Competence Discovery and Composition

Badrina Gasmi Boumezoued, Hassina Nacer, Nacer Boudjlida

Competence Discovery and Composition

(In Proceedings of of the 3rd International Symposium ISKO-Maghreb'2013 (http://www.isko-maghreb.org/): Concepts and Tools for Knowledge Management. Marrakech, Morocco, 8-9 November 2013 (Among the awarded papers).

Badrina Gasmi-Boumezoued
A. Mira University, Béjaia, Ageria
Email: badrinagasmi@gmail.com

Hassina Nacer
A. Mira University, Béjaia, Algeria
Email: sino_nacer@yahoo.fr

Nacer Boudjlida
Lorraine University, LORIA, France
Email: Nacer.Boudjlida@loria.fr

***Abstract*—The capture, the structuring and the exploitation of competences of an "object" (like a business partner, an employee, a software component, a Web service, etc.) are crucial problems in various applications, like cooperative and distributed applications or e_business applications. The work we describe here concerns competence advertising, organization, discovery and composition. Indeed, one of the originality of the proposal is in the nature of the answers the intended system can return when seeking for individuals fitted with given competences: answers may be composite ones in that sense that when no single object meets the search criteria, we attempt to find out what a set of objects, when pooled together, do satisfy the whole search criteria. Conceptual Graphs (CGs) are used as a knowledge representation formalism and operations on graphs are used as a search mechanism. A client/server prototype, viewed as a federation of mediators, has been developed as a proof of concept.**

***Keywords*—Knowledge management applications, Composite answers, Conceptual graphs, Mediators federation.**

I. Introduction

A competence management process [1] can be achieved following three steps: (1) *Competence identification*: it consists in describing competences under a formal representation. (2) *Competence organization*: once represented, competences are organized, classified and structured in order to be efficiently exploited and (3) *Competence use*: it consists in exploiting the organized competences. In this work, we aim at exploiting the competences for their discovery, i.e. when searching for entities that meet given needs.

Competence management and discovery find their application in different domains, like component-based programming, semantic-based Web services discovery [2], e-business, human resources management and even enterprise knowledge management [3]. For example, in the e-business domain, we see the application of our work when seeking for possible partners or subcontractors. In human resource management, considering employees enrollment as an example, the application of our wok can be useful when looking for employees satisfying a given work position profile.

In this paper, we aim at proposing a generic approach which can be instantiated in different domains. The ultimate goal is to define a method for competence management and apply the method for competence discovery and composition in distributed knowledge bases. A significant originality of the proposed approach resides in the type of answers we aim at providing. Indeed, when no unique entity satisfies the search criteria, the system attempts to determine a composite answer, i.e. a set of entities that satisfy the whole search criteria, every entity in the resulting set satisfying part of the criteria.

For competence representation and management, we rely on a knowledge representation using Conceptual Graphs (CGs) [4]: we not only represent knowledge as graphs but the reasoning is made thanks to graph-based operations. From a system architecture point of view, we use a mediator-based architecture [5], i.e. a set of distributed and cooperative mediators.

The presentation of this work is structured as follows. Section II presents related work and the work background. Section III presents the proposed approach for competence management and discovery. Section IV provides an overview of the implementation of the approach whereas concluding remarks are in section V.

II. Related Work and Background

The current work is related to three main bodies of research: (i) Knowledge Representation (section II-A), (ii) competence representation and discovery (sections II-B and II-C) and (iii) heterogeneous and distributed architectures (section II-D). We briefly discuss important studies in these research areas.

A. Knowledge Representation

During the past 40 years, a wide variety of Knowledge Representation (KR) formalisms has been developed. In general, these formalisms fall into two categories: (1) those that follow a "logical approach" (like Description Logic [6]) and provide a general reasoning machinery and a representation language which is usually a variant of the first-order predicate calculus and (2) those that follow a "non-logical approach" (like Semantic Networks [7] and CGs [4], [8]) that use graphical interfaces that enable representing knowledge manipulation according to *ad-hoc* data structures. CGs are briefly introduced hereafter.
CGs are presented as a general model for knowledge representation. They were conceived to represent the semantics of natural languages; they evolved to become complete systems in the sense of logic. A CG description represents **ontological knowledge** in a structure called *support* which introduces the vocabulary of the studied domain. The *support* is implicitly used in the representation of **factual knowledge** as labeled graphs called conceptual graphs.

The support consists of (an example is in figure 1) *(i) a hierarchy of concept types* organized around the relation of specialization/generalization, *(ii) a set of relation types* organized into several hierarchies, each of them organizes relation types having the same arity, *(iii) a set of markers or referents* (denoted by I in figure 1) that refers to specific concepts (an unspecified concept can be referenced using a generic marker denoted as *), *(iv) a conformity relation* (τ in figure 1) which relates markers to concept types and *(v) signatures of relations* which represent all the graphs which express constraints associated with every relation. A signature defines the number of the relation's arguments and their types. A graph signature is constituted by elementary graphs from which we can construct more complex graphs.

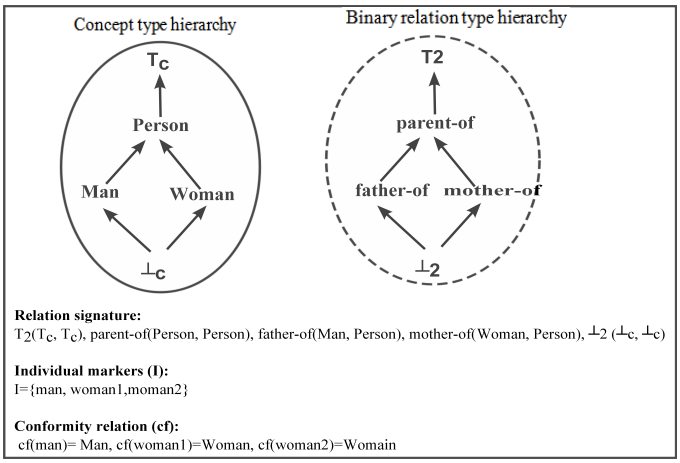

Fig. 1. Conceptual Graph Support

Furthermore, a CG is composed of: (1) A set of concept-nodes labeled from a support. A concept is composed of a *referent* that identifies the represented object, a type which classifies the represented object and (2) a set of relation-nodes labeled from a support. A relation is composed of a label which identifies the type of the relation and a set of edges linking the relation to its related concepts.

CGs can have different concrete notations such as graphical representation, textual notation and Conceptual Graph Interchange Format (CGIF) [9].

In a graphical notation, called display form (DF) (see figure 2), concepts are represented by rectangles and relations are represented by circles or ovals. The arcs that link the relations to the concepts are represented by arrows.

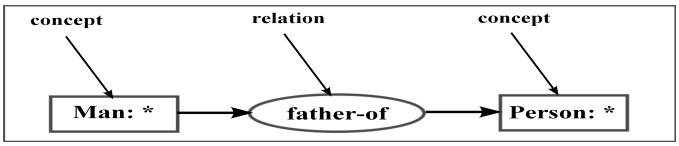

Fig. 2. Conceptual Graph Example

In a textual notation, called linear form (LF), concepts are represented by square brackets and relations are represented by parenthesis. Under a LF notation, the CG of figure 2 is expressed as: [Man: *]→(father-of)→[Person: *].

The CGIF notation has a syntax that uses co-reference labels to represent the arcs. The example in figure 2 is expressed in CGIF as: [Man: *m] [Person: *p] (father-of ?m ?p). *m and *p are variable definitions and ?m and ?p are references to defined variables.

A CGs being a logic system, it can easily be translated under a predicate logic form. As an example, the CG in figure 2 is expressed as: $\exists$ m $\exists$ p: Person(p) $\wedge$ Man(m) $\wedge$ father-of(m,p).

Furthermore, a variety of operations and extensions [8] are defined on CGs. We recall hereafter those that are necessary to the comprehension of the remainder of this paper.

-**Projection:** is defined as an application $\prod$ of the nodes of a graph H towards the nodes of a graph G such as: (1) for each concept c in H, $\prod$(c) is either a specialization or the same as c, (2) for each relation r in H, $\prod$(r) is either a specialization or the same as r, (3) if the i^{th} edge of r is linked to a concept c in H, then the i^{th} edge of $\prod$(r) must be linked to $\prod$(c) in G.

-**The Normalization operation** returns a graph under a normal form which respects a structure where the markers are unique by merging concepts having the same individual marker. The normal form of a graph avoids semantic and logical ambiguity in CGs. Formally, let H be a CG, and C be the set of its concepts. H is under its normal form if for each couple of concepts (c1, c2) c1 and c2 $\in$ C, referent(c1) $\neq$ referent(c2).

-**The Disjoint sum** consists in drawing another CG next to the original CG [10]. Formally, let H1 and H2 be two CGs, and let (C1, R1, E1) and (C2, R2, E2) the concept set, the relation set and the edge set of H1 and H2 respectively. The disjoint sum of H1 and H2 is a CG H(C, R, E) such as (1) C is the union of C1 and C2, (2) R is the union of R1 and R2 and (3) E is the union of E1 and E2.

-**Headed graphs** are graphs that have a certain node chosen as the semantic head.

-**Conceptual graph rules** [11] were proposed as an extension of simple CGs to represent "IF A THEN B" knowledge where A and B are simple CGs. Formally, a graph rule is constituted from an hypothesis graph A, a conclusion graph B and a set of attach points corresponding to connection links between A and B. The rule application mechanism in a CG is based on the projection operation.

B. Competence Representation

Competence representation is a sub-field of KR which extends current KR languages to be more suited for competence description [12]. In [13], competences are methods of object-oriented software. Furthermore, DL is used to describe the intended semantics of these objects and the possible constraints involving their methods. In [14], entities are software objects and competences are the capabilities of a software object. In [5], entities are a set of activities (or functions) describing a given domain, an activity being described by the set of the required competences to carry it out. These competences represent the set of properties (or attributes) of the activities and their intended semantic is expressed using DL.

C. Competence Discovery

Competence discovery consists in searching entities having a set of required competences in order to satisfy a given objective. Answers to a competence discovery request may be of two types: (1) single answers, when single entities satisfy the search criteria, (2) cooperative or composite answers when no single entity, but a set of entities, meets the search criteria. In [12], competence discovery is defined as a query-answer process that attempts to find out which kind of entities owns a competence, and who they are. In [5], a request X is viewed in term of DL language as a concept having the given competences and the request evaluation consists in locating this concept in the concept classification hierarchy. The answers of a request are the individuals or the instances of all the concepts subsuming X. In an extended work [15], the authors present a method to produce composite answers thanks to the notion of "complementary objects" that is founded on the complement concept in DLs [16].

D. Heterogeneous and Distributed Architectures

In order to satisfy a competence search request in an heterogeneous and distributed environment like Internet, we have to cope

with competence descriptions expressed in different formalisms either locally or remotely. This facility requires techniques to transform a competence description from one formalism into another, together with communication between the systems managing the various competence descriptions. Different heterogeneous and distributed architectures are candidate to the implementation of these systems, like Service Oriented Architectures, Peer to Peer (P2P) architectures [17], [18] and Mediator-based architectures, the latter being the one we rely on.

A mediation architecture [19] tries to solve the problem of the access and the integration of information by introducing the notion of a mediator as "a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of application". The mediation can be of two types:

-**Centralized mediation:** where only one mediator is considered. In this case, all the data sources are stored in the same base.

-**Distributed mediation:** (or federation of mediators) in which a set of mediators agree to be considered as a single entity when applications demand for services to the federation. Distributed mediation systems have become a reference architecture to integrate both structured and semi-structured data [19], [20]. In addition, many mediator-based approaches have been proposed in the literature. In [21], a single mediator is designed to offer an adequate level of decision-making integration of heterogeneous computer systems. The Conflict Resolution Environment for Autonomous Mediation (CREAM) system has been implemented and it provides various user groups with an integrated and collaborative facility to achieve semantic interoperability among participating heterogeneous information sources [22]. The KRAFT (Knowledge Reuse And Fusion/Transformation) architecture provides a generic infrastructure for knowledge management applications. It supports virtual organization using mediator agents [23]. In [5], [15], [24], [25], an architecture based on a heterogeneous federation of mediators has been adopted. In this architecture, great emphasis is on cooperation and heterogeneity aspects.

Now, let us turn toward our actual proposal for competence management and discovery.

III. Proposal: Competence Management and Discovery using CGs

In this section, we present the approach we propose for competence management and discovery using conceptual graphs as a competence representation formalism and operations on graphs as a reasoning mechanism. The mediator-based architecture, as well as the system architecture, will be described in section IV.

A. Conceptual Architecture

In the proposed approach, a mediator-based architecture has been adopted as described in [5]. It is very similar to the notion of discovery agency in the Web service architecture [24]. In this architecture, an "entity", called exporter, publishes its competences at one or more mediators (arrow (a) in figure 3). Entities, called importers, send requests to the mediator asking for exporters fitted with a given set of competences (arrow (b) in figure 3). The mediator explores its competence base to try to satisfy the request. The competence search process is founded on the exported competences and on relationships between them, these relationships being transparently established by the mediator. When the request can be satisfied by some exporters, the references of these exporters are sent back to the importer (arrow (c) in figure 3).

In this architecture, some cases may conduct to a failure of the request when only one mediator is involved. But, if we assume a grouping of mediators, these cases are typical cases where cooperation of mediators is required. When a mediator partner fails in the satisfaction of a request, we need to determine what is missing to the entities to satisfy request. That missing part is then transmitted to a mediator in the federation who, in turn, behaves like the preceding mediator. Therefore, satisfying a request may fall

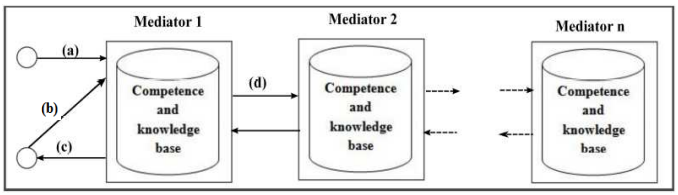

Fig. 3. The Mediator-Based Architecture

under different cases [12]:
1. there exist exporters that fully satisfy the request;
2. there exist exporters that partly satisfy the request but, when "combining" or composing the competences of different exporters one can fully satisfy the request;
3. no single exporter nor multiple exporters satisfy the request. In the latter situation, the mediator may initiates a cooperation process with other mediators to attempt to satisfy the request (arrow (d) in 3).

In addition, in a federated mediator architecture, the competence discovery can fall under the following situations:

1. *Homogeneous local satisfaction* where the request and the knowledge base are in the same KR language, and the knowledge base is located in one server.

2. *Homogeneous distributed satisfaction:* where the request and the knowledge base are in the same KR language, and the knowledge base is distributed in several servers.

3. *Heterogeneous satisfaction:* where the request and the knowledge base are in different KR languages and the knowledge base may be distributed.

In this work, we only deal with the homogeneous distributed satisfaction.

B. Competence Representation

Using CGs, competences are represented by relations and entities are represented by concepts. For example, saying that a programmer p has competences in Java programing is represented as shown in figure 4.

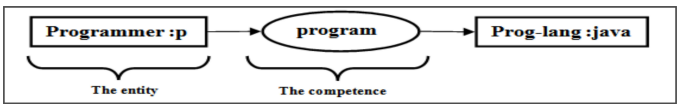

Fig. 4. Competence Representation Example

However, the simple CG model does not allow to adequately represent entities and their competences. Indeed, in a simple CG

model, the semantic of a concept type or a relation type is only given by its position in the type hierarchies; the only mechanism that enables defining a type is the specialization/generalization relation. This representation of types is poor and misses a lot of expressivity to represent generic information about types and also some relation properties such as transitivity, symmetry and reflexivity. To deal with these problems, we propose to use CG rules as described hereafter.

1. *Concept type definition:* To represent generic information about concept types, these types must be defined. "Concept type definition" is defined here as "an either necessary or necessary and sufficient conditions that entities must verify in order to belong to a concept type". These conditions are formalized using conceptual graph rules. For example, the concept type *Mother* defined as a "woman that is *mother_of* a person" is defined as follows:

$$[Mother : *x] \Rightarrow [Woman :?x] \rightarrow (mother\_of) \rightarrow [Person : *]$$
$$Woman : *x] \rightarrow (mother\_of) \rightarrow [Person : *] \Rightarrow [Mother :?x].$$

2. *Relation type definition:* In the same way, a "Relation type definition" is "an either necessary or necessary and sufficient conditions which must be verified in order to belong to a relation type". For example, the relation type *grandmother_of* can be defined as follows:

$$[Woman : *x] \rightarrow (grandmother\_of]) \rightarrow [Person : \ *y] \Rightarrow [Woman : *x] \rightarrow (mother\_of) \rightarrow [Person : *] \rightarrow (parent\_of) \rightarrow [Person :?y].$$
$$[Woman : *x] \rightarrow (mother\_of]) \rightarrow [Person : *y] \rightarrow (parent\_of) \rightarrow [Person : *y] \Rightarrow [Woman :?x] \rightarrow (grandmother\_of) \rightarrow [Person :?y].$$

3. *Meta-knowledge on relations:* Relation properties are also formalized using CG rules. For example, the following rules enables expressing the fact that the relations *parent_of* and *child_of* are symmetric ones:

(1) $[Person : *x] \rightarrow (child\_of) \rightarrow [Person : *y] \Rightarrow [Person : *y] \rightarrow (child\_of) \rightarrow [Person :?x].$
(2) $[Person : *x] \rightarrow (parent\_of) \rightarrow [Person : *y] \Rightarrow [Person :?y] \rightarrow (parent\_of) \rightarrow [Person :?x].$

As a result of the rule-based representation we propose, the domain representation is composed of (1) *Ontological knowledge*, represented by the support, to which we add a component named "Rule base" (RB) containing the set of rules used to define the types and the relation properties and, (2) *Factual knowledge*, represented by CGs labeled from the support. In this work, CGs serve for representing entities together with their acquired competences. Each graph is then published in one of the mediators of the federation. The set of the competences that are published in a given mediator are collected into a single CG named "Competence Base" and denoted as CB.

A CB is built and updated every time where a new competence (represented by a CG noted P) is published. For each published graph P, we follow the three following steps:

(1) Disjoint sum of the graphs P and CB in order to add published competences to the CB.
(2) Normalize the graph CB: this normalization avoids graph redundancy and then minimizes the search space.
(3) Apply the rules that are present in RB on the graph CB. This is a very important step : it allows reasoning over the CB in order to add all implicit knowledge that is not directly published into the CB.

C. Competence Discovery

Section III-C1 presents the representation of a request while sections III-C2 to III-C3 present its satisfaction process.

1) Request Representation: The request is represented as a headed CG form noted RG in which:
(i) The searched entities are represented by the head t of RG. We introduce a special marker ? logically equivalent to the * marker in order to indicate such a node.
(ii) The requested competences are represented by relations which are directly attached to the node t.
(iii) The rest of RG represents conditions on the requested competences.

For example, seeking for men having some competences in UML (Unified Modeling Language) and some competences in programing using languages that support classes is represented as shown in figure 5.

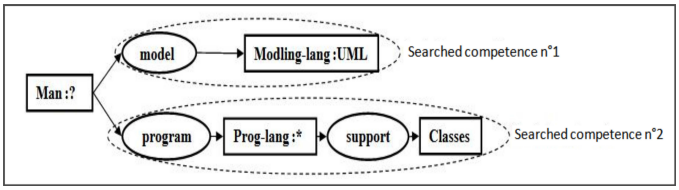

Fig. 5. A Request Example

2) Local Request Satisfaction: The local satisfaction of a request R runs as follows: *(i)* Normalize R in order to minimize its size and as a consequence to minimize the search and to avoid logical and semantic ambiguities, *(ii)* delete from R all the connected components that do not contain the head t, because these components are independent from the searched entities, *(iii)*Project R on BC and *(iv)* if at least one projection is found, then there is at least one single answer to the request. Answers are then all the projections (images) of the head node t. Otherwise, search for possible composite answers to the request.

As an example, the satisfaction of the request R in the left part in figure 6 is the circled concept, the right part of the figure being the concept base.

In order to find possible composite answers to a request R, we decompose R into sub-requests where every sub-request consists in searching entities having one of the required competences and we proceed as follows:
(i) Decompose R into n sub-requests $R_i (i \in [1, n])$, each R_i containing the head of R connected to one of the sub-graphs representing a discovery request for one competence, together with conditions on it (see section III-C1).
(ii) Satisfy all the sub-requests, one independently from the others.
(ii) If all the sub-requests are satisfied then composite answers are the compositions of the answers of the sub-requests.

As an example, to find composite answers to the request in figure 5, R is decomposed into two sub-requests (figure 7).

In addition, the satisfaction of a sub-request R_i proceeds as follows:

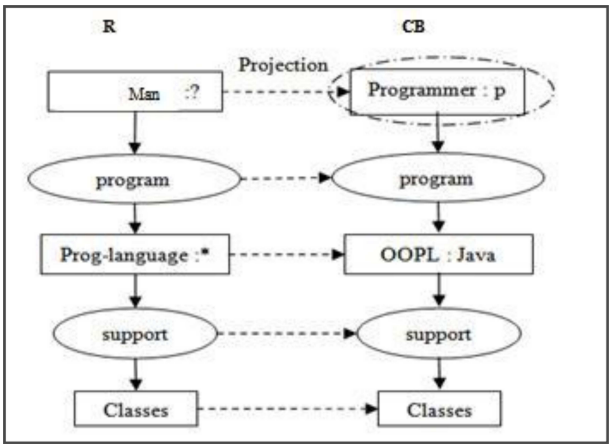

Fig. 6. A Local Request Satisfaction Example

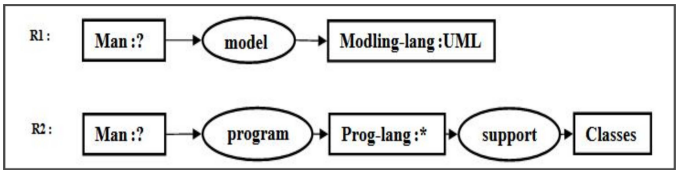

Fig. 7. A Request Decomposition Example

(i) Project R_i on BC.
(ii) If at least one projection is found then R_i is locally satisfied and the replies to R_i are images of the head node.
(iii) Otherwise, try the distributed satisfaction of R_i thanks to the cooperation with other mediators and this is explained in the coming section.

3) Cooperative Request Satisfaction: In a federation of mediators, part of a sub-request R_i may be satisfied in one of the mediators of the federation whereas another part may be satisfied in another one. In term of conceptual graphs, this means that a part of the graph that represents R_i may be projected on the CB in one mediator whereas another part may be projected on the CB in another one, as illustrated in the following example, considering the sub-request R2 in figure 7.

Assume that two mediators M1 and M2 are available (figure 8 shows parts of their competence bases denoted CB1 and CB2 respectively).

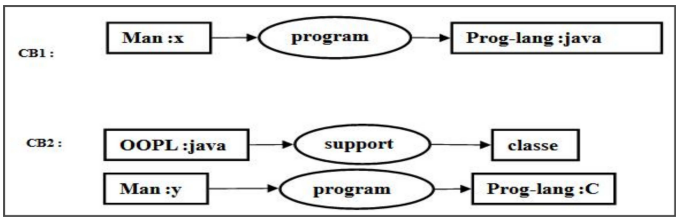

Fig. 8. Competence Base Examples

In both the mediators, only a part of R_2 is satisfied: in CB1, there is a person having java-programming competences and in CB2, we know that java supports classes. So, in order to satisfy a sub-request in a federation, it is sufficient to find which parts of R_i can be projected on the CB of a mediator and which parts cannot. However, the projection operation such as defined in the CG formalism does not allow to find this type of information. For that reason we propose to proceed according to the following steps:
Step1: Decompose R_i into elementary parts containing only one relation.
For example, the sub-request R_2 in figure 7 is decomposed into two parts (figure 9).

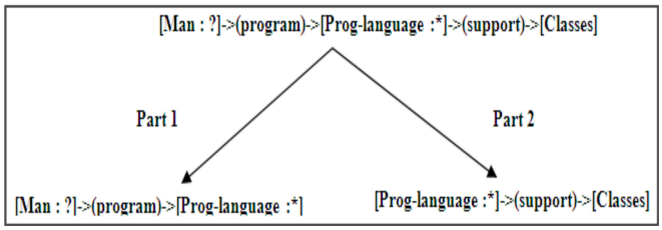

Fig. 9. Sub-Request Decomposition Example.

Step2: Project these parts on each CB in the federated mediators:

(1) The projection of the two parts on CB1 is shown in figure 10.
(2) Add the projection of the two parts on CB2 (figure 11).

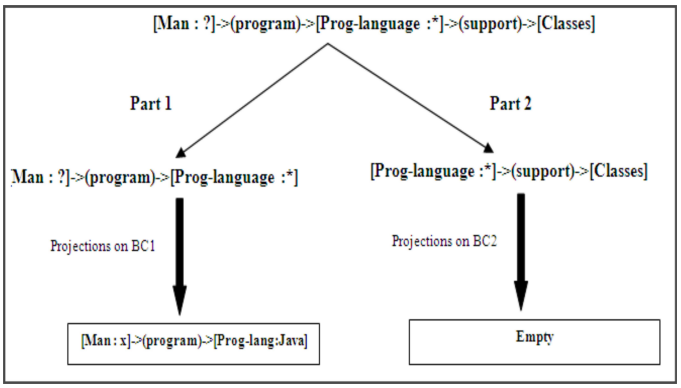

Fig. 10. Sub-Request Parts Projection on CB1

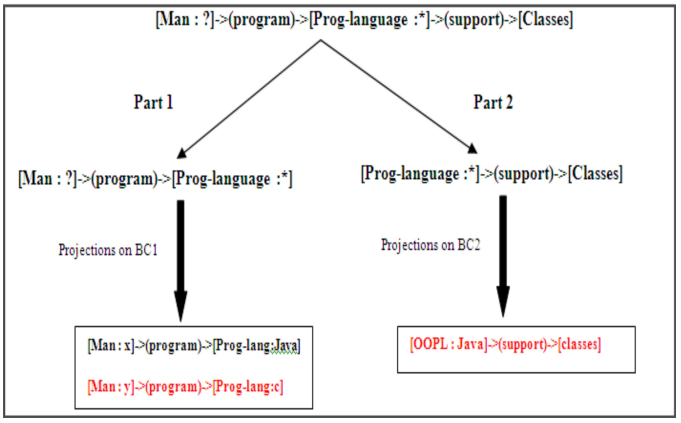

Fig. 11. Sub-Request Parts Projection on CB2

Step3: Check whether the projections can be joined and if they do, then the sub-request is satisfied and the satisfactions are the projections of the sub-requests' heads (see the dotted parts in figure

12).

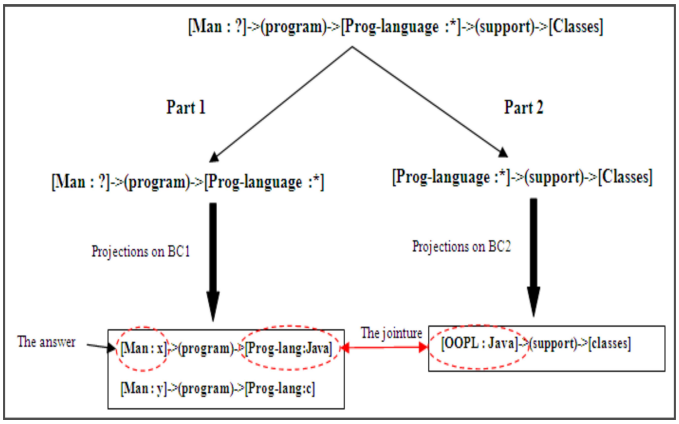

Fig. 12. Sub-Request Satisfaction Verification

Let us now describe the prototype we developed as a support of our proposals.

IV. Implementation

For an experimental validation of the proposed approach, we implemented a prototype using many software components. There exist several tools which implement CGs in particular for research purposes and for information extraction [26]–[29]. However, few of these tools offer a complete software environment for the widest possible use of the model: the storage and the manipulation of a large number of graphs. For that reason, we choose to use the CoGITaNT library (Conceptual Graphs Integrated Tools allowing Nested Typed graphs), a library of C++ classes (open source, developed at LIRM Montpellier, CNRS, France) which allows developing applications based on the CG knowledge representation scheme.

We illustrate hereafter the prototype functioning thanks to examples. We present first, the domain population and then examples of competence discovery.

1) Domain population: as an example, we consider the computer science competence management domain represented in terms of a concept type hierarchy (figure 13), a relation type hierarchy (figure 14), rules used to define concept types, relation types and relations properties.

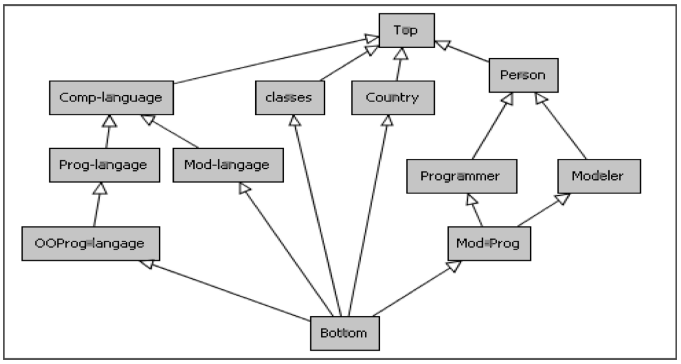

Fig. 13. The Concept Hierarchy

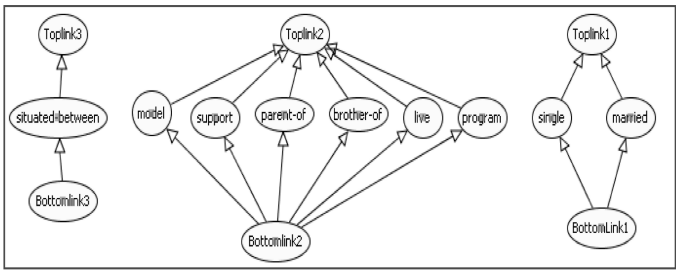

Fig. 14. The Relation Hierarchy

2) Competence bases are in the figures 15 and 16.

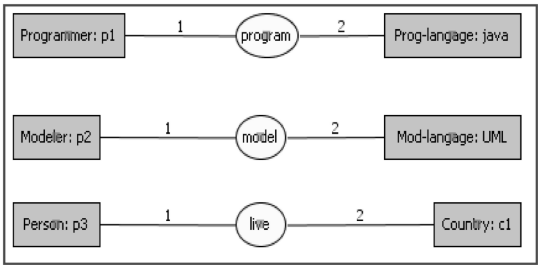

Fig. 15. Competence Base of the Mediator M1.

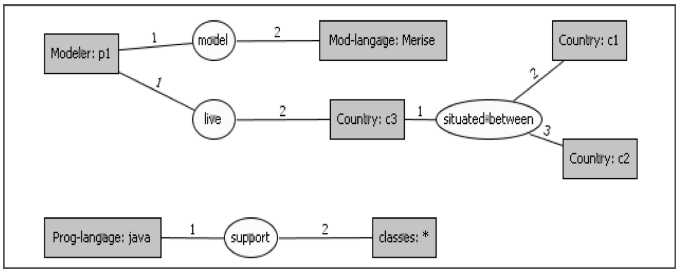

Fig. 16. Competence base of the mediator M2.

3) A local query-satisfaction example is shown in the figure 17, in which *graph2* denotes the query.

4) A distributed query-satisfaction example: the figures 18 and 19 illustrate the result of the query denoted as *graph1* in the figure 18.

V. Discussion and Concluding Remarks

In this paper, we presented an approach for competence management and discovery using conceptual graphs (CG) to provide a semantic description of an application domain. Acquired competences are organized under a CG form that is built and updated every time a new competence is published. The advantage of this organization form is that the application of graph rules at publication time facilitates the search and may reduce the response time, since all implicit information are available thanks to the application of these rules at publication time. For competence discovery, we use operations on graphs and the projection is used as a basic operation in the discovery process. For distributed satisfaction of a request, we use another form of graph decomposition where a sub-request is decomposed into elementary parts containing only one relation. In addition, for experimentation purposes, we implemented a federated mediation prototype based on the client/server architecture of COGITANT [26].

Fig. 17. Local satisfaction of a request

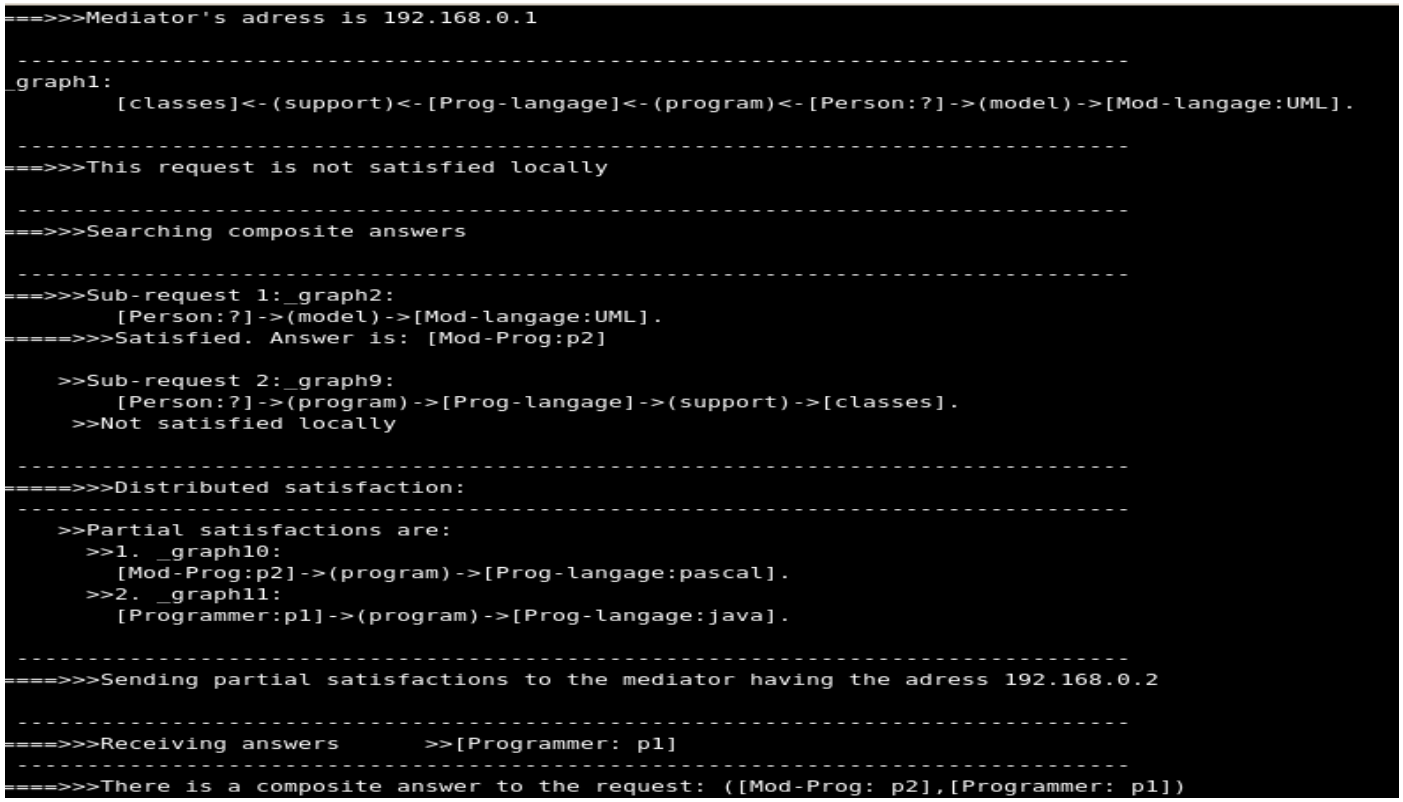

Fig. 18. Distributed Request Satisfaction in M1.

The prototype is fully written in C++ programming language and it has been successfully verified under Linux and MICROSOFT Windows XP operating systems.

Further work is to consider the complexity of the search algorithm and to cope with heterogeneous mediators cooperation, i.e. mediators where knowledge bases are described in different languages. An additional on-going research topic concerns the dynamic and semantic-based identification of possible cooperating mediators for unsatisfied parts of a competence request together with a performance comparative analysis of a P2P implementation against an implementation using cloud computing technology.

REFERENCES

[1] Giuseppe Berio and Mounira Harzallah. Knowledge management for competence management. *Journal of Universal Knowledge Management*, 0:21–28, 2005.

[2] Hassina Talantikite, Djamil Aissani, and Nacer Boudjlida. Semantic annotations for web services discovery and composition. *Computer Standards and Interfaces*, 31:1108–1117, 2009.

[3] Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge-creating company: how Japanese companies create the dynamics of innovation*. Oxford University Press, New York, 1995.

[4] John F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, USA, 1984.

[5] Nacer Boudjlida. A Mediator-based architecture for capability management. In *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA)*, pages 45–50, Cambridge, USA, November 2002.

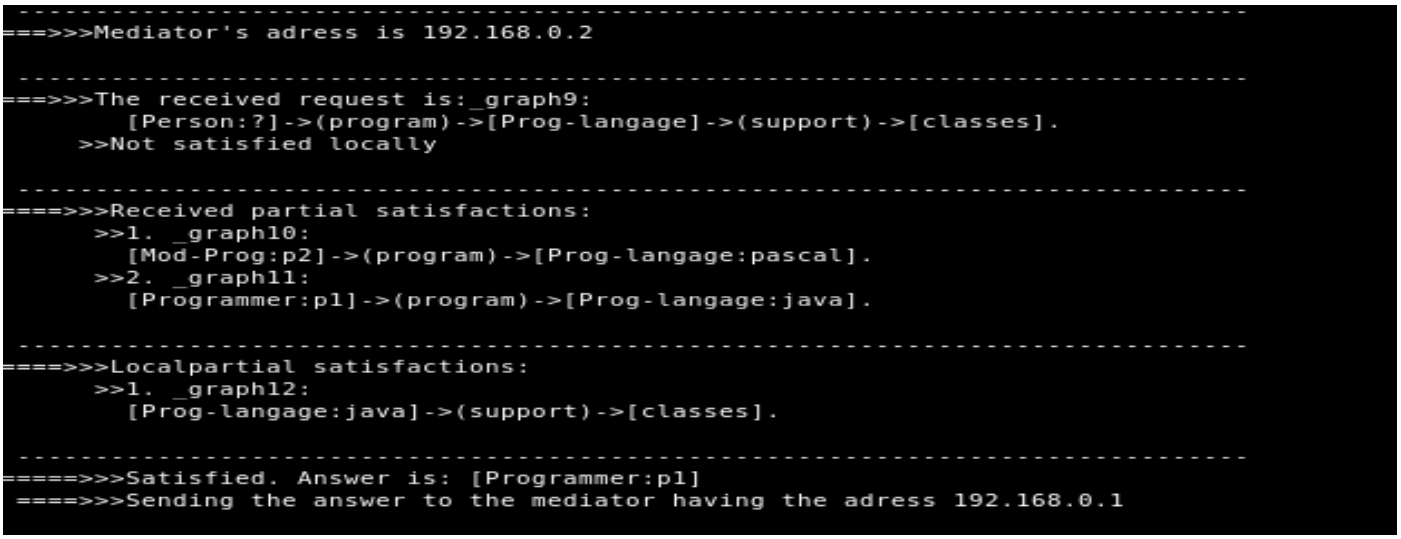

Fig. 19. Distributed Request Satisfaction in M2.

[6] Description logics. Available from: http://www.dl.kr.org/, [accessed 5 june 2012].

[7] M. R. Quillian. Semantic memory. In *Semantic Information Processing*, pages 227–270. MIT Press, 1968.

[8] Michel Chein and Marie-Laure Mugnier. *Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer, London, 2008.

[9] CGIF. "Conceptual Graphs Interchange Format". Available from: http://www.webkb.org/doc/CGs.html, [accessed 5 june 2012].

[10] Nicolas Nicolov, Chris Mellish, and Graeme Ritchie. Sentence generation from conceptual graphs. In *Proceedings of the 3rd International Conferences on Conceptual Structures (ICCS)*, pages 74–88, CA, USA, August 1995.

[11] Eric Salvat. *Raisonner avec des opérations de graphes: graphes conceptuels et règles d'inférence*. Phd thesis, Montpellier II University, 1997.

[12] Dong Cheng. *Competence Management and Discovery in Heterogeneous Environments (Gestion et découverte de compétences dans des environnements hétérogènes)*. Phd thesis, Henri Poincaré-Nancy1 University, France, 2008.

[13] Alex Borgida and Prem Devanbu. Adding more dl to idl: Towards more knowledgeable component interoperability. In *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, pages 378–387, CA, USA, May 1999.

[14] Mohammed Bouchikhi and Nacer Boudjlida. Using larch to specify the behavior of objects in open distributed environments. In *Proceedings of the 5th Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI)*, pages 275–287, Tunis, TUNISIA, December 1998.

[15] Nacer Boudjlida and Dong Cheng. Complement concept and capability discovery. In *Proceedings of the 1rst International Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI), in connection with the 16th Conference on Advanced Information Systems Engineering (CAiSE)*, pages 337–342, Riga, Latvia, June 2004.

[16] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48:1–26, 1991.

[17] Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, and all. Data management for peer-to-peer computing: A vision. In *Proceedings of the 5th International Workshop on the Web and Databases (WebDB)*, pages 89–94, June 2002.

[18] Michael Moore and Tatsuya Suda. Adaptable peer-to-peer discovery of objects that match multiple keywords. In *International Symposium on Applications and the Internet : Workshops (SAINT Workshops'04)*, pages 402–407, Tokyo, Japan, 2004.

[19] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.

[20] Yannis Papakonstantinou, Hector Garcia-molina, and Jeffrey Ullman. Medmaker: A mediation system based on declarative specifications. In *Proceedings of the 12th International Conference on Data Engineering (ICDE)*, pages 132–141, February 1996.

[21] H.-K. Tönshoff, I. Seilonen, G. Teunis, and P. Leito. A mediator-based approach for decentralised production planning, scheduling and monitoring. In *Proceedings of the CIRP International Seminar on Intelligent Computation in Manufacturing Engi-neering(ICME)*, June 2000.

[22] Sudha Ram, Jinsoo Park, and Yousub Hwang. Cream: A mediator based environment for modeling and accessing distributed information on the web. In *Proceedings of the 19th British National Conference on Databases: Advances in Databases (BNCOD)*, pages 58–61, London, UK, July 2002.

[23] Alun Preece. A mediator-based infrastructure for virtual organisations. In *Agents-2001 Workshop on Intelligent Agents in B2B E-Commerce*, Montreal, Canada, 2001.

[24] Dong Cheng and Nacer Boudjlida. Capability management and discovery in description logic. In *Proceedings of the 3^{rd} International Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI), in connection with the 18th Conference on Advanced Information Systems Engineering (CAiSE)*, pages 678–681, June 2006.

[25] Dong Cheng and Nacer Boudjlida. An architecture for heterogenous federated mediators. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE)*, June 2005.

[26] Cogitant : Conceptual graphs integrated tools allowing nested typed graphs. Available from:http://cogitant.sourceforge.net, [accessed 5 june 2012].

[27] Cogui: Outil de création graphique de graphes conceptuels. Available from :http://www.lirmm.fr/cogui, [accessed 5 june 2012].

[28] Charger: Editeur de graphes conceptuels. Available from: http://sourceforge.net/projects/charger/", [accessed 5 june 2012].

[29] Finnegan Southey and James G. Linders. Notio - a java api for developing cg tools. In *Proceedings of the 7th International Conference on Conceptual Structures (ICCS)*, pages 262–271, Blacksburg, Virginia, USA, July 1999.