nu(C \sqcap \neg P) \models \pi(\bot)(a)$. The same argument could be repeated for DLs in which $C \sqsubseteq \bot$ (concept satisfiability) is a problem in any complexity class $\mathcal{C}$ above PTIME, such as NP, or CO-NP, or PSPACE. We can conclude with the following theorem, whose proof is in the above argument.

Theorem 1 *Let $\mathcal{L}$ be a DL whose subsumption problem is complete for some complexity class $\mathcal{C}$, and such that either $\mathcal{L}$ is closed under concept negation, or $\mathcal{L}$ contains $\bot$. If the transformation $\nu(\cdot)$ could be computed in polynomial time, then $\mathcal{C} \subseteq$ PTIME.*

Recall that PTIME is provably strictly contained in EXPTIME, hence the above theorem in this case says—by contraposition—that $\nu(\cdot)$ cannot run in polynomial time at all, *e.g.*, for $\mathcal{L} = \mathcal{SHIN}(D)$. For $\mathcal{C}$ below EXPTIME and above PTIME, *e.g.*, $\mathcal{C} =$ NP, the claim is conditioned to $\mathcal{C} \subseteq$ PTIME, which is considered unlikely.

Regarding the transformation $\pi(\cdot)$, a similar argument could be developed. In fact, $C \sqsubseteq P$ iff $\top \sqsubseteq (\neg C \sqcup P)$, where $\neg C \sqcup P$ is a concept that still belongs to a DL which is at least as expressive as $\mathcal{ALC}$. So, one could decide $C \sqsubseteq P$ by first transforming $\top$ into $\nu(\top)$—some constant database—then transforming $\neg C \sqcup P$ into $\pi(\neg C \sqcup P)$, and then decide $\nu(\top) \models \pi(\neg C \sqcup P)(a)$. The latter problem can be solved in PSpace with respect to $|\pi(\neg C \sqcup P)|$. If $\pi(\cdot)$ could be performed in polynomial time, it would yield a query $\pi(\neg C \sqcup P)$ whose size is polynomial in $|\neg C \sqcup P|$. Overall, $C \sqsubseteq P$ would be a problem solvable in PSpace also with respect to the size of C and P.

Theorem 2 *Let $\mathcal{L}$ be a DL whose subsumption problem is complete for some complexity class $\mathcal{C}$, and such that $\mathcal{L}$ is closed under concept disjunction and negation. If the transformation $\pi(\cdot)$ could be computed in polynomial time, then $\mathcal{C} \subseteq$ PSpace.*

Hence, for languages like $\mathcal{SHIN}(D)$, finding a polynomial-time transformation $\pi(\cdot)$ would imply ExpTime $\subseteq$ PSpace, a statement that—although not yet disproved—is considered very unlikely in complexity theory.

We conclude that to look for polynomial-time transformations, one should limit the choice of the DL to those in which $C \sqsubseteq P$ is a problem in PTime (for polynomial-time $\nu(\cdot)$) or in PSpace (for polynomial-time $\pi(\cdot)$). It seems unreasonable to use different DLs for *curricula* and profiles, so the stronger PTime-condition dominates the choice. This motivates our choice of $\mathcal{FL}_0$ for expressing *curricula* and profiles, since $\mathcal{FL}_0$ is one of the DLs having a polynomial-time subsumption problem.

C Alphabet for Profile Definition

The complete list of possible conjuncts of a profile P is reported Table 6.

Notice that for each profile conjunct in which the feature *years* is defined, it is mandatory to specify also the feature *lastdate*, in order to be aware whether the experience level is up to date. Moreover, the proposed approach considers only features of the form $=_n p$ in the profile storage phase, whereas it manages features p in the form $\{\leqslant_n p, \geqslant_n p, =_n p\}$ in the reasoning phase: intuitively a candidate specifying her level of work experience sets the years to a natural number, while the management of information in the profiles may need the whole set of order relations.

Table 6 Skill Reference Template

Entry point(R_j^0)	**Category**	**Feature Description (DL syntax)**
hasLevel	Level	$\exists hasLevel.(Level \sqcap (\geqslant, \leqslant, =)_n mark)$
hasJobTitle	JobTitle	$\exists hasJobTitle.(jobTitle \sqcap (\geqslant, \leqslant, =)_n years \sqcap =_n lastdate)$
hasIndustry	Industry	$\exists hasIndustry.(Industry \sqcap (\geqslant, \leqslant, =)_n years \sqcap =_n lastdate)$
hasKnowledge	Knowledge	$\exists hasKnowledge.(Knowledge \sqcap \forall skillType.Type \sqcap (\geqslant, \leqslant, =)_n years \sqcap =_n lastdate)$
hasComplementarySkill	ComplementarySkill	$\exists hasComplementarySkill.(ComplementarySkill \sqcap (\geqslant, \leqslant, =)_n years \sqcap =_n lastdate)$
knowsLanguage	Language	$\exists knowsLanguage.(Language \sqcap =_n readingLevel \sqcap =_n verbalLevel \sqcap =_n writingLevel)$

D Candidate Profiles Example Set

1 - Mario Rossi

- *Level*: Computer Science Engineering (mark 110), Secondary School (mark 60), Master Degree
- *JobTitle*: Database Administrator(4 years), Project Manager (2 years)
- *Industry*: Banking (4 years), IT and Telematics Applications (2 years)

- *Knowledge*: Cplusplus (5 years), Java (5 years), Visual Basic(5 years)
- *ComplementarySkill*: Cooperation (5 years), LeaderShip (5 years)
- *Language*: English (excellent writing, verbal and reading), French (good writing)

2 - Daniela Bianchi
- *Level*: Computer Science Engineering (mark 110), Secondary School (mark 60), Bachelor
- *JobTitle*: Database administrator (4 years), Project Manager (2 years)
- *Industry*: Banking (4 years), IT and Telematics Applications (2 years)
- *Knowledge*: Cplusplus (2 years), Java (6 years), Visual Basic (1 years)
- *ComplementarySkill*: Cooperation (5 years), LeaderShip(5 years)
- *Language*: English (excellent verbal, writing and reading), French (good writing)

3 - Lucio Battista
- *Level*:Managerial Engineering (mark 104), Secondary School (mark 60), Master Degree, CCDP
- *JobTitle*: Database Administrator (4 years), Project Manager (2 years)
- *Industry*: Banking (4 years), IT and Telematics Applications (2 years)
- *Knowledge*: DBMS (2 years)
- *ComplementarySkill*: Cooperation (5 years), LeaderShip (5 years)
- *Language*: English (excellent verbal, writing and reading), French (good writing)

4 - Mariangela Porro
- *Level*: Managerial Engineering (mark 104), Secondary School (mark 60), Master Degree, Master after master
- *JobTitle*: Database Administrator (4 years), Network computer systems Administrator (4 years)
- *Industry*: Banking (4 years), IT and Telematics Applications (2 years)
- *Knowledge*: DBMS (2 years), Internet Technologies (2 years)
- *ComplementarySkill*: Learning Strategy (8 years)
- *Language*: English (good verbal, writing and reading)

5 - Nicola Marco
- *Level*: Electronics Engineering (mark 104), Bachelor, Master after Master
- *JobTitle*: Database Administrator (2 years), Network computer systems Administrator (2 years)
- *Industry*: Banking (4 years), IT and Telematics Applications (2 years)
- *Knowledge*: DBMS (5 years), Internet Technologies (5 years)
- *ComplementarySkill*: Learning Strategy (8 years)
- *Language*: English (good writing, verbal and reading)

6 - Carla Buono
- *Level*: Statistics (mark 106), Master Degree, Master after Master
- *JobTitle*: Cost Estimator (4 years), Budget Analysts (10 years)
- *Industry*: Banking (4 years), Business Strategic Management (2 years), Finance Banking (1 years)
- *Knowledge*: Sales and Marketing (2 years), Administration and Management (4 years), Mathematics (10 years)
- *ComplementarySkill*: Critical thinking (8 years), monitoring (8 years)
- *Language*: English (excellent writing, verbal and reading knowledge), French (good writing knowledge)

7 - Marcello Cannone
- *Level*: Managerial Engineering Degree (mark 106)
- *JobTitle*: Training and Development Manager (2 years)
- *Industry*: Sales, Banking and Consumer Lending
- *Knowledge*: Economics and Accounting (4 years), WorkflowManagement
- *ComplementarySkill*: Visualization, Spatial orientation, Verbal abilities
- *Language*: English, German (excellent writing and reading knowledge, basic verbal knowledge)

8 - Carmelo Piccolo
- *Level*: Mechanical Engineering (mark 79)
- *JobTitle*: Patternmaker Metal and Plastic, Process Planner (6 years)
- *Industry*: Engineering Services (14 years), Clothing and Textile Manufacturing (11 years)
- *Knowledge*: VBScript, Process Performance Monitoring
- *ComplementarySkill*: Systems Skills, Complex problem solving (10 years), Visual Color Discrimination (14 years)
- *Language*: English (basic writing knowledge), French (excellent reading knowledge)

9 - Elena Pomarico
- *Level*: Computer Science Engineering, Secondary School, Bachelor
- *JobTitle*: Database Administrator, Project Manager
- *Industry*: Banking, IT and Telematics Applications
- *Knowledge*: CplusPlus, Java, Visual Basic
- *ComplementarySkill*: Cooperation, Leadership
- *Language*: English (excellent writing, reading and verbal knowledge), French (good writing knowledge)

10 - Domenico De Palo
- *Level*: Computer Science Engineering (mark 110), Doctoral Degree
- *JobTitle*: Project Manager (4 years), Teachers (4 years), Database Administrator (4 years)
- *Knowledge*: OOprogramming (6 years), Artificial intelligence (4 years), Internet technologies (4 years)
- *ComplementarySkill*: Cooperation (6 years), Complex problem solving (5 years)
- *Language*: English (excellent verbal knowledge)