Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

"Ontology-based Clustering in a Peer Data Management System"

Por

Carlos Eduardo Santos Pires

TESE DE DOUTORADO

Recife, Pernambuco, Brasil
Abril 2009

Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

Carlos Eduardo Santos Pires

"Ontology-based Clustering in a Peer Data Management System"

Submitted in partial fulfillment of the requirements for the degree of doctor of Philosophy.

Supervisor: Ana Carolina Salgado (Docteur)

Recife, Pernambuco, Brasil
Abril 2009

Pires, Carlos Eduardo Santos
Ontology-based clustering in a Peer Data Management System / Carlos Eduardo Santos Pires - Recife : O Autor, 2009.
xix, 155 p. : il., fig., tab.

Tese (doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2009.

Inclui bibliografia, glossário e apêndice.

1. Sistemas de gerenciamento de banco de dados. 2. Ontologia. 3. Cluster analysis. I. Título.

005.74 CDD (22. ed.) MEI2009- 078

To my father Antonio Carlos and my mother Sonia Maria.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my scientific advisor, Prof. *Ana Carolina Salgado*, for her continuous and caring help, advice, and encouragement during the four years of my Ph.D. studies at UFPE. Particularly, I am thankful for her teaching me how to be deep and precise, and how to learn to think clearly. She has always been supportive and encouraging when I have needed more inspiration. Without her help, I definitely would not have come this far.

I am also very grateful to Prof. *Mokrane Bouzeghoub* and Prof. *Zoubida Kedad* for their support and advice during my doctoral internship at the Université de Versailles Saint-Quentin-en-Yvelines (UVSQ). Their impressive experience in computer science has made me see research from a much wider perspective. I could not forget my friends *Sofiane Abbar* and *Juan Carlos* for the productive discussions and suggestions to my, at that time, under-development work. In short, thanks a lot to all the PRiSM team. I hope I'll be back soon!

While in Paris, I was very lucky to meet *Franceline Reynaud*, *Mariane Lira*, *Ricardo Pimenta*, and *Carol Vimieiro* who became my friends for life. *C'est sûr que la Maison du Brésil ne va pas être la même sans vous. Coucou! Reveillez-vous!* Good luck with your theses and let's have our first wine meeting after France.

I would also like to thank CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) and FACEPE (*Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco*) for the financial support during the development of this work.

Special thanks to my Brazilian colleagues *Damires Yluska*, *Bernadette Lóscio*, and *Rosalie Belian* for many helpful discussions which have contributed substantially to this thesis. They have created a friendly and joyful atmosphere, which has been a great pleasure for me to work. I also would like to thank all my friends from the Center for Informatics of UFPE *Vaninha Vieira*, *Fábio Avila*, *Guilherme Amorim*, *Rilson Nascimento*, *Paulo Roberto*, and *Adeline*

Sousa. Many thanks to *Rocir Leite*, *Thiago Pacheco*, *Victor Alencar*, and *Guilherme Souza*. You helped me so much with the implementation issues.

I eternally thank all my family for their interminable support and understanding: my parents *Antonio Carlos* and *Sonia Maria*, my brother *Marco Antonio*, my sister in law *Andreia Rosa* as well as my little nephews *Caio César* and *Felipe Rosa*. I am particularly thankful to my beloved fiancée, *Vládia Freire*, without whom it would have been impossible to complete this thesis.

Thanks God!

ABSTRACT

Peer Data Management Systems (PDMS) are advanced P2P applications which enable users to transparently query several distributed, heterogeneous, and autonomous data sources. Each peer represents a data source and exports its entire data schema or only a portion of it. Such schema, named exported schema, represents the data to be shared with the other peers of the system and is commonly described by an ontology.

The most studied data management issues in PDMS are related to schema mappings and query processing. These issues can be improved if peers are efficiently disposed in the overlay network according to a semantic-based approach. In this context, the notion of semantic community of peers is of great importance since it aims at logically approximating peers with common interests about a specific topic. However, due to the dynamic behavior of peers, the creation and maintenance of semantic communities is a challenging issue in the current stage of development of PDMS.

The main goal of this thesis is to propose an ontology-based process to incrementally cluster semantically similar peers that compose communities of a PDMS. In this process, peers are grouped according to their corresponding exported schema (an ontology) and ontology management processes (e.g. matching and summarization) are used to assist peer connection. A PDMS architecture is proposed to facilitate the semantic organization of peers in the overlay network. In order to obtain the semantic similarity between two peer ontologies we propose a global similarity measure as output of an ontology matching process. To optimize ontology matching an automatic process for summarizing ontologies is also proposed. A simulator has been developed resembling the architecture of the PDMS. The proposed ontology management processes have also been developed and included in the simulator. Experimentations of each application in the context of the PDMS as well as the results obtained from these experiments are presented.

Keywords

Peer-to-Peer, Peer Data Management Systems, Semantic Community, Ontology Matching, Ontology Summarization, Similarity Measure

RESUMO

Os Sistemas P2P de Gerenciamento de Dados (PDMS) são aplicações P2P avançadas que permitem aos usuários consultar, de forma transparente, várias fontes de dados distribuídas, heterogêneas e autônomas. Cada *peer* representa uma fonte de dados e exporta seu esquema de dados completo ou apenas uma parte dele. Tal esquema, denominado esquema exportado, representa os dados a serem compartilhados com outros *peers* no sistema e é comumente descrito por uma ontologia.

Os dois aspectos mais estudados sobre gerenciamento de dados em PDMS estão relacionados com mapeamentos entre esquemas e processamento de consultas. Estes aspectos podem ser melhorados se os *peers* estiverem eficientemente dispostos na rede *overlay* de acordo com uma abordagem baseada em semântica. Nesse contexto, a noção de comunidade semântica de *peers* é bastante importante visto que permite aproximar logicamente *peers* com interesses comuns sobre um tópico específico. Entretanto, devido ao comportamento dinâmico dos *peers*, a criação e manutenção de comunidades semânticas é um aspecto desafiador no estágio atual de desenvolvimento dos PDMS.

O objetivo principal desta tese é propor um processo baseado em semântica para agrupar, de modo incremental, *peers* semanticamente similares que compõem comunidades em um PDMS. Nesse processo, os *peers* são agrupados de acordo com o respectivo esquema exportado (uma ontologia) e processos de gerenciamento de ontologias (por exemplo, *matching* e sumarização) são utilizados para auxiliar a conexão dos *peers*. Uma arquitetura de PDMS é proposta para facilitar a organização semântica dos *peers* na rede *overlay*. Para obter a similaridade semântica entre duas ontologias de *peers*, propomos uma medida de similaridade global como saída de um processo de *ontology matching*. Para otimizar o *matching* entre ontologias, um processo automático para sumarização de ontologias também é proposto. Um simulador foi desenvolvido de acordo com a arquitetura do PDMS. Os processos de gerenciamento de ontologias propostos também foram desenvolvidos e incluídos no simulador. Experimentações de cada processo no contexto do PDMS assim como os resultados obtidos a partir dos experimentos são apresentadas.

Palavras-chave

Peer-to-Peer, Sistemas P2P de Gerenciamento de Dados, Comunidade Semântica, *Ontology Matching*, Sumarização de Ontologias, Medida de Similaridade

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

API	*Application Programming Interface*
DAG	*Directed Acyclic Graph*
DHT	*Distributed Hash Table*
PDMS	*Peer Data Management System*
P2P	*Peer-to-Peer*
OAEI	*Ontology Alignment Evaluation Initiative*
OPDMS	*Ontology-based Peer Data Management System*
OWL	*Ontology Web Language*
RDF	*Resource Description Framework*
SAT	*Propositional SATisfiability*
SON	*Semantic Overlay Network*
SWRL	*Semantic Web Rule Language*
TTL	*Time-to-live*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*

CHAPTER 1

INTRODUCTION

"Never in the field of human conflict was so much owed by so many to so few"
Winston Churchill

In the last few years, there has been a growing interest in the *Peer-to-Peer* (P2P) computing paradigm, primarily boosted by the popular file-sharing applications that enable massive data sharing among millions of users [Kantere *et al.*, 2008]. The P2P paradigm is characterized by a fully distributed and cooperative network design, where peers collectively form a system without any supervision [Rocha *et al.*, 2004].

In a P2P system, peers communicate through an *overlay network*, i.e. a virtual (logical) network which runs as an overlay on top of a physical network [Doval and O'Mahony, 2003]. According to the overlay topology employed, P2P systems are categorized into three kinds [Sung *et al.*, 2005]: (i) *unstructured*, where a peer may join and leave the network without any notification and may connect to any other peer it wishes [Freenet, 2009]; (ii) *structured*, where peers are organized into a rigid structure and connections between peers are fixed according to a certain protocol, e.g. Chord [Stoica *et al.*, 2001]; and (iii) *hybrid*, where data sharing is decentralized but a centralized directory is available [Milojicic *et al.*, 2002]. Particularly, some works [Fiorano, 2003; Sung *et al.*, 2005] also include the *super-peer* category, where a centralized topology is embedded in a decentralized one [Yang and Garcia-Molina, 2003].

Among several P2P applications that have been proposed, *Peer Data Management Systems* (PDMS) [Halevy *et al*., 2003b; Lenzerini, 2004; Tatarinov and Halevy, 2004; Valduriez and Pacitti, 2004; Halevy *et al*., 2006; Mandreoli *et al*., 2007; Kantere *et al*., 2008; Lodi *et al*., 2008] play a leading role in sharing semantically rich information. In a PDMS, each peer is an autonomous source that makes available an *exported schema* [Sung *et al*., 2005]. Sources store and manage their data locally, revealing part of their schemas to the other peers. Due to the lack of a single global schema [Aberer *et al*., 2002], each peer expresses and answers queries based on its exported schema.

Peers also perform local coordination with their *acquaintees*, i.e. their one-distance neighbors in the overlay network [Bernstein *et al*., 2002]. During an acquaintance, two peers exchange information about their exported schemas and create *schema mappings*. Query processing in a PDMS consists in propagating the query, submitted in any of the peers, on paths of limited depth in the corresponding overlay network [Ng *et al*., 2003]. At each routing step, the query is reformulated to the exported schema of its new host based on the respective schema mappings [Tatarinov and Halevy, 2004].

1.1 Problem Definition

In a PDMS, the connection of a new peer requires the definition of the peer's neighbors in the overlay network. Although less dynamic than in traditional P2P file sharing systems, peer connection is of great importance, especially because peers can share content belonging to distinct knowledge domains. In this sense, an arbitrary approach to connect peers is considered inefficient regarding query processing, since peers sharing different content can become neighbors [Löser *et al*., 2003]. Consequently, "poor" quality schema mappings are established between them [Heese *et al*., 2005] and incorrect and/or inconsistent results can be obtained [Aberer *et al*., 2002].

Another problem caused by the arbitrary connection of peers in a PDMS is concerned with query processing. Such problem occurs if semantically similar peers are logically positioned far from each other in the overlay network [Castano *et al*., 2003]. As a consequence, the overall query processing task is affected. For instance, a query may have to be reformulated several times from peer to peer until it reaches *relevant peers*, i.e. peers that are able to answer the query [Kantere *et al.*, 2008]. During query routing, many irrelevant peers can be contacted. As a result, query processing time is increased and an excessive number of inconsistent query results may be returned by the involved peers. In some cases, an arbitrary approach to connect peers in the overlay network can cause a complete isolation of peers sharing semantically similar content [Castano *et al.*, 2003]. Relevant peers cannot contribute with important data because queries do not reach them.

1.2 Motivation

Data retrieval in a PDMS can be improved if peers are efficiently disposed in the overlay network according to some kind of organization such as, for example, a semantic-based approach [Castano and Montanelli, 2005; Li and Vuong, 2007]. In this context, the notion of *semantic community* of peers (community, for short) is of great importance, since it aims at logically approximating peers with common interests about a specific topic [Castano and Montanelli, 2005]. In a semantic community, when a query is posed at a peer the query is transmitted only among the other peers of that community. In short, semantic communities enforce sharing of distributed resources and semantic collaboration in an effective way [Li and Vuong, 2005].

PDMS that employ the notion of semantic communities can obtain several benefits. For instance, schema mappings are established between semantically similar peers. Since the scope of a query is restricted to the community where it has originally been posed, the query is answered only by a few (but relevant) peers [Castano and Montanelli, 2005]. Thus, query processing tends to be improved in a PDMS. Such kind of peer organization can offer other benefits: (i) increase system scalability as the number of messages transmitted through the network is minimized; and (ii) avoid unnecessary processing effort and storage space in participating peers.

Due to the dynamic behavior of peers, the creation and maintenance of semantic communities is a challenging issue in the current stage of development of PDMS [Castano and Montanelli, 2005]. The availability of advanced techniques for query propagation on a semantic basis is another relevant issue. In this sense, ontologies are more and more employed for describing the knowledge shared by peers [Nejdl *et al.*, 2002; Castano *et al.*, 2003; Xiao, 2006; Li and Vuong, 2007]. As a result, ontology matching techniques [Giunchiglia *et al.*, 2004; Hai, 2005; Castano *et al.*, 2006; Hu and Qu, 2008] are required to deal with the different concept meanings in the ontology-based schemas provided by different peers.

1.3 Objectives

Our main assumption in this work is that the establishment of schema mappings and consequently query processing can be improved if semantically similar peers are logically grouped in a PDMS overlay network. In this sense, the main goal of this thesis is to propose a process for clustering peers in a PMDS. To achieve this objective, we propose a PDMS architecture which is designed to facilitate the connection of peers according to their corresponding exported schema (i.e. an ontology).

Peer connection in the proposed PDMS is mainly an incremental clustering process. When a new peer arrives, it searches for a community where peers share schemas (ontologies) belonging to the same knowledge domain. Then, within a community, the new peer joins an existing semantic cluster (cluster, for short) where peers share similar schemas. A semantic cluster is represented by a cluster ontology which describes the schemas of the peers within the cluster. In addition, each cluster maintains a link to its semantic neighbors in the overlay network, i.e. to other semantically similar clusters.

Before a new peer joins a semantic cluster it is necessary to determine the similarity between them. To this end, we propose a global similarity measure that indicates the similarity degree between the ontology of the new peer and a cluster ontology. Such measure is obtained from an ontology matching process which also produces an ontology alignment between the elements of the matching ontologies.

In order to avoid the matching against all existing clusters, an initial cluster is provided to each new peer. From such initial cluster, the search for a semantically similar cluster is started by following the semantic neighbors of the initial cluster in the overlay network. The initial cluster is provided from a semantic index where each entry corresponds to a summary of a cluster ontology. Thus, in this thesis, we also propose an automatic process for summarizing cluster ontologies.

1.4 Expected Contributions

This thesis presents the following contributions:

- The specification of a semantic-based architecture for a Peer Data Management System (PDMS);
- The specification, implementation, and validation of an ontology matching process that considers, besides the traditional terminological and structural matching techniques, a semantic-based one;
- The specification, implementation, and validation of an automatic process to summarize ontologies;
- The specification, implementation, and validation of an incremental process for clustering peers in the proposed PDMS.

1.5 Thesis Outline

The reminder of this thesis is organized as follows.

- **Chapter 2** is divided into two distinct parts which are essential to the comprehension of this thesis. The first part describes ontologies and some related processes such as ontology matching, ontology merging, and database schema to ontology mapping. The second part offers an overview of clustering and discusses incremental clustering algorithms. In addition, it presents the commonly used criteria to evaluate a clustering result;
- **Chapter 3** discusses Peer Data Management Systems (PDMS) and its main data management issues: schema mappings and query processing. It emphasizes the importance of organizing peers according to a semantic-based approach and presents the mains challenges to achieve this goal. The chapter also describes some existing PDMS that propose a semantic-based approach to organize peers in an overlay network. Finally, it presents a comparative analysis of such PDMS;
- **Chapter 4** describes the importance of using ontologies in PDMS. It also presents the specification our Ontology-based PDMS (OPDMS), whose overlay network is mainly designed to assist the organization of peers according to their exported schema. Such specification comprises a detailed description of the system's architecture, the distinct types of peers, as well as their internal modules, and the different types of ontologies used in the

system. Moreover, the specification describes how schema mappings and query processing are handled in the proposed PDMS;

- **Chapter 5** describes a semantic-based ontology matching process in which the resulting correspondences are generated as a combination of linguistic, structural and semantic matching algorithms. The correspondences are used to compute a global similarity measure between two (peer) ontologies which is used for clustering semantically similar peers in the proposed PDMS. Particularly, the semantic matcher is described including the rules that are applied to identify the semantic relationships between ontology elements. To clarify matters, it presents a case study illustrating how the global measure is used. An experimentation of the ontology matching process with some obtained results is also provided;
- **Chapter 6** presents an automatic process to summarize cluster ontologies. Firstly, an overview of the proposed summarization process and a notation to represent ontologies are provided. Then, the two measures (centrality and frequency) used to determine the relevance of ontology concepts are described. Next, the chapter presents the proposed process to summarize ontologies, the summarization algorithm, and an illustrative example. The results of applying the proposed process to real world ontologies according to different criteria are exposed. Finally, related work is discussed.
- **Chapter 7** presents an ontology-based process for clustering peers in the proposed PDMS. It describes how a new peer searches for a particular semantic community in the overlay network. Besides, it presents the main requirements for clustering peers within a semantic community and the incremental clustering algorithm used to manage semantic clusters. The main steps to connect a new peer and disconnect a participating peer are detailed. Considerations about maintaining the semantic clusters are also presented. Finally, experimental results are shown and discussed.
- **Chapter 8** concludes the thesis stating our research contributions and some future works.

CHAPTER 2

BACKGROUND

"Childhood is measured by sounds, smells and sights,
before the dark hour of reason grows"
John Betjeman

This chapter offers an overview of *ontologies* and *clustering*, two essential issues that are needed to understand this thesis. Basically, the chapter is divided into two parts. The first one describes ontologies and its main elements (e.g. concepts and properties). Heterogeneity problems between distinct ontologies are discussed as well as some related ontology management processes, e.g. ontology matching, ontology merging, and database schema to ontology mapping. The second part of this chapter offers a definition of clustering and discusses some important clustering issues such as object set availability and sensitivity to input order. It also describes some incremental clustering algorithms. Finally, the chapter presents classical criteria to evaluate the results of a given clustering algorithm.

2.1 Ontologies

The most quoted definition of ontology is "*an explicit specification of a conceptualization*" [Gruber, 1993]. A *conceptualization* is an abstract model that describes objects, concepts, and other elements, particular to some (usually restricted) domain, as well as relationships that hold between these elements. An *explicit specification* means that the elements and relationships in the abstract model are given explicit names and semantics, expressed in some *formal language*. In practice, this usually means a logic-based language, as it allows for automated reasoning [Noy, 2004a]. Ontologies were developed by the Artificial Intelligence community to facilitate knowledge sharing and reuse [Guarino, 1998]. They have been used as a fundamental concept in the Semantic Web[1] [Berners-Lee *et al.*, 2001; Staab and Stuckenschmidt, 2006]. Carrying semantics for particular domains, ontologies are largely used for representing domain knowledge[2].

Ontologies are expressed in an ontology language. There is a large variety of languages for describing ontologies [Staab and Studer, 2004], including: DAML+OIL, RDF, and OWL. Most of these languages share the same kinds of elements, often with different names but comparable interpretations. For instance, OWL [Smith *et al.*, 2004] is a semantic markup language for publishing and sharing ontologies on the Web. In the last years, it has become the most recommended ontology language for representing knowledge in the Semantic Web. Basically, OWL deals with the following kinds of elements [Smith *et al.*, 2004]:

- **Concepts or Classes** are the main elements of an ontology. These are interpreted as a set of individuals in the domain. They are introduced in OWL by the *owl:Class* construct;
- **Individuals or objects or instances** are interpreted as a particular individual of a domain. These are introduced in OWL by the *owl:Thing* construct;
- **Relations** are the ideal notion of a relation independently to what it applies. Relations are interpreted as a subset of the cartesian product of the domain. These are introduced in OWL by the *owl:ObjectProperty* or *owl:DatatypeProperty* constructs;
- **Datatypes** are particular parts of the domain that specify values as opposed to individuals. However, values do not have identities. *String* and *Integer* are examples of datatypes;
- **Data values** are simple values.

[1] The Semantic Web W3C Initiative, http://www.w3.org/2001/sw/
[2] The National Cancer Institute Ontology, http://www.mindswap.org/2003/CancerOntology/nciOncology.owl

Moreover, in OWL, elements can be connected by various kinds of relations, including:

- **Specialization** between two classes or two properties is interpreted as the inclusion of their interpretations. Specialization is introduced in OWL by the *rdfs:subClassOf* or *rdfs:subPropertyOf* constructs;
- **Exclusion** between two classes or two properties is interpreted as the exclusion of their interpretations, i.e. when their intersection is empty. Exclusion is introduced in OWL by the *owl:disjointWith* construct;
- **Instantiation or typing** between individuals and classes, property instances and properties, values and datatypes is interpreted as membership. Instantiation is expressed in OWL with the *rdf:type* construct.

Ontology interpretation is not left to the users that read the diagrams or to the knowledge management systems implementing them, it is specified explicitly [Euzenat and Shvaiko, 2007]. The semantics provides the rules for interpreting the syntax which do not provide the meaning directly but constrains the possible interpretations of what is declared. The semantics of ontologies can be constrained by additional axioms.

2.1.1 Types of Heterogeneity

Over the last years, ontologies have become one of the most common ways of expressing knowledge in different distributed and opened applications [Euzenat and Shvaiko, 2008], e.g. semantic P2P systems and multi-agent systems. In such systems, the content shared by the actors is commonly described by ontologies. Since actors have different interests, use different pieces of knowledge, and, most often, reason at different levels of detail, heterogeneity in such systems cannot be avoided. These characteristics lead to diverse kinds of heterogeneity (even in the same knowledge domain) and, therefore, should be carefully taken into consideration.

Heterogeneity does not lie solely in the differences between goals of the applications according to which they have been designed or in the expression formalisms in which ontologies have been encoded. Some of the most obvious types of heterogeneity include: (i) *syntactic heterogeneity*: occurs when two ontologies are not expressed in the same ontology language; (ii) *terminological heterogeneity*: occurs due to variations in names when referring to the same elements in different ontologies; and (iii) *conceptual heterogeneity*: also called semantic heterogeneity [Euzenat, 2001], stands for the differences in modeling the same domain of interest. This can happen due to the use of different axioms for defining concepts or due to the use of totally different concepts.

In the literature, there have been many different classifications to types of heterogeneity [Batini *et al.*, 1986; Kashyap and Sheth, 1998; Euzenat, 2001]. To deal with such types of heterogeneity between ontologies is the goal of ontology matching.

2.1.2 Ontology Matching

There have been several definitions for the expression *ontology matching* [Doan *et al.*, 2003; Zhdanova and Shvaiko, 2006; Ehrig, 2007; Euzenat and Shvaiko, 2007; Zhang *et al.*, 2008]. According to [Euzenat and Shvaiko, 2007], ontology matching is the process of finding relationships or correspondences between elements of two distinct ontologies (denoted O_i and O_j), generally describing the same or similar domains.

The output of such process is called *ontology alignment* (alignment, for short) and is denoted by A'. An ontology alignment A' contains a set of correspondences indicating which elements of the two ontologies logically correspond to each other (i.e. match). An overview of the ontology matching process is illustrated in Figure 2.1. Optionally, some other parameters can be introduced into the matching process: an *input alignment*, A, which is to be completed by the process; *matching parameters*, P, e.g. weights and thresholds; and *external resources*, R, used during the matching process, e.g. common knowledge or domain specific thesauri.

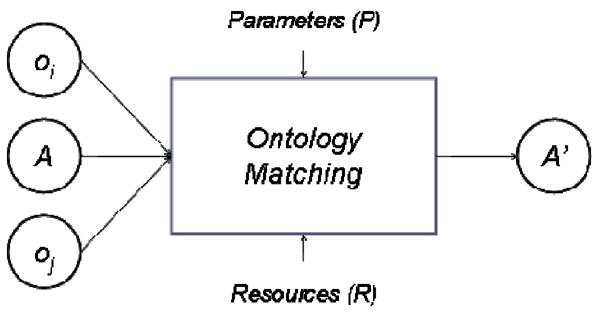


Figure 2.1. The ontology matching process [Euzenat and Shvaiko, 2007].

The correspondences can be produced by one or more matching algorithms (or *matchers*) which are executed sequentially or in parallel. The correspondences are expressed as *relationships* (e.g. equivalence, subsumption, and disjointness) as well as *similarity values* between 0 (strong dissimilarity) and 1 (strong similarity). Similarity values (or level of confidence) can be viewed as a measure of trust in the case that the correspondence holds. Alignments are used for various tasks, including ontology merging, query processing, and data translation.

Formally, each correspondence of an alignment A' can be defined as a 5-tuple: $\langle id, e_i, e_j, r, n\rangle$, where *id* is a unique identifier of the given correspondence; e_i and e_j are the two matched elements (with $e_i \in O_i$ and $e_j \in O_j$); r is the relationship holding between e_i and e_j; and n expresses the level of confidence underlying such correspondence [Euzenat and Shvaiko, 2007].

Although ontology elements can be related by different types of relationships (e.g. equivalence, subsumption, or disjointness) most of the ontology matching algorithms mainly consider the equivalence relationship ($\equiv$), meaning that the matched elements are the same. As a result, the relationship type is commonly omitted (equivalence is assumed) and correspondences are frequently resumed to a 4-tuple: $\langle id, e_i, e_j, n\rangle$.

Ontology Matching Techniques

In order to solve the matching problem, several matching techniques have been proposed [Batini *et al.*, 1986; Larson *et al.*, 1989; Kashyap and Sheth, 1996; Parent and Spaccapietra, 1998; Rahm and Bernstein, 2001; Wache *et al.*, 2001]. These works address the matching problem from different perspectives, e.g. Artificial Intelligence, Information Systems, and Databases. [Shvaiko and Euzenat, 2007] have attempted to consider the above mentioned works together in order to provide a general classification of matching techniques, focusing on schema-based matching methods.

The classification distinguishes between elementary (or basic) matching techniques and composition of techniques. Elementary techniques comprise:

- **String-based techniques**: consider the names and name descriptions of ontology elements;

- **Language-based techniques**: consider names as words in some natural language, e.g. French;
- **Constraint-based techniques**: consider the internal constraints being applied to the definitions of elements, e.g. types, cardinality of attributes, and keys;
- **Alignment reuse**: represent an alternative way of exploiting external resources, which record alignments of previously matched ontologies;
- **Upper level and domain specific formal ontologies**: upper level ontologies can be used as external sources of common knowledge, while domain specific formal ontologies can be used as external sources of background knowledge;
- **Graph-based techniques**: consider the ontology elements or their instances to compare their relationships with other elements or their instances;
- **Taxonomy-based techniques**: are also graph algorithms which consider only the specialization relationship;
- **Repository of structures**: unlike alignment reuse, it stores only similarities between ontologies in order to avoid the matching operation over dissimilar ontologies;
- **Model-based techniques**: handle the input based on its semantic interpretation;
- **Data analysis and statistics techniques**: take advantage of a representative sample of a population in order to find similarities and discrepancies.

For classifying the elementary techniques, [Shvaiko and Euzenat, 2007] have introduced two synthetic classifications: (i) *Granularity/Input Interpretation*, is based on the matcher granularity (element-level or structure-level) and on how the techniques generally interpret the input information; and (ii) *Kind of Input*, is based on the kind of input which is used by the elementary matching techniques.

In the *Granularity/Input Interpretation* classification, elementary techniques are distinguished according to the following classification criteria:

- **Element-level vs. structure-level**: element-level matching techniques compute correspondences by analyzing elements or instances of those elements in isolation, ignoring their relations with other entities or their instances. Structure-level techniques compute correspondences by analyzing how elements or their instances appear together in a structure;

- **Syntactic vs. external vs. semantic**: the key characteristic of syntactic techniques is that they interpret the input with regard to its sole structure following some clearly stated algorithm. External are the techniques exploiting auxiliary (external) resources of a domain and common knowledge in order to interpret the input. These resources may be human input or some thesaurus expressing the relationships between terms. Semantic techniques use some formal semantics to interpret the input and justify their results.

The *Kind of Input* classification is concerned with the type of input considered by a particular technique:

- The first level is categorized depending on which kind of data the algorithms work on: strings (terminological), structure (structural), models (semantics), or data instances (extensional). The two first ones are found in the ontology descriptions. The third one requires some semantic interpretation of the ontology and usually uses some semantically compliant reasoner to deduce the correspondences. The last one constitutes the actual population of an ontology;
- The second level of this classification decomposes further these categories if necessary: terminological methods can be string-based (considering the terms as sequences of characters) or based on the interpretation of these terms as linguistic objects (linguistic). The structural methods category is split into two types of methods: those which consider the internal structure of entities, e.g., attributes and their types (internal), and those which consider the relationship of entities with other entities (relational).

The overall classification of Figure 2.2 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kinds of manipulated objects) manner in order to reach the *Basic Techniques* layer.

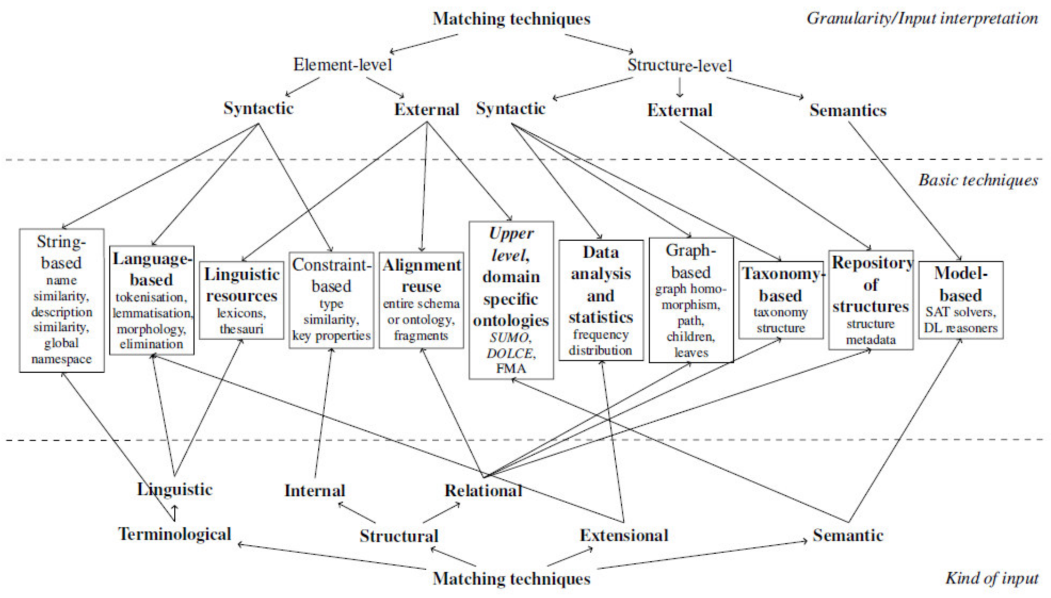


Figure 2.2. Classifications of elementary matching approaches [Shvaiko and Euzenat, 2007].

Matching Composition

The elementary matching techniques are the building blocks on which a matching solution is built. Once the similarity between ontology elements is available, the alignment remains to be computed. Thus, building a matching system involves some aspects such as (i) organising the combination of various similarities or matching algorithms; and (ii) aggregating the results of the basic methods in order to compute a combined similarity value between elements.

So far, we have presented the matching process as producing an alignment between two ontologies (Figure 2.1). A natural way of composing the basic matchers consists of improving the matching through the use of *sequential composition* (Figure 2.3). For instance, one might like to first use a matcher based on labels before running a semantic matcher.

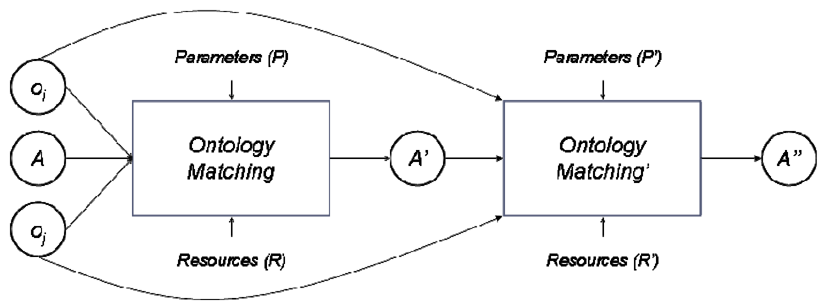


Figure 2.3. The sequential composition of matchers [Euzenat and Shvaiko, 2007].

Another way to combine matchers consists of running several different algorithms independently and aggregating their results: this is called *parallel composition* (Figure 2.4). Aggregation techniques can be very different: it can correspond to choosing one of the results on some criterion or merging their results through some operator. For instance, it can consist of running several matching algorithms in parallel and selecting the correspondences which are in all of them (intersection is then used as an aggregation operator) or selecting all the correspondences with their highest confidence.

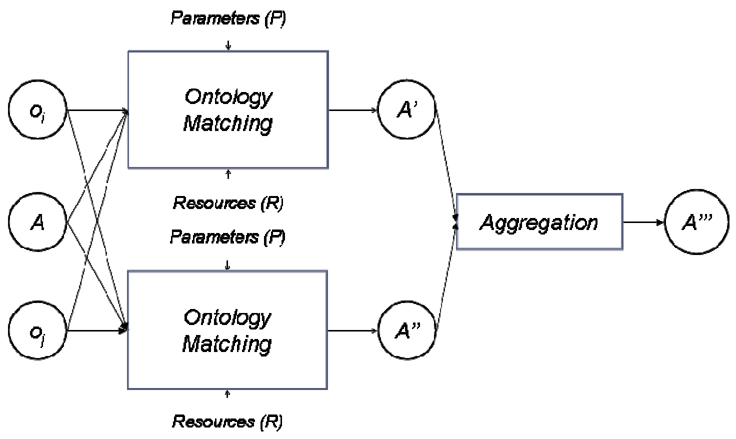


Figure 2.4. The parallel composition of matchers [Euzenat and Shvaiko, 2007].

Combined similarity is concerned with the aggregation of individual similarity values. Ontology elements such as classes are very often involved in many different relationships. For instance, computing the similarity between two classes requires the aggregation, in a single similarity measure, of the similarity obtained from their names, the similarity of their superclasses, as well as the similarity of their instances and that of their properties. In this sense, to calculate a combined similarity it is common to apply aggregation operators on the individual similarity values, e.g. maximum, minimum, average, and weighted sum [Aumüller *et al.*, 2005].

Matching Tools

Several ontology matching tools have emerged during the last years. Each one exploits a particular matching technique or a combination of them. A comparison of matching systems can be found in [Rahm and Bernstein, 2001;

Noy, 2004b; Doan and Halevy, 2005]. Particularly, a comparison of ontology matching tools is presented in [Euzenat and Shvaiko, 2007].

The increasing number of systems available for ontology matching has created a need to establish a consensus for evaluation of these systems. In 2006, the *Ontology Alignment Evaluation Initiative*[3] (OAEI) was created with the goal of organizing evaluation campaigns aiming at evaluating ontology matching systems. OAEI campaigns consist of applying matching systems to ontology pairs and evaluating their results. A systematic benchmark series has been produced in order to identify the areas in which each ontology matching algorithm is strong or weak. Anyone developing ontology matchers can participate by evaluating their systems and sending the results to the organizers.

In this section, our purpose is not to compare ontology matching tools in full detail, but rather to demonstrate how the matching techniques have been exploited. To this end, we summarize some of the most prominent tools: *COMA/COMA++* [Do and Rahm, 2002; Aumüller *et al.*, 2005; Hai, 2005], *H-Match* [Castano *et al.*, 2006], *Falcon-OA* [Hu and Qu, 2008], and *S-Match* [Giunchiglia and Shvaiko, 2003; Giunchiglia *et al.*, 2004].

COMA/COMA++ (University of Leipzig)

COMA (COmbination of Match Algorithms) [Do and Rahm, 2002] is a schema matching system supporting different applications and multiple schema formats, e.g. ontologies and relational schemas. It provides an extensible library of ontology matching algorithms which are executed in parallel and combined according to different strategies. Most of them implement string-based techniques. A matcher implements a thesaurus look-up (table of synonyms). The reuse-oriented matcher tries to reuse previously obtained correspondences for entire new schemas or for their fragments. The tool can be used as an evaluation platform to systematically examine and compare the effectiveness of different matchers and combination of matchers. COMA presumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of the match. COMA++ [Aumüller *et al.*, 2005; Hai, 2005] is implemented on top of COMA by elaborating in more detail the

[3] http://oaei.ontologymatching.org/

alignment reuse operation. The tool offers a more efficient implementation of the COMA matching algorithms and a repository of alignments. Also, it provides a generic data model to uniformly support ontologies expressed in different languages.

H-Match (Università degli Studi di Milano)

H-Match [Castano *et al.*, 2006] is an ontology matching algorithm for dynamically matching distributed ontologies. The semantic affinity between ontology concepts is evaluated by exploiting both the linguistic features of the concepts (linguistic affinity) and semantic relationships among them in an ontology (contextual affinity). The evaluation of the linguistic features is based on a thesaurus which is built by exploiting WordNet [Miller, 1995]. The meaning of each term used as a name of a concept or a property depends on the set of terminological relationships that it has with other terms in the thesaurus. Moreover, the meaning of a concept also depends on its properties and semantic relationships with other concepts in the ontologies. A weighted function is used to combine linguistic and contextual similarity values and thus produce a combined semantic affinity. By exploiting ontology descriptions, H-Match offers four different matching models: (i) *surface matching*: consider only the names of concepts; (ii) *shallow matching*: consider both concept names and concept properties; (iii) *deep matching*: consider concept names and the whole context of concepts (relationships); and (iv) *intensive matching*: consider, in addition to the features of the deep model, also property values.

Falcon-AO (China Southwest University)

Falcon-AO [Hu and Qu, 2008] is an automatic ontology matching system that helps realizing interoperability between (Semantic) Web applications that use different but related ontologies. It supplies a library of linguistic and structural matchers, and provides a robust combination of their alignments. One of the matchers implements a divide-and-conquer strategy to find block mappings between large-scale ontologies. The ontologies are adjusted before executing the matchers. In this sense, coordination rules are used to eliminate superfluous axioms and reduce structural heterogeneity between the ontologies to be matched. A similarity combination approach is used to gradually tune up the thresholds based on the measures of both the linguistic and the structural comparability. Linguistic matchers are first used for assessing the similarity between ontology concepts on the basis of their name and text annotations. If the result has a high confidence, then it is directly returned for extracting an alignment. Otherwise, the result is used as input for the structural matcher that tries to find an alignment on the basis of the relationships between concepts.

S-Match (University of Trento)

S-Match is a semantic-based matching tool which implements the idea of semantic matching as initially described in [Giunchiglia and Shvaiko, 2003]. A first version of the S-Match system is proposed in [Giunchiglia *et al.*, 2004]. Later the system has undergone several evolutions, including extensions of libraries of element- and structure-level matchers, adding alignment explanations as well as iterative semantic matching [Giunchiglia *et al.*, 2005; Giunchiglia *et al.*, 2006; Giunchiglia *et al.*, 2007]. S-Match takes as input two tree-like structures (e.g. classifications, XML schemas, and ontologies), and for each pair of nodes from the two trees, it returns a logic relationship (e.g. equivalence or subsumption), which is supposed to hold between the two nodes of the graphs. The relationships are determined by (i) expressing the ontology elements as logical formulas, and (ii) reducing the matching problem to a propositional validity problem. The elements are translated into propositional formulas which explicitly express the concept descriptions as encoded in the ontology structure and in external resources, such as WordNet [Miller, 1995].

Comparative Analysis

Table 2.1 is an excerpt from a comparison table presented in [Euzenat and Shvaiko, 2007]. It summarizes how the discussed ontology matching tools cover the matching techniques included in the *Kind of Input* classification discussed at the beginning of this section. Only the techniques which consider the conceptual part of an ontology are considered, i.e. linguistic, structural, and semantic. External resources are the methods used by linguistic matchers to interpret ontology descriptions. From such comparison, we can state some important remarks: (i) in general, each individual matching tool innovates on a particular aspect; (ii) matching techniques are not used in isolation; (iii) most tools employ a combination of the linguistic and structural matching techniques; and (iv) semantic techniques are rarely exploited.

Table 2.1. Matching techniques used by the different ontology matching tools [Euzenat and Shvaiko, 2007].

	Terminological	External Resource	Structural	Semantic
COMA / COMA++	String-based, Language-based, Data types	Dictionary of synonyms and Abbreviation tables, Alignment reuse, Repository of structures	DAG (tree)	-
H-Match	Domain compatibility, Language-based, Domains and ranges	Thesaurus	Matching of neighbors (thesaurus), Relationships	-
Falcon-AO	String-based	WordNet	Structural affinity	-
S-Match	String-based, Language-based	WordNet	-	Propositional SAT

2.1.3 Ontology Merging

Ontology merging is a first natural use of ontology matching [Noy and Musen, 2000; Davies *et al.*, 2006; Euzenat and Shvaiko, 2007]. Other uses include ontology transformation, data translation, mediation, and reasoning. The notion of ontology merging is closely related to that of schema integration in databases.

As depicted in Figure 2.5, the merging process consists in obtaining a new ontology O_k from two, possibly overlapping, source ontologies O_i and O_j. The matched entities in the source ontologies are related as prescribed by the alignment A' generated as a result of an ontology matching process. The source ontologies remain unaltered, along with ontology mappings between each source ontology and the merged ontology. Different kinds of ontology mappings can be defined between a merged ontology and the source ontologies, e.g. concept mappings and property mappings. The merged ontology contains the knowledge of the source ontologies. Merging can be presented as the following operator: $Merge\ (O_i, O_j, A') = O_k$.

When the source ontologies are expressed in the same language, merging often involves putting the ontologies together and generating bridge axioms. Such constructs correspond to formulas, in an ontology language, that express the alignments so that it is possible to integrate the elements of an ontology within one another. Merging does not usually require a total alignment: those elements which have no corresponding element in the other ontology will remain unchanged in the merged ontology. Ontology merging is especially used when it is necessary to carry out reasoning involving several ontologies.

Protégé [Noy and Musen, 2003; Noy, 2004b] and *Rondo* [Melnik *et al.*, 2003] are tools that offer independent operators for ontology merging. *OntoMerge* [Dou *et al.*, 2005] takes bridge rules expressed in predicate calculus and can merge ontologies in OWL. The *Alignment API* [Euzenat, 2004] can generate axioms in the languages OWL or SWRL for merging ontologies. Other systems are able to match ontologies and merge them directly: *FCA-merge* [Stumme and Mädche, 2001], *SKAT* [Mitra *et al.*, 1999], and *DIKE* [Palopoli *et al.*, 2003]. *OntoBuilder* [Modica *et al.*, 2001] uses ontology merging as an internal operation: the system creates an ontology that is mapped to query forms. This ontology is merged with a global ontology so that queries can be directly answered from such global ontology.

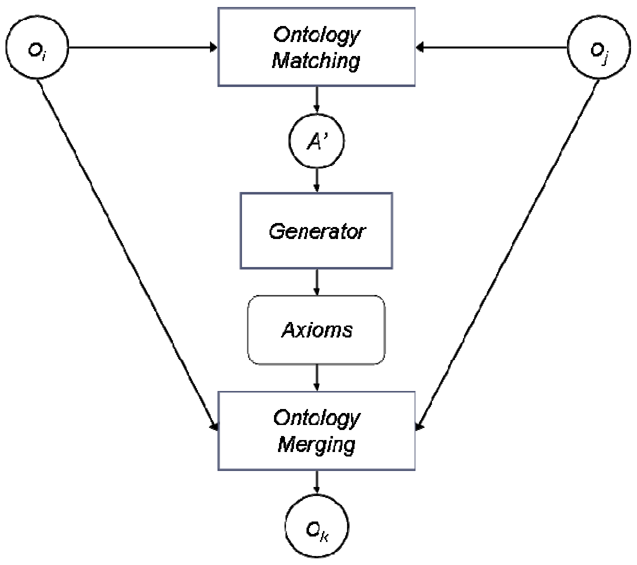


Figure 2.5. Ontology merging: from two matched ontologies O_i and O_j, resulting in an alignment A'. This allows the creation of a new ontology O_k covering the matched ontologies [Shvaiko and Euzenat, 2007].

2.1.4 Database Schema to Ontology Mapping

The popularity of ontologies is rapidly growing since the emergence of the Semantic Web [Berners-Lee *et al.*, 2001]. To date, the amount of available Web ontologies continues increasing at a phenomenal rate. However, most of the data sources available on the Web still store their data in other types of data sources, e.g. relational databases, XML documents, and web pages. Therefore, in order to achieve an efficient interoperability between heterogeneous data sources, an effective way is to discover mappings between relational database schemas and ontologies which can better express the semantic of data sources [Ghawi and Cullot, 2007].

Database to ontology mapping is the process whereby a database schema and an ontology are semantically related at a conceptual level, i.e. correspondences are established between the database schema elements and the ontology elements [Ghawi and Cullot, 2007]. Such process is considered an interdisciplinary research in both Database and Semantic Web communities [Hu and Qu, 2007]. Currently, there are many approaches and tools to deal with database to ontology mapping. Basically, they can be classified into two main categories: (i) *approaches for creating a new ontology from a database* [Nyulas *et al.*, 2007]; and (ii) *approaches for mapping a database to an already existing ontology* [Ghawi and Cullot, 2007].

In the first category, an ontology model is initially created from a relational database model and then the contents of the database are migrated to the generated ontology. The mappings are simply the correspondences between each created ontology element (e.g. concept and property) and its original database schema element (e.g. table and column). In these approaches, the database model and the generated ontology are very similar. Mappings are quite direct and complex mapping situations do not usually appear. The creation of the ontology structure may be straightforward, involving direct transformations of database tables to ontology concepts and columns into properties. This type of direct mapping is not sufficient for expressing the full semantics of the database domain. The creation of the ontology structure may require the discovery of additional semantic relations between database elements (like the referential constraints) and take them into account while constructing ontology concepts and relations between them. In the first category, we can note some relevant projects: *DataMaster* [Nyulas *et al.*, 2007], a Protégé plug-in that allows to automatically import schema definition and data into Protégé; and *Relational.OWL* [de Laborda and Conrad, 2005], an OWL ontology representing abstract schema elements of relational databases.

The approaches for mapping a database to an already existing ontology consider that an ontology and a legacy database already exist. The goal is to create mapping between them, and/or populate the ontology by the database contents. Mappings here are more complex than those in the previous case because different levels of overlap between the database domain and the ontology's one can be found, and those domains do not always coincide because the modeling criteria used for designing databases are different from those used for designing ontology models [Barrasa *et al.*, 2004]. In this category, several languages have been proposed to formally express database to ontology mappings: *D2R map* [Bizer, 2003], a declarative XML-based language to describe mappings between relational database models and ontologies implemented in RDFS; and *R2O* [Barrasa *et al.*, 2004], another declarative language that describes mappings between database schemas and ontologies.

Besides languages, mapping approaches include some tools like *KAON Reverse*[4], a prototype for mapping relational database content to ontologies; *Vis-A-Vis* [Fuxman *et al.*, 2006], a Protégé plug-in that allows the mapping of relational databases to existing Protégé ontologies; *DB2OWL* [Cullot *et al.*, 2007], a tool that automatically generates ontologies from database schemas as well as mappings that relate the ontologies to the information sources; and *RDBToOnto* [Cerbah, 2008], a user oriented tool that supports the complete transitioning process from access to the input databases to generation of populated ontologies. A complete comparison of database to ontology matching approaches can be found in [Ghawi and Cullot, 2007].

2.2 Clustering

In the following sections, we present an overview of *clustering* starting with its definition and the similarities with the classification problem. We also describe object set availability and sensitivity to input order, two clustering issues which are associated to the main problem dealt in this thesis: clustering peers in a PDMS. Then, incremental clustering algorithms are discussed. Finally, we describe some of the most commonly used criteria to evaluate and assess the results of a clustering algorithm: external and internal criteria.

[4] http://kaon.semanticweb.org/alphaworld/reverse/view

2.2.1 Definition

Classification is the process of organizing and categorizing objects (or data) in distinct classes [Xu and Wunsch, 2005]. In classical object classification, a collection of labeled (possibly pre-classified) objects is provided and the problem is to label a newly encountered, yet unlabeled, object. Typically, the given labeled objects are used to learn the descriptions of classes which in turn are used to label a new object. In general, classical object classification is based on some general principles: (i) a model is initially created based on object distribution; (ii) the model is then used to classify new objects; and (iii) given the model, a class can be predicted for new objects.

As illustrated in Figure 2.6, classification problems can be subdivided into two different categories: *non-exclusive* or *exclusive*. In a non-exclusive classification, an object can be assigned to several classes. Differently, in an exclusive classification, an object is assigned to exactly one class. Furthermore, an exclusive classification can be subdivided into *supervised* or *unsupervised*. In the former, the class labels and the number of classes are known in advance whereas in the latter such information is not available.

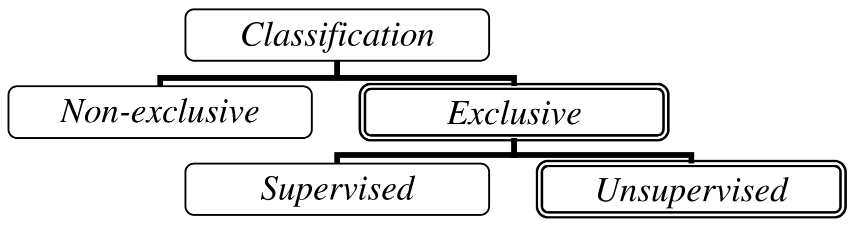


Figure 2.6. A taxonomy of classification types.

Clustering [Theodoridis and Koutroumbas, 2003; Gan *et al.*, 2007] is a type of exclusive and unsupervised classification. It is an automatic process of partitioning a finite set of objects in a set of meaningful sub-classes, called *clusters*. A cluster is a collection of objects that are “similar” to one other and thus can be treated collectively as one group [Berkhin, 2002]. The problem of clustering is to group a given collection of unlabeled objects into meaningful clusters. In a sense, labels are also associated with clusters, but these categories of labels are object-driven, i.e. they are obtained solely from the objects.

Clustering helps users to understand the natural grouping or structure in an object set. A good clustering method will produce high quality clusters in which the intra-class (i.e. intra-cluster) similarity is high and the inter-class similarity is low. The quality of a clustering result depends on the similarity or distance measure used by the method and its implementation. When implementing a clustering algorithm on a computer, attention must be paid to questions related to computational efficiency.

In the last decades, clustering techniques have been extensively studied in different research areas, e.g. statistics, machine learning, and data mining. Many clustering algorithms have been proposed in the literature. In general, such algorithms can be classified into several clustering categories: *partitioned* [Hartigan and Wong, 1979], *hierarchical* [Jain *et al.*, 1999], *density-based* [Ester *et al.*, 1996], *grid-based* [Wang *et al.*, 1997], and *model-based* [Cheeseman and Stutz, 1996]. An excellent survey of these clustering categories can be found in [Jain *et al.*, 1999]. Some advances in clustering are presented in [Kotsiantis and Pintelas, 2004].

2.2.2 Clustering Issues

The importance of clustering issues varies according to the area of study and the type of problem to be solved. For example, in data mining some of the main explored issues are: scalability to large datasets, time complexity, and interpretability of results [Berkhin, 2002]. For the purpose of clustering peers in a PDMS some important clustering issues that have to be considered are: *object set availability* and *sensitivity to input order*.

The first issue is related to the availability of the objects in an object set. Basically, there are two main approaches to perform clustering. The first one is to consider the object set as a whole and begin to organize it into meaningful clusters. This is called *batch approach*. The second approach is based on the assumption that it is possible to add one object from the object set at a time to the clustering space. This is called *incremental approach* or *online approach* or *single-pass approach*.

Sensitivity to input order is one of the major issues in incremental clustering [Fisher *et al.*, 1992]. An incremental algorithm is *order-independent* if it generates the same partition for any order in which the data is presented. Otherwise, it is *order-dependent*. This property is illustrated in Figure 2.7 where there are 6 two-dimensional objects labeled 1 to 6. If these objects are presented to an incremental clustering algorithm such as the Leader algorithm [Hartigan, 1975] in the order 2, 1, 3, 5, 4, 6 then the two clusters obtained are shown by ellipses. If the order is 1, 2, 6, 4, 5, 3, then two clusters are generated as shown by the triangles.

Since in PDMS peers arrive one at a time, incremental clustering algorithms are the main basis for the peer clustering approach proposed in this thesis. Thus, in the next section, we characterize some of the most popular incremental clustering algorithms available in the literature.

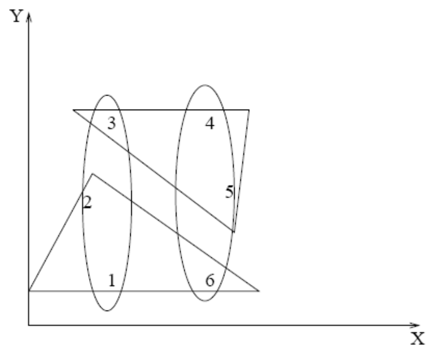


Figure 2.7. The leader algorithm is order-dependent [Jain *et al.*, 1999].

2.2.3 Incremental Clustering Algorithms

Incremental clustering algorithms are based on the assumption that it is possible to consider objects one at a time to existing clusters. A new object is assigned to a cluster without significantly affecting the existing clusters. Basically, incremental clustering algorithms follow the high level description described as follows [Jain *et al.*, 1999].

```
(1) Assign the first object to a cluster.
(2) Consider the next object. Either assign this object to one of the
existing clusters or to a new cluster. This assignment is done based on
some criterion, e.g. the similarity between the new object and the existing
cluster centroids; and a threshold.
(3) Repeat step 2 until all the objects are clustered.
```

The number of comparisons of incremental clustering for n objects is $O(n^2)$. In the worst case, each object creates a new cluster so each object must be compared to all the others. If the number of clusters is m, the number of comparisons time is $O(nm)$, and if m is bounded by a constant, $O(n)$.

Some important issues have to be considered concerning incremental algorithms. The first one is the selection of the threshold value. In some cases, it defines the range around the cluster centroid[5] in which objects have to rely for being integrated into a cluster. This is comparable to the problem of the user-specified number of clusters in non-incremental methods such as the k-means algorithm [Hartigan and Wong, 1979]. The second issue is the sensitivity to the input order. Typically, clustering results differ significantly for objects presented in different sequences. The third issue is the information loss due to the abstraction model chosen to summarize a cluster. Finally, the last issue is related to efficiency. In many incremental algorithms, including the high level description contained in [Jain *et al.*, 1999], before a new object is allocated to a cluster it is compared with all existing clusters. In some cases, it can be unacceptable in terms of scalability and performance.

Over the last years, several incremental clustering algorithms have been proposed in the literature: *Leader* [Hartigan, 1975], *BIRCH* [Zhang *et al.*, 1997], *Bubble* and *Bubble-FM* [Ganti *et al.*, 1999], *COBWEB* [Fisher, 1987], and *CLASSIT* [Gennari *et al.*, 1989]. A high level description of the Leader algorithm [Hartigan, 1975] is described as follows.

```
(1) Let s be a similarity threshold.
(2) Let the first object assigned to cluster $C_1$ be the defining object, $c_1$
(3) For each object $c_{i+1}$
      (3.1) Find the closest cluster, $C_j$
      (3.2) If $dist(c_{i+1},C_j) > d$, add $c_{i+1}$ to $C_j$
      (3.3) Otherwise, create a new cluster with the defining object $c_{i+1}$.
```

BIRCH [Zhang *et al.*, 1997] uses a hierarchical data structure called *CF-tree* for partitioning the incoming data points in an incremental and dynamic manner. BIRCH can typically find a good clustering with a single scan of the data and improve the quality further with a few additional scans. BIRCH is order-dependent as it may generate different clusters for different orders of the same input data. Bubble and Bubble-FM [Ganti *et al.*, 1999] clustering algorithms are extensions of BIRCH to general metric spaces (categorical values in attributes).

COBWEB [Fisher, 1987] is a popular hierarchical clustering algorithm for categorical data. It dynamically builds a classification tree by processing one data point at a time. Each cluster is associated with conditional probabilities for categorical attribute-values pairs. During the classification tree construction,

[5] The object which represents the central point of a cluster.

every new data point is descended along the tree and the tree is potentially updated. Decisions are based on an analysis of a category utility. There is a similar incremental hierarchical algorithm for all numerical attributes called CLASSIT [Gennari *et al.*, 1989]. CLASSIT associates normal distributions with cluster nodes. Both algorithms can result in highly unbalanced trees.

All the above discussed algorithms are order-dependent. Except for the Leader algorithm all the other algorithms reconsider the existing clusters after a new object is inserted. However, to this end, they necessarily use some kind of centralized data structure (e.g. a tree) which offers a detailed and up-to-date view of all existing clusters at a certain moment. Unfortunately, due to many different reasons (e.g. scalability and performance), this kind of assumption cannot be considered in dynamic environments such as PDMS. Next, we describe some commonly used criteria to evaluate and assess the results of a clustering algorithm.

2.2.4 Cluster Validity

Clustering is an unsupervised process since there are no predefined classes and no examples that would indicate grouping properties in the data set [Jain *et al.*, 1999; Theodoridis and Koutroumbas, 2003]. The majority of the clustering algorithms behave differently depending on the features of the data set and the initial assumptions for defining clusters. Thus, in most applications the resulting clustering scheme requires some sort of evaluation regarding its validity. Evaluating and assessing the results of a clustering algorithm is the main subject of *cluster validity* [Batistakis *et al.*, 2002a; Batistakis *et al.*, 2002b]. In this section, we present the fundamental concepts of clustering validity. More specifically, we discuss the cluster validity approaches based on the classical *external* and *internal* criteria [Halkidi *et al.*, 2001; Toledo, 2005].

Fundamental Concepts

Let C denote the clustering structure resulting from the application of a clustering algorithm r on a data set X. Also, let N be the size of X, i.e. $N = |X|$. C may be a hierarchy of clusters, as is the case with the hierarchical algorithms, or a flat set of clusters. Cluster validity can be approached in two possible directions. First, we may evaluate C in terms of an independently drawn structure, which is imposed on X a priori and reflects our intuition about the clustering structure of *X*. The criteria used for the evaluation of this kind are called *external criteria*. In addition, external criteria may be used to measure the degree to which the available data confirm a pre-specified structure, without applying any clustering algorithm to *X*. Second, we may evaluate *C* in terms of quantities that involve the objects of *X* themselves, for example, the similarity matrix. The criteria used for this kind of evaluation are called *internal criteria*. Next, we present the fundamental considerations and the representative indices for the external and internal criteria.

External Criteria Measures

External criteria is used for the comparison of a clustering structure *C*, produced by a clustering algorithm, with a partition *P* of *X* drawn independently from *C*. In the following, we consider the validation task concerning a clustering structure, *C*, resulting from a specific clustering algorithm, in terms of an independently drawn partition *P* of *X*. Let $C = \{C_1,\ldots,C_m\}$ and $P = \{P_1,\ldots,P_s\}$. Note that the number of clusters in *C* need not be the same as the number of groups in *P*. Our goal is to define appropriate statistical indices to be used for the hypothesis test.

Let n_{ij}, denote the number of objects that belong to C_i and P_j. Consider a pair of objects (x_v, x_u). We refer to it as (a) **SS** if both objects belong to the same cluster in *C* and to the same group in *P*; (b) **DD** if both objects belong to different clusters in *C* and to different groups in *P*; (c) **SD** if the objects belong to the same cluster in *C* and to different groups in *P*; and (d) **DS** if the objects belong to different clusters in *C* and to the same group in *P*. Let ***a***, ***b***, ***c***, and ***d*** be the number of SS, SD, DS, and DD pairs of objects of *X*, respectively. Then, $a + b + c + d = M$, where ***M*** is the total number of possible pairs in *X*, i.e. $M = N * (N - 1) / 2$.

Let $m_1 = a + b$ be the number of pairs of objects that belong to the same cluster in *C* and $m_2 = a + c$ be the number of pairs of objects that belong to the same group in *P*. Using the preceding definitions, statistical indices (statistics, for short) can be defined in order to measure the degree to which *C* matches *P*. Such statistical indices are:

Rand Index [Theodoridis and Koutroumbas, 2003]: measures the fraction of the total number of pairs that are either in the same cluster and in the same partition, or in different clusters and in different partitions. The Rand index is defined as

$$R = \frac{a+d}{M} \tag{1}$$

Jaccard Coefficient [Batistakis *et al.*, 2002a]: measures the proportion of pairs that are in the same cluster and in the same partition from those that are either in the same cluster or in the same partition. The coefficient is defined as

$$J = \frac{a}{(a+b+c)} \tag{2}$$

Fowlkes-Mallows (FM) Index [Fowlkes and Mallows, 1983]: corresponds to the probability that two random objects are in the same cluster given they are in the same group, and the probability that two random objects are in the same group given they are in the same cluster. The FM Index is defined as

$$FM = \sqrt{\frac{a}{a+b}\frac{a}{a+c}} = \frac{a}{\sqrt{m_1 m_2}} \tag{3}$$

The values of these statistical indices are between 0 and 1. However, a requirement for achieving the maximum value is to have the same number of partitions and clusters (m = s), which, in general, is not always possible. For all the above defined indices, it is clear that the larger their value (1) the higher the agreement between *C* and *P*.

Another popular statistic that is frequently used in conjunction with external criteria is the Hubert's Γ statistic [Batistakis *et al.*, 2002b]. It measures the correlation between two matrices, *X* and *Y*, of dimension *N* x *N*, drawn independently of each other, where *X*(*i*,*j*) and *Y*(*i*,*j*) are the (*i*,*j*) elements of the matrices *X* and *Y*, respectively. High values of Γ indicate close agreement between *X* and *Y*. The normalized Hubert's Γ statistic is defined as

$$\Gamma^{\wedge} = \frac{Ma-(a+b)(a+c)}{\sqrt{(a+b)(a+c)(M-(a+b))(M-(a+c))}} \tag{4}$$

Internal Criteria Measures

In this approach, the goal is to verify whether the clustering structure produced by a clustering algorithm fits the data, using only information inherent in the data set. In this sense, one of the most used indices is the **Silhouette Index** [Rousseeuw, 1987]. Such index is useful when it is seeking compact and clearly separated clusters.

In order to construct silhouettes we need a partition of some clustering algorithm, and the similarity matrix containing all the similarities between objects. For a given cluster, this method assigns to each object of the cluster a quantitative measure $s(i)$, known as the *silhouette width*. The silhouette width indicates the membership of object i in the cluster it has been assigned. Let i be any object in the data set, and denote by C_j the cluster to which object i has been assigned. Let $a(i)$ the average dissimilarity between i and all the other objects in cluster C_j. Consider any cluster C_k different to cluster C_j, and compute $b(i) = \min C_k \neq C_j\ d(i, C_k)$ ($k = 1, 2, \ldots, c$; $k \neq j$). Then, the silhouette width is defined as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{5}$$

A neighbor of object i is the cluster C_k for which the minimum is obtained, that is, $d(i,C_k) = b(i)$. Cluster C_k represents the second best choice for object i. From the equation (5) we can see that $-1 \leq s(i) \leq 1$. A value of $s(i)$ close to 1 is obtained when the dissimilarity $a(i)$ is much smaller than the smallest between dissimilarity $b(i)$. Therefore we can say that object i is well clustered. On the other hand, if $s(i)$ take values close to -1 implies that $a(i)$ is much larger than $b(i)$. In this case, we can say that object i has been misclassified, so object i may be reassigned. If $a(i)$ and $b(i)$ have similar values then $s(i)$ is about zero. In this situation, object i lies equally far away from both cluster C_j and C_k. If the data consist of similarities and $a'(i)$ and $d'(i,C)$ represent the corresponding average similarities, then $b'(i) = \max_{C \neq A} d'(i,C)$. The interpretation is in the same way as before. In this case, the silhouette width is defined as

$$s(i) = \frac{a'(i) - b'(i)}{max\{a'(i), b'(i)\}} \tag{6}$$

It is also possible to calculate a cluster silhouette S_j, called *average silhouette width*, which represents the heterogeneity of cluster C_j. This quantitative measure can be obtained using:

$$S_j = \frac{1}{m} \sum_{i=1}^{m} s(i) \tag{7}$$

We can also consider an overall or *global silhouette width* denoted by GS_u, and define as:

$$GS_u = \frac{1}{c}\sum_{j=1}^{c} S_j \qquad (8)$$

where U is any partition, $U \leftrightarrow C : C_i \cup C_2 \cup \ldots \cup C_c$.

2.3 Considerations

In this chapter, we have presented an overview of ontology and clustering. Both concepts form the main basis of this work. Ontologies are mainly used to describe the content exported by peers in a PDMS. Clustering techniques are used to organize peers in an overlay network according to their exported ontology. We have also discussed two classical criteria used to evaluate the results of a given clustering algorithm: external and internal. Both criteria are based on statistical indices and aim at measuring the degree to which a data set confirms a previously specified structure. Although a high computational cost is required to compute the statistical indices, the produced results are suitable for quantitative evaluation of a clustering result. In this sense, external and internal criteria are used to evaluate the peer clustering process proposed in this thesis. In the following chapter, we describe Peer Data Management Systems (PDMS) and emphasize the importance of clustering peers according to their semantics.

CHAPTER 3

PEER DATA MANAGEMENT SYSTEMS

"No verão, um balde de água faz uma colher de lama; no outono, uma colher de água faz um balde de lama"
Provérbio ucraniano

The increasing use of computers and the development of communication infrastructures have led to a wide range of data sources being available through networks such as *Peer Data Management Systems* (PDMS) [Halevy *et al.*, 2003a; Halevy *et al.*, 2003b]. This setting is characterized by having a diversity of perspectives, dynamic data, and the possibility of intermittent participation. It is generally composed of a set of autonomous and heterogeneous data sources (i.e. peers) which are linked by means of mappings or correspondences.

In this chapter, we characterize PDMS and discuss their main data management issues: *schema mappings* and *query processing*. Besides, we discuss the importance of organizing peers in an overlay network based on their semantics and discuss the main challenges to achieve this goal. Afterwards, we describe some existing PDMS that propose a semantic-based approach to organize peers in the overlay network. Finally, we present a comparative analysis of such PDMS.

3.1 PDMS Definition

A Peer Data Management System (PDMS) [Ng *et al.*, 2003; Tatarinov *et al.*, 2003; Ruzzi, 2004; Valduriez and Pacitti, 2004; Halevy *et al.*, 2006; Pires *et* *al.*, 2006; Mandreoli *et al.*, 2007; Lodi *et al.*, 2008] is an advanced P2P application which enables users to transparently query several distributed, heterogeneous, and autonomous data sources. Differently from the traditional P2P file sharing systems, PDMS deal with more semantically rich structures (e.g. XML documents, relational tables, and ontology concepts). Therefore, in such systems peers use a more complex specification of their contents, e.g. database schemas [Bernstein *et al.*, 2002] and formal ontologies [Rousset *et al.*, 2006], than the classical P2P file sharing systems [Leibowitz *et al.*, 2003].

PDMS are considered the result of blending the benefits of P2P systems, such as lack of a centralized authority, with the richer semantics of databases. In general, PDMS are rather useful when sources have overlapping structured or semi-structured data and users want to access additional related information stored in other peers. PDMS may be used for data exchanging, query answering, and information sharing in almost every application domain, for example scientific research and educational systems.

Due to the intrinsic characteristics of the P2P computing paradigm [Walkerdine *et al.*, 2002], the assumption that all participating peers rely on a single global schema, such as in traditional data integration systems [Wiederhold, 1992], cannot be made [Aberer *et al.*, 2002]. Otherwise, the global schema may need to be updated every time the system evolves [Giunchiglia and Zaihrayeu, 2002; Young, 2004]. Instead, each peer represents an autonomous data source and provides an *exported schema* (or *peer schema*) to be shared with the other peers of the system.

Among those exported schemas, *schema mappings*, i.e. correspondences between schema elements, are generated. Schema mappings are needed to establish (meaningful) information exchange between peers. The peers with which a particular peer *P* maintains schema mappings are commonly called the *neighbors* of *P*. Consequently, these peers form the *neighborhood* of *P*. Query processing in a PDMS is accomplished by traversing the schema mappings, reformulating the queries, executing them on the peers, and gathering the results at the peer that requested the data.

Sharing data in a PDMS is made easier, because each peer can map its schema to the most convenient peers in the system rather than to a mediated schema. In addition, peers can query the system using their own schema rather than using a mediated schema that may be foreign to them.

3.2 Data Management Issues in PDMS

The most studied data management issues in PDMS are probably related to *schema mappings* [Madhavan and Halevy, 2003; Tatarinov *et al.*, 2003; Hai, 2005] and *query processing* [Katchaounov, 2003; Ng *et al.*, 2003; Castano *et al.*, 2004; Tatarinov and Halevy, 2004; Mandreoli *et al.*, 2006a; Hose *et al.*, 2007; Souza, 2007; Montanelli and Castano, 2008; Kantere *et al.*, 2009]. In this section, we offer an overview of such issues and describe some of the techniques commonly applied by well-known PDMS.

3.2.1 Schema Mappings

As illustrated in Figure 3.1, two types of mappings are commonly used in PDMS [Sung *et al.*, 2005]: *local mappings* and *schema mappings*. Since data sources can be heterogeneous in a PDMS, a common data model is needed to describe exported schemas using a uniform conceptual notation. Thus, local mappings are used to define *correspondences* between the elements available in the data source and a peer's exported schema. Moreover, since exported schemas can use different names or formalisms to represent the same data, schema mappings are needed to define (*semantic*) *correspondences* between the elements (e.g. relations and attributes) in two exported schemas for the purpose of sharing and integrating data [Madhavan and Halevy, 2003]. As with traditional distributed databases, the reason behind integrating data using schema mappings in PDMS is to provide a uniform querying environment that hides the heterogeneity and distribution of the data sources.

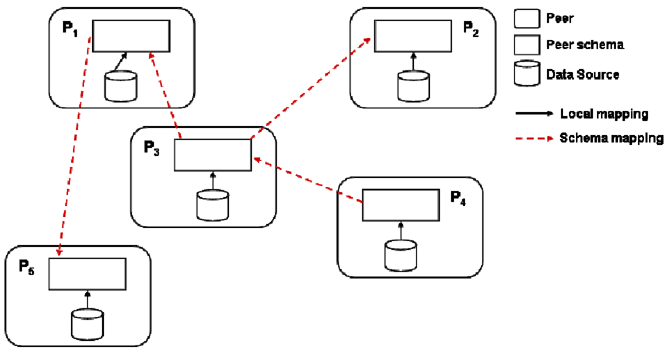


Figure 3.1. The different types of mappings in a PDMS.

Most P2P systems assume the existence of schema mappings among peers, but do not specify how these mappings are determined. However, the definition of the schema mappings is probably one of the hardest aspects of the data sharing process. It is considered an expensive process that in some cases requires human intervention [Rousse and Berman, 2006]. In traditional data integration systems, schema mapping is typically specified manually and at design time. Differently, in PDMS peers need to coordinate their data sources on-the-fly, therefore ultimately requiring run time schema mapping. [Rahm and Bernstein, 2001] propose an approach for determining schema mappings automatically.

Basically, schema mappings can be categorized in one of three groups: *pair mappings*, *peer-mediated mappings*, and *super-peer mediated mappings* [Sung *et al.*, 2005]. Pair mappings are the simplest approach to define schema mappings in PDMS. As illustrated in Figure 3.1, schema mappings are defined between pairs of peers. The mappings are stored at the peer interested in accessing the other peer's data. For example, the Local Relational Model (LRM) follows this approach to schema integration [Bernstein *et al.*, 2002]. In the LRM proposal, a peer specifies translation rules and coordination formulas that define how its exported schema relates to an exported schema on another peer. Such agreement between two peers is referred to as an *acquaintance* [Kantere *et al.*, 2003]. A semantic network is then defined by the acquaintance links between peers.

In peer-mediated mappings, a peer can define a schema mapping that relates two or more peers. Hence, these mappings are a generalization of pair mappings. Piazza [Tatarinov *et al.*, 2003] and PeerDB [Ng *et al.*, 2003] are systems that follow this approach for integration. Super-peer mediated mappings can be defined when peers are organized according to a super-peer topology [Yang and Garcia-Molina, 2003]. In such systems, mediated schemas are defined at the super-peer level. *Super-peer to peer* mediated schemas contain the mappings for all the regular peers associated to the super-peer. Schemas between super-peers, called *super-peer to super-peer* mediated schemas, are also defined to implement data sharing between peers associated to different super-peers.

3.2.2 Query Processing

In PDMS, where frequent changes in schemas or in source availability are common, decentralized techniques for query processing need to be applied. The suitability of these techniques has been demonstrated by the success of early P2P file sharing systems. In such systems, queries are routed to peers with the goal of locating files. However, in systems sharing structured data such as PDMS, queries are more expressive. Therefore, query processing should follow more traditional optimization, plan generation, and execution stages.

Query processing in a PDMS can be described as illustrated in Figure 3.2. Suppose that a query Q is formulated at an initial peer (P_4) according to its respective exported schema. The first step consists in identifying the *relevant peers* (i.e. the peers that can answer Q and, consequently, the peers to which Q should be sent) among the neighbors of P_4. In order to determine the relevant peers, several techniques have been proposed, e.g. semantic routing indexes [Mandreoli *et al.*, 2006b; Mandreoli *et al.*, 2007], expertise tables [Faye *et al.*, 2007], and routing tables [Li and Vuong, 2007]. In general, such techniques are based on query type and query specifications such as required attributes, predicates, and data filters.

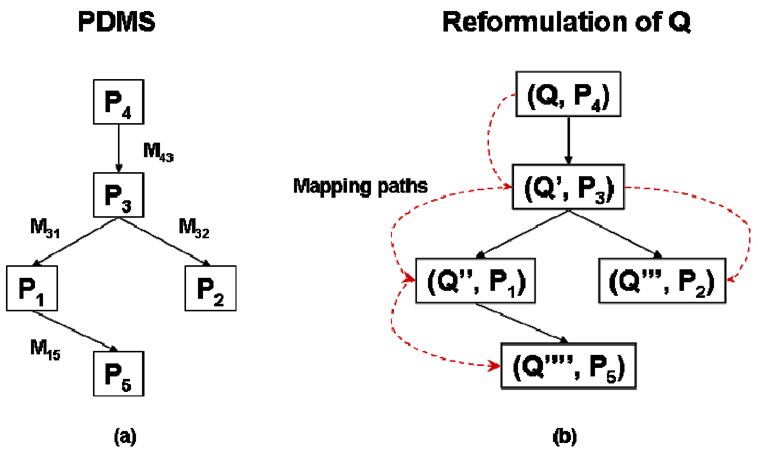


Figure 3.2. Typical query processing in a PDMS.

In the example, assume that P_3 is a relevant peer. Once the relevant peers are identified, Q is reformulated according to the corresponding schema mappings and distributed among relevant peers. At each relevant peer the same process is repeated so that Q can reach other peers in the network (in the example, P_1, P_2, and P_5). Due to scalability reasons, query distribution can be possibly restricted to a certain limit, e.g. *time-to-live* (TTL). Processing and filtering of results is done incrementally by visited peers as Q is distributed. The results are sent from the visited peers to P_4 where the combination of results is normally done.

In this scenario, incomplete and approximate answers are acceptable, as long as they are good enough for a PDMS application. This is because some schema mappings involved in query answering may become temporarily unavailable or invalid. Moreover, a query may have to be reformulated several times from peer to peer until it reaches peers that are able to answer it. Successive query reformulation produces query versions that can deviate from the original query. Obviously, if the chain of schema mappings used for the reformulating is "poor" in information relevance to the query (i.e. query parts cannot be reformulated accurately), this can result in fast degradation within a few hops [Kantere *et al.*, 2009].

In the PeerDB system [Ng *et al.*, 2003], queries are executed in two steps: in the first step, peers are selected based on the amount of metadata intersection between query terms and peer schemas; in the second step, queries are submitted to the selected peers and results are sent back to the query initiator. In PeerDB, users are involved in the selection of potentially promising peers from the set of peers obtained in the first step of query execution.

Concerning query reformulation, in the Local Relational Model [Bernstein *et al.*, 2002], a Query Manager component located at each peer, uses data translation rules and semantic dependencies to reformulate queries submitted at a peer to match the schemas of other peers. In the Piazza system [Halevy *et al.*, 2003a], storage mappings are used to associate queries with suitable data relations, while description mappings are used to associate query results at one peer to results at other peers. Based on these description mappings, a reformulation algorithm is capable of producing query expressions which are equivalent to a given query. In [Souza *et al.*, 2009], the authors propose a semantic-based approach for reformulating queries in a PDMS in which queries are semantically enriched with additional elements obtained from a domain ontology.

3.3 PDMS and Ontologies

Since peers in a PDMS can store heterogeneous data sources, a common metadata model is needed to describe the exported schemas using a uniform conceptual notation. The use of a common model facilitates the definition of schema mappings and, consequently, improves query processing. Some examples of commonly used metadata models include relational schemas, XML schemas, RDF schemas, and OWL ontologies. Among such metadata models, ontologies turn out to be the most elaborate form [Euzenat and Shvaiko, 2007]. The distinctive feature of ontologies is that their interpretation is not left to the knowledge management systems implementing them, but is specified explicitly. Due to its rich expressiveness, ontologies have been considered as a basis for making explicit the content of data sources and, consequently, as a means for promoting interoperability in PDMS [Nejdl *et al.*, 2002; Castano *et al.*, 2003; Xiao, 2006; Li and Vuong, 2007].

Particularly, [Xiao, 2006] has introduced a new definition for the blending of PDMS and Ontologies' researches. In his work, such blending has lead to the emergence of *Ontology-based Peer Data Management Systems* (OPDMS). The main goal of such systems is to provide semantic interoperability between peers. According to [Xiao, 2006], in an OPDMS ontologies are used not only as a uniform conceptual notation to represent exported schemas, but also to describe schema mappings between peers. Particularly, in OPDMS the terms *ontology mappings* or *semantic mappings* are used interchangeably to refer to schema mappings [Wicaksana and Yétongnon, 2006].

When exported schemas are represented by ontologies, the identification of ontology mappings can be assisted by an ontology matching process. Once the correspondences between ontology elements are identified, they can be used for the purpose of query answering. In a PDMS, the ontologies representing exported schemas are designed and developed independently. Thus, even if ontologies are used as a uniform conceptual notation, users still may follow diverse modeling principles and patterns to encode the same real-world object. Moreover, since peers are meant to be totally autonomous, users may use different terminologies in order to represent their data, even if they refer to the same domain of interest. Ontology matching is somewhat similar to database schema matching, but it has many particularities due to the structural and conceptual differences between ontologies and database schemas [Uschold and Gruninger, 2004]. Ideally, the ontology matching processes used in PDMS should deal with the heterogeneity problems discussed in Section 2.2.

3.4 Semantic-based Peer Grouping in PDMS

One of the first solutions for semantic-based peer grouping was proposed for P2P file sharing systems [Yang and Garcia-Molina, 2003]. In this work, peer grouping is treated as a supervised process which is guided by a predetermined semantic classification. The files shared by peers are classified into categories and peers are allocated into *Semantic Overlay Networks* (SONs) according to such classification. Clearly, this strategy can be applied when the data shared by peers do not differ in terms of structure and vocabulary. However, in PDMS, as each peer represents an individual organization, different peers can adopt distinct exported schemas. Therefore, peer grouping should be considered an unsupervised process and semantic communities should be formed as a result of a common agreement among peers [Castano and Montanelli, 2005; Kantere *et al.*, 2008; Lodi *et al.*, 2008].

Several solutions have already been proposed for the dynamic formation and maintenance of semantic communities of peers in PDMS. Some solutions [Ramaswamy *et al.*, 2003; Castano and Montanelli, 2005; Doulkeridis *et al.*, 2006; Kantere *et al.*, 2008] assume that the network is already populated with a predetermined number of peers and then the participating peers start the formation of semantic communities in an ad-hoc manner. In such systems, semantic communities are autonomously emerging [Castano and Montanelli, 2005]. Communities originate from a declaration of interest of a peer and group those peers which spontaneously agree with the declaration, since they can offer relevant resources for the community.

Only a few solutions [Li and Vuong, 2005; Lodi *et al.*, 2008] consider the problem of forming semantic communities from scratch. In such solutions, the network is initially empty and new peers are added to related semantic communities as they join the network. If an interesting semantic community is not found then a new one is formed. Moreover, some systems [Kantere *et al.*, 2008; Lodi *et al.*, 2008] enable peer participation in multiple communities while others are more restrictive in the sense that each peer can take part in only one community [Li and Vuong, 2007].

Several distinct semantic communities can coexist in a PDMS. Within each community peers can be organized according to some existing P2P network topology, e.g. unstructured [Leibowitz *et al.*, 2003], structured [Stoica *et al.*, 2001; Ratnasamy *et al.*, 2001], or super-peer [Yang and Garcia-Molina, 2003]. Particularly, concerning the super-peer topology peers also need to be organized in a more specific level within the community, i.e. the clusters. Thus, an approach to insert peers into appropriate clusters is essential.

The task of dynamically forming and maintaining semantic communities of peers in a PDMS imposes several important challenges. Next, we enumerate some of these challenges and succinctly define each one.

Similarity (or distance) measurement

A semantic community should contain only semantically similar peers. In order to determine if two peers are semantically similar (or dissimilar) it is necessary to measure their semantic similarity (or distance) [David and Euzenat, 2008]. In this sense, peer exported schemas are commonly used to calculate the similarity (or distance) measure between peers.

Neighborhood search

All peers in a semantic community share content associated to the same knowledge domain. However, a peer can be more semantically related to some particular peers in a community than to others. Therefore, even after a new peer joins a semantic community it is still necessary to determine the peer's neighborhood inside the community [Lodi *et al.*, 2008]. In this sense, an efficient strategy for routing peers to other related peers should be available. Such strategy can be associated to the type of network topology (e.g. unstructured, structured, or super-peer) adopted to organize peers within the community.

Neighborhood selection

During the definition of a peer's neighborhood several possible candidate peers can be identified in a semantic community. Therefore, the new peer needs to choose a subset of the identified peers in order to determine its neighborhood.

Range-based [Lodi *et al.*, 2008] and *threshold-based* [Castano and Montanelli, 2005] techniques can be used to select the most similar peers. However, such techniques are not very restrictive in the sense that a high number of peers can be chosen to form a peer's neighborhood. Since the physical capacity (e.g. network bandwidth) of a peer is limited it could be necessary to limit the number of neighbors in order to avoid the consuming of all peer resources [Ramanathan *et al.*, 2002; Zhuang *at al.*, 2004].

Neighborhood maintenance

Due to the dynamicity of peer participation in a PDMS it is necessary to adjust the neighborhood of each peer as new peers join the system or participating peers leave the system. For instance, such adjustment can be done periodically (e.g. network stabilization) [Xu and Srimani, 2005] or by monitoring peer's answers during query processing [Ramanathan *et al.*, 2002; Ng *et al.*, 2003].

3.5 Existing Semantic-based PDMS

Several PDMS have been proposed in the literature: *Piazza* [Halevy *et al.*, 2003a; Halevy *et al.*, 2003b; Tatarinov *et al.*, 2003], *PeerDB* [Ng *et al.*, 2003; Ooi *et al.*, 2003], *Hyperion* [Arenas *et al.*, 2003], *APPA* [Valduriez and Pacitti, 2004], and *Xpeer* [Bellahsène and Roantree, 2004]. In general, such initial systems propose solutions to problems associated with schema mappings and query processing.

In the last few years, research on PDMS has focused on semantic-based techniques to overcome the main limitations of initial PDMS, especially those limitations related to query processing. In this light, several PDMS have been proposed employing some kind of semantic-based approach to organize peers in the network: *Sunrise* [Lodi *et al.*, 2008], *Helios* [Castano *et al.*, 2004], and *OntSum* [Li and Vuong, 2007]. In this section, we offer a more detailed description of such PDMS focusing on the way that semantic communities are formed and maintained. These PDMS were chosen because they propose solutions to the challenges related to semantic community management discussed in Section 3.4. At the end of the section, we also present a comparative analysis of such PDMS.

3.5.1 Sunrise (University of Bologna)

[Lodi *et al.*, 2008] propose a solution for the creation and maintenance of a flexible network organization for PDMS that clusters together heterogeneous peers which are semantically related. Each peer in the network is represented by a set of concepts describing its main topics of interest. The representation of each peer derives from the peer's exported schema as it describes the semantic content of the underlying data. The network is organized in a set of Semantic Overlay Networks (SONs) [Crespo and Garcia-Molina, 2002] in such a way to assist each new peer in the selection of the semantically closest peers as its neighbors. A SON is a group of semantically related peers locally connected through a link structure. Peers are assigned to one or more SONs on the basis of their exported set of concepts. A sample of network made up by two SONs supporting a web of research-related data is shown in Figure 3.3. The network includes various peers. Some of them, such as the *EDBT Association (EDBT Ass.)* and the *University of Rome (URome)* are monothematic, i.e. they only deal with publications and university people, respectively. Other peers, instead, are concerned with both themes, e.g. *Stanford*.

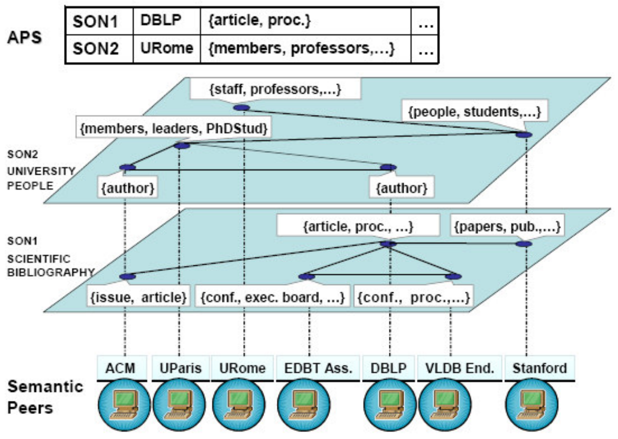


Figure 3.3. Sample of network organization [Lodi *et al.*, 2008].

Similarity between peers is captured by a distance function *d* considering their exported sets of concepts. The function quantifies the distance between two given concepts by comparing their WordNet [Miller, 2005] hypernymy hierarchies. Two variants of linguistic distances are considered. The first distance is obtained by computing the depths of the concepts in the WordNet hypernymy hierarchy and the length of the path connecting them. The other distance considers the number of links connecting the two given concepts in the hypernymy hierarchy as well as the height of the hierarchy (16 in WordNet).

The approach for organizing peers is inspired on incremental clustering algorithms. The network evolves incrementally to assimilate new peers. When a peer joins the system, it first performs a coarse-grained neighbor selection by accessing the *Access Point Structure (APS)*. APS is a centralized structure which maintains a summarized representation of each SON available in the network. The APS helps new peers to decide which SONs to join or whether to form new SONs by providing useful information such as the most representative concepts of each SON. Such representative concepts are compared to the new peer's concepts.

When a new peer has chosen its semantically closest SONs, it navigates the link structure within each selected SON with the aim of searching for its preferred neighbors, i.e. the semantically nearest peers. In particular, two types of neighbor selection are supported: each peer is allowed to select either the *k* semantically nearest peers (*k-NN selection*) or the peers in the SON for which the distance between their SON's concepts and the peer SON's concepts are below a given threshold (*range-based selection*). The topology of the network is strongly influenced by the type of neighbor selection. A k-NN selection limits the number of neighbors and thus controls the degree of connectivity. This is not possible in a range-based selection where it is only possible to provide an estimation of the number of neighbors based on the SON statistics maintained at the APS level.

Adopting a broadcast-based approach to search neighbors could imply wasting precious resources. Therefore, the authors propose that the neighbor selection process to be guided by a distributed index mechanism which maintains at each node specifically devised indices named *Semantic Clustering Indices* (SCIs) [Mandreoli *et al.*, 2006a; Mandreoli *et al.*, 2006b]. The collection of SCIs distributed across the peers that the new peer visits, drives its navigation towards the peers in the same SON containing the concepts nearest to its concepts. The Sunrise semantic framework is presented in [Mandreoli *et al.*, 2007].

3.5.2 Helios (Università degli Studi di Milano)

Helios [Castano *et al.*, 2003] is a system for ontology-based knowledge discovery and sharing in peer-based open distributed systems. It addresses the problem of forming semantic communities of peers in a P2P system. Each peer provides an ontology representing the resources it exposes to the network. Ontologies are represented in the H-Model [Castano *et al.*, 2004], a language independent ontology model capable of describing a number of ontology specification formalisms (e.g. OWL, RDF(S), and UML). Each peer implements a *Semantic Matchmaker* component for matching ontologies in order to find which concepts match in different ontologies and at which level. The Semantic Matchmaker is based on the H-Match algorithm [Castano *et al.*, 2004] and performs dynamic ontology matching by taking into account both linguistic and contextual features of the concepts. H-Match performs ontology matching at different levels of depth, implementing four different matching models: *surface*, *shallow*, *deep, and intensive* [Castano et al., 2006].

In Helios, a semantic community of peers is identified by a unique *Community Identifier* (CID), and a subject category or topic area of interest called *community Identity Card* (ICard), defined as an ontology [Castano and Montanelli, 2005]. The semantic community formation process is addressed under certain assumptions: (i) each peer can be member of multiple communities; (ii) no centralized authority (e.g. super-peer) is expected to coordinate the community discovery and formation process; and (iii) the choice of joining an emergent community depends on the matching results. A semantic community of peers emerges when a peer, called *community founder*, invokes a semantic handshake process which is composed of the following tasks:

ICard advertisement. The founder P_f defines a CID and an ICard describing the topic area of interest of the emerging community, along with a set of commitment constraints specifying the conditions required for the community establishment (e.g. minimum number of members). Then, P_f composes an *Invitation Message* containing the CID and the ICard created, as well as a TTL parameter defining the maximum number of hops allowed for the invitation propagation, the matching model to be used for affinity evaluation, and the matching threshold *t* specifying the minimum semantic affinity value required to consider a concept of the ICard and a concept of a peer ontology as matching concepts. Then, the invitation message is sent to all P_f neighbors in order to advertise the new community.

Member identification. Each invited peer P_i invokes the Semantic Matchmaker in order to compare the incoming ICard with its peer ontology. P_i is relevant for the community if the Semantic Matchmaker identifies concepts in the peer ontology with a semantic affinity higher than the specified threshold *t* with respect to the ICard. In this case P_i replies to P_f with an *Interest Message* reporting the portion of its peer ontology related to the matching concepts found to be relevant for the community by the Semantic Matchmaker. Independently from the matching results and if TTL ≥ 0, P_i forwards the invitation message to all its neighbors.

Request approval. Once the interest messages are received, P_f has to evaluate which peers are admitted in the community. For this reason, P_f invokes its Semantic Matchmaker and compares each peer ontology portion received by the interested peers with its peer ontology. For each candidate peer, the goal of this comparison is to evaluate whether the provided knowledge matches the knowledge of P_f, and then to assess whether they share a common perspective of the community interests. If the matching results are higher than *t*, P_f admits the peer in the community and sends an *Approval Message* to the admitted peer.

Community commitment. Once the *Request approval phase* is completed, P_f verifies if the commitment constraints are satisfied. In this case, a *Commitment Message* is sent to all the admitted peers and the semantic community is effectively established. If the committed constraints are not satisfied, P_f stops the community formation. In this case, the admitted peers wait for the commitment message until a predefined timeout expires.

Figure 3.4 illustrates an example where the handshake algorithm is applied to a P2P network and the community founder (P_f), represented by a double hoop, sets an initial TTL = 2. Dashed lines represent random P2P connections and the path followed by the invitation message (continuous line) defines a tree structure where the root is identified by P_f and the leafs are represented by the invited peer with TTL = 0. Each invited peer negotiates its participation in the community directly with P_f. Once it is admitted, the peer exploits the tree structure and communicates within the community through its community neighbors. The authors define the community neighbors of a community member P_m as the peer that invited P_m in the community (i.e. P_m predecessor) and the peers that P_m invited in the community (i.e. P_m successors). An invited peer not interested in the community or discarded by P_f is to be pruned from the tree structure of the community. For this reason, after the approval phase, each community member P_m notifies to its predecessor P_p of its presence in the community. If P_p is not member of the community, it forwards the P_m notification to its predecessor P_g and notifies P_m that P_g is its new predecessor. As an example, consider peer E in Figure 3.4. The community members peer H and peer K notify peer E of their participation. Peer E has not joined the community and is to be pruned from the community tree. Then, peer E forwards the notification to peer B and notifies peer H and peer K that peer B is the new predecessor of peers H and K.

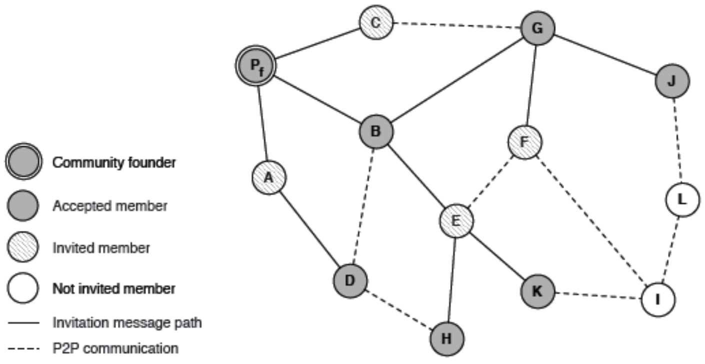


Figure 3.4. Example of aggregation of a semantic community [Castano and Montanelli, 2005].

3.5.3 OntSum (University of British Columbia)

OntSum [Li and Vuong, 2007] is a PDMS which uses an ontology-based approach to address the routing issues of expressive queries. Peers use ontologies to describe their shared content. A metric to measure peers' ontology similarity is used to organize peers according to their semantic properties. The network topology is reconfigured with respect to peer's ontological similarity, so that peers with similar ontologies are close to each other. As proposed in Description Logics [Baader *et al.*, 2003], the system divides a peer's ontology into two parts: (i) *taxonomical box (T-Box)*: stores conceptual knowledge; and (ii) *assertion box (A-Box)*: represents the concrete knowledge about individuals. A peer's T-Box concepts are indexed into a vector (*ontological signature* *vector*). The semantic similarity between two peers A and B is defined as the cardinality obtained from the intersection between *V(A)* and *V(B)* divided by the cardinality of *V(A)*, where *V(A)* and *V(B)* are the ontology signature vectors of peers A and B. Two peers are semantically similar if their similarity is beyond a *similarity threshold*.

Figure 3.5 illustrates a high level view of the network topology. Peers form multi-layered clusters reflecting the semantic locality: peers with similar ontological topics form a big domain; inside the domain, peers may create smaller clusters if they share the same ontology. For instance, all peers in the *medical* domain are interested in medically related information. They may be interested in different aspects of the medical resources, and they may use different ontologies to describe their contents, but since they share the similar interests (*medicine*), they connect with each other through some links. Peers N_1, N_2, N_5, and N_8 use the same ontology: *ont₁*, so they form a same-ontology cluster. The term "domain" is used to represent a group of clusters sharing similar ontological topics while the term "cluster" is used to denote the same-ontology cluster. Clusters and domains do not have fixed boundaries; they are formed by randomly connecting relevant peers.

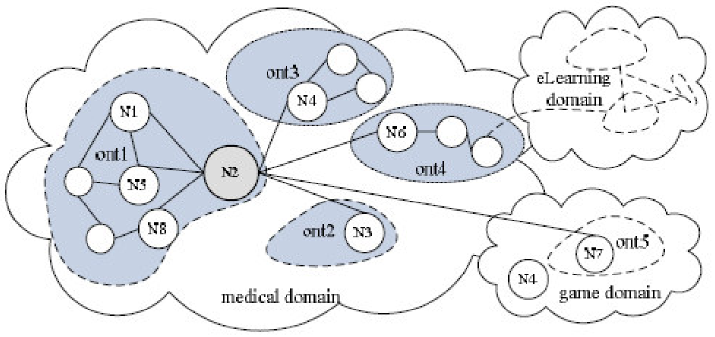


Figure 3.5. The network topology [Li and Vuong, 2007].

To form this multi-level structured network a peer distinguishes three kinds of neighbors based on their semantic similarity. Two peers A and B can be neighbors at different levels: (i) *zero-distance neighbor* (or same-ontology neighbor, intra-cluster neighbor), if sim(A,B) = 1; (ii) *short-distance neighbor* (or semantically related neighbor) if sim(A,B) $\geq$ t ($0 < t < 1$ is A's semantic threshold); and (iii) *Long-distance neighbor* (or semantically unrelated neighbor) if sim(A,B) < t. A peer always tries to find as many close neighbors as possible, but it also keeps some long distance neighbors to reach out to other ontological clusters.

Peers in the system may pose two types of queries: *neighbor-discovery query* and *resource-discovery query*. The neighbor-discovery query is used to construct the ontology-based network topology. When a new peer joins the network, it issues neighbor-discovery query to find semantically related neighbors, so that it can join their domain and cluster by connecting to them. The resource-discovery query is used to locate desirable resources in the network. To efficiently route queries, two routing schemes are proposed: *inter-cluster routing* and *intra-cluster routing*. The former quickly locates semantically related clusters; while the latter efficiently finds desirable resources satisfying the query constraints. Related with the two routing schemes, two routing tables are maintained at each node: (i) *inter-cluster routing table*: stores the abstract semantic knowledge of its neighboring clusters (short-distance and long-distance neighbors, their semantic similarities to the peer, and their semantic signature vector); and (ii) *intra-cluster routing table*: used to forward queries inside a cluster.

A peer joins the network by connecting to one or more bootstrapping neighbors. Then the joining peer issues a *neighbor-discovery query*, and forwards the query to the network through its bootstrapping neighbors. The neighbor-discovery query is routed mainly according to the inter-cluster routing table. A neighbor-discovery query message includes several parts: (i) the querying peer's ontology signature vector; (ii) a similarity threshold which is a criteria to determine if a peer is semantically related to the query; (iii) a query TTL to decide how far the query should be propagated; and (iv) a list of clusters the query has passed through, so that the query would not be forwarded to the same cluster. When a peer N receives a neighbor-discovery query Q which tries to find neighbors for a new joining peer X, N computes the semantic similarity between X and itself; if N is semantically related to X, N will send a Neighbor Found reply. If the query's TTL does not expire, N computes the semantic similarity between X and each of its neighbors, and forwards the query to semantically similar neighbors. If no semantically similar neighbors are found, the query will be forwarded to N's long-distance neighbors.

3.5.4 Comparative Analysis

Table 3.1 presents a comparative analysis of the previously discussed PDMS. All of them employ a semantic-based approach to organize peers in the overlay network. The comparative criteria were chosen according to the main interest of our work. They indicate how the PDMS deal with the challenges related to forming and maintaining semantic communities discussed in Section 3.4. The criteria are described as follows.

- *Schema representation*: refers to the metadata model that is used to represent peer exported schemas;
- *Network topology*: indicates the P2P topology that is used to organize peers in the network;
- *Network population*: indicates the network population status at the moment communities start being formed;
- *Domains*: indicates if semantic communities are originated spontaneously from the peers or if predefined semantic communities are initially available;
- *Multiple communities*: indicates if a peer can participate in multiple communities (or clusters);
- *Neighborhood search*: refers to the strategy that is used by new peers to find other semantically similar peers in a community;
- *Semantic similarity measure*: corresponds to the measure used to calculate the (semantic) similarity between peers;
- *Neighborhood selection*: refers to the method used to determine the semantic neighborhood of a peer in a community (or cluster).

Table 3.1. A comparison of PDMS employing a semantic-based approach to organize peers in the network.

PDMS	Schema represent.	Network topology	Network population	Domains	Multiple communities	Neighborhood search	Semantic similarity measure	Neighborhood selection
OntSum	Ontologies	Unstructured	Not empty	Predefined semantic domains	No	Flooding; first short-distance links, then long-distance links; inter-cluster table	Semantic similarity measure (ontology matching)	Threshold
Sunrise	Generic (Ontologies, Relational, XML)	Unstructured	Empty	Non existing domains	Yes	Centralized Access Point Structure (APS) followed by SCI	Semantic distance between concepts (clustroids)	Range-based and kNN-based algorithms
Helios	Ontologies	Unstructured	Not empty	Non existing domains	Yes	Flooding	Ontology matching	Threshold

Concerning OntSum, the authors do not mention anything about the ontology matching algorithm used in the system. A very simple and asymmetric global measure is used to compute the semantic similarity between two peers' ontologies. Moreover, it is hard to accept that peers in a PDMS share exactly the same ontology, especially when the authors assume that peers' ontologies can differ in terms of structure and vocabulary. In this work, we propose a symmetric global measure to determine the similarity between peer schemas (ontologies). The measure is obtained as a result of an ontology matching process which uses linguistic, structural, and semantics matchers.

Sunrise concentrates all efforts related to peer grouping in a centralized structure called *Access Point Structure* (*APS*). The APS is an index structure in which each entry contains the most representative concepts of a SON. Such concepts are obtained from the peers that participate in the SON. In this sense, the APS maintains a summarized representation of each SON available in the network. Each time a new peer joins a SON, if it has at least one concept that is not listed in the most representative concepts of the SON, then the APS needs to be updated. The same occurs if a participating peer leaves the system. Thus, the frequency of updates in the APS can be intense and consequently bring scalability problems to the system.

A distinguishing feature of Helios with respect to the other PDMS is that semantic communities are formed in an ad-hoc manner after peers are connected to the system. Such approach enables the formation of dynamic communities since no classification or set of semantic domains needs to be available a priori. However, since the initial neighborhood of peers is defined randomly then unrelated peers may become neighbors in the network. Therefore, during the process of community formation many unrelated peers can be accessed and unnecessary ontology matching comparisons may be executed. The complete absence of any kind of centralized control does not enable a *community founder* to verify the existing communities before starting the formation of a new one. Thus, it is possible to coexist multiple communities dealing with the same topic. To avoid these problems, we consider the use of predefined semantic communities. Each community is initially empty and new peers are added to the community as they join the system.

3.6 Considerations

In this chapter, we have described Peer Data Management Systems (PDMS) and two of its data management issues: schema mappings and query processing. We have shown the importance of organizing peers in a PDMS overlay network according to their semantics and the main challenges to achieve this goal. A description of existing PDMS and a comparative analysis of them were also provided. According to such comparison, we conclude that (i) none of the discussed systems take advantage of the benefits provided by the mixed use of P2P network topologies to facilitate the formation of semantic communities; and (ii) ontologies are mostly used to represent peer exported schemas. In the next chapter, we propose a PDMS in which ontologies are used in a broader way to improve some of its main services. The overlay network of the proposed PDMS is mainly designed to assist the connection of peers according to their exported schema (i.e. an ontology).

CHAPTER 4

SPEED: A SEMANTIC-BASED PEER DATA MANAGEMENT SYSTEM

"The problems that exist in the world today cannot be solved by the level of thinking that created them"
Albert Einstein

In this chapter we discuss how ontologies can be employed in a PDMS to improve its main services. We describe an Ontology-based PDMS (OPDMS) whose overlay network is mainly designed to assist the organization of peers according to their exported schema (i.e. an ontology). Such description includes an overview of the system's architecture, the distinct types of peers as well as their internal modules, and the different types of ontologies used in the system. We also describe how schema mappings and query processing are handled in the proposed OPDMS.

4.1 Ontology-based PDMS

An ontology typically provides a vocabulary describing a domain of interest and a specification of the meaning of terms in that vocabulary. Ontologies are viewed as the "silver bullet" for many applications [Fensel, 2004] such as, for example, database integration, P2P systems, e-commerce, semantic web services, and social networks.

Xiao [Xiao, 2006] has introduced the concept of OPDMS through two important issues: (i) ontologies are used in local sources as a uniform conceptual metadata representation; and (ii) ontology mappings are established between peers' ontologies to allow query processing. In this thesis, we argue that ontologies may be used in a broader way to enhance PDMS services. Considering that, we propose an extension to the original OPDMS description. We define an OPDMS as *a PDMS which is conceived for supporting dynamic ontology-based knowledge sharing, and this knowledge must be employed to improve its services* [Pires *et al.*, 2008]. Moreover, based on our analysis of the state-of-the-art on PDMS presented in Chapter 3, we have identified a set of high-level requirements that an OPDMS should fulfill:

R_1) *Exported schema representation:* peer's metadata should be mapped onto an ontology description, using a common model;

R_2) *Global conceptualization*: an ontology should represent a high-level view over a set of heterogeneous peer exported schemas;

R_3) *Support for correspondences identification*: an ontology may also be used to assist the identification of correspondences between peer exported schemas, i.e. between ontologies;

R_4) *Support for query processing*: query answering in a PDMS may use a global ontology in a twofold way: *a*) as a high-level view of the sources; and *b*) as a terms' reference for query reformulation between peers. The former is concerned with query formulation, i.e. the user can formulate a query using the global ontology without specific knowledge of the different data sources stored in the peers. The latter is concerned with query reformulation, i.e. the query is reformulated into a target query over other connected peers, according to the defined correspondences among them;

R_5) *Semantic index*: a semantic index can be built according to the main terms or categories referring to a set of ontologies. Such index must enable efficient location of peers;

R_6) *Semantic matching capabilities*: a semantic matching component is needed for matching ontologies in order to find out which concepts match in different ontologies and (possibly) at which level. Such capability can be used for the organization of peers in the network and the definition of semantic correspondences between peers.

A system should take into account the previous requirements not only to be considered an OPDMS, but also to take full advantage of using ontologies for semantic enrichment. Table 4.1 illustrates the requirements fulfilled by the three PDMS discussed in Chapter 3 (OntSum, Sunrise, and Helios). None of them satisfies all the identified requirements. In this sense, we propose a new OPDMS architecture satisfying all the requirements. Particularly, how the requirements R_1, R_2, R_3, R_5, and R_6 are satisfied by the OPDMS are discussed throughout this work, while R_4 is treated in more details in [Souza, 2009]. In the following sections, we describe the proposed OPDMS.

Table 4.1. High-level requirements fulfilled by PDMS.

PDMS	R_1	R_2	R_3	R_4	R_5	R_6
OntSum	👍	👎	👍	👎	👎	👍
Sunrise	👍	👎	👍	👎	👍	👎
Helios	👍	👎	👍	👎	👎	👍
SPEED	👍	👍	👍	👍	👍	👍

4.2 System Architecture

In this section, we propose an OPDMS, named *SPEED* (**S**emantic **PEE**r-to-Peer **D**ata Management System) [Pires, 2007a]. The system utilizes a mixed P2P network topology: DHT [Stoica *et al.*, 2001], super-peer [Yang and Garcia-Molina, 2003], and unstructured [Freenet, 2009]. The strengths of such topologies are exploited in order to assist peer organization in the network according to their exported schemas. SPEED's main goal is to cluster semantically similar peers in order to facilitate the establishment of semantic mappings between peers and, consequently, improve query processing on a large number of data sources [Pires *et al.*, 2009c]. Next, we present an overview of the SPEED's architecture.

4.2.1 Architecture Overview

A DHT network is used to link particular peers that represent a certain knowledge domain. Peers are grouped according to their knowledge domain (e.g. *education* and *health*), forming a semantic community. When a new peer wishes to join the system, it has to find its corresponding knowledge domain. Within a semantic community, peers are organized in a finer grouping level, i.e. clusters. In other words, semantically similar peers are clustered in a super-peer network considering their exported schemas.

As illustrated in Figure 4.1, three distinct types of peers are considered in the proposed system: *data peers*, *integration peers*, and *semantic peers*. A **data peer** is a simple computer or a server storing an autonomous data source. DP_{ij1}, DP_{ij2}, and DP_{ijk} are examples of data peers.

Data peers are logically organized in a super-peer network. In this sense, **semantic clusters** are formed according to data peers' exported schema. Each semantic cluster has a special type of peer named **integration peer**. In fact, an integration peer is a data peer with higher computational capacity. It is responsible for defining and maintaining schema mappings as well as for managing query processing and data peer's metadata. For instance, IP_{ij} is the integration peer of the cluster formed by the data peers DP_{ij1}, DP_{ij2}, and DP_{ijk}.

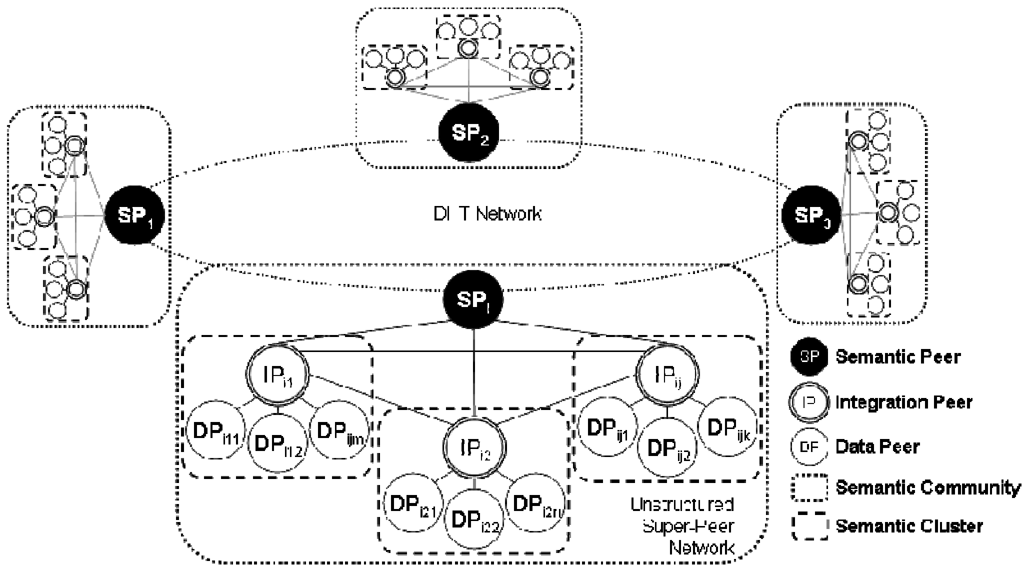


Figure 4.1. An overview of the SPEED architecture.

A set of clusters sharing content belonging to a common knowledge domain forms a **semantic community**. Each semantic community has a special type of peer named **semantic peer**. SP_i is an example of a semantic peer. Semantic peers are connected through a DHT network, while integration peers are connected through an unstructured network. Our approach assumes that an integration peer names its respective cluster, while a semantic peer names its corresponding semantic community (e.g. semantic cluster IP_{ij} and semantic community SP_i). The semantic community SP_i is formed by the clusters IP_{i1}, IP_{i2}, and IP_{ij}.

4.2.2 Architecture Formalization

In this section, we provide a formalization of some important concepts and terminologies which are necessary to a complete understanding of the SPEED's architecture. Other important definitions are presented in Section 4.3.

Definition 1 (Data Peer). A data peer DP_{ijk} is a simple computer or a server storing an autonomous data source, e.g. a relational database. The content shared by a data peer is accessed through an exported schema. Such schema provides access to the entire content available in its data source or only to a portion of it.

Definition 2 (Integration Peer). An integration peer IP_{ij} is a distinguishing data peer offering better computational resources in terms of availability, network bandwidth, processing power, and storage capacity. Therefore, an IP_{ij} is responsible for managing important tasks in the proposed PDMS, e.g. query processing and data integration.

Definition 3 (Requesting Peer). A requesting peer RP_n is a peer wishing to join the system. To this end, a RP_n must provide an exported schema. Once connected, a RP_n assumes the role of a data peer or an integration peer.

Definition 4 (Semantic Cluster). A semantic cluster CL_{ij} (cluster, for short) corresponds to a logical set of data peers sharing semantically similar exported schemas. One of the data peers in a semantic cluster CL_{ij} is necessarily an integration peer IP_{ij}. A data peer DP_{ijk} participates in only one semantic cluster CL_{ij}. Formally, $CL_{ij} = \{IP_{ij}, (DP_{ij1},\ldots,DP_{ijk})\}$, where k is the number of data peers in CL_{ij}; with $k \geq 0$. If $k = 0$, then a semantic cluster CL_{ij} contains only an integration peer IP_{ij}. For instance, at the moment a cluster is created.

Definition 5 (Semantic Community). A semantic community CM_i (community, for short) is a logical set of semantic clusters sharing content associated to a common knowledge domain, e.g. *education* and *health*. Formally, $CM_i = \{SP_i, (CL_{i1},\ldots,CL_{ij})\}$, where j is the number of semantic clusters in a semantic community CM_i; with $j \geq 1$.

Definition 6 (Semantic Peer). A semantic peer SP_i is a special type of peer associated to a semantic community CM_i. It acts as an entry point for its semantic community. A SP_i is responsible for forwarding a requesting peer to an initial semantic cluster CL_{ij}. Such initial cluster is obtained from a semantic index. Only one semantic peer is allowed per semantic community.

Definition 7 (Semantic Index). A semantic index is a structure located at a semantic peer SP_i describing the content available in the clusters (CL_{i1},…,CL_{ij}) of a semantic community CM_i. Each index entry represents an individual cluster CL_{ij} and contains a pointer to the corresponding cluster. In short, a semantic index stores the following information: (i) *cluster summary*: a summarized representation of the schemas shared by the peers of a particular cluster CL_{ij}; and (ii) *cluster address*: the network address of the corresponding integration peer IP_{ij}. A semantic index is used to assist the connection of requesting peers. Such process is detailed in Chapter 7.

Definition 8 (SPEED). SPEED is a Semantic-based PDMS composed of multiple semantic communities. Formally, SPEED = {CM_1,…,CM_i}, where *i* is the number of semantic communities; with $i \geq 1$.

Definition 9 (Semantic Neighbor). Two distinct clusters CL_{ij} and CL_{ik} are considered semantic neighbors (neighbors, for short) if they (i) belong to the same community CM_i; (ii) share semantically similar content; and (iii) are one-distance neighbors in CM_i's overlay network. Particularly, the item (ii) is better detailed in Chapter 7.

Definition 10 (Semantic Neighborhood). The set of semantic neighbors of a cluster CL_{ij} composes the semantic neighborhood (neighborhood, for short) of CL_{ij} and is denoted by N_{ij}.

Definition 11 (Relevant Peer). Given that a query Q is being processed at an integration peer IP_{ij}, each semantic neighbor of IP_{ij} or each data peer DP_{ijk} in the corresponding cluster CL_{ij} that is capable of answering Q (integrally or partially) is considered a relevant peer.

4.3 Ontologies in SPEED

In SPEED, ontologies are employed in many different ways. The semantic knowledge, expressed by ontologies, is used as a way to enrich some important PDMS services and provide users with more complete results. Since peers can store heterogeneous data sources, ontologies are used as the system's common data model. Moreover, considering a cluster of semantically similar peers, an ontology is used as a conceptual representation of the cluster to provide a shared understanding of the terms that are being shared. Particularly, in this work our main interest is in demonstrating how ontologies can be employed to assist peer organization in the proposed OPDMS network.

In this sense, four distinct types of ontologies are employed (Figure 4.2): (i) *local ontologies*, resembling the schema of the data sources stored in data peers and integration peers; (ii) *cluster ontologies*, representing the schemas of the peers in the semantic clusters; (iii) *summarized cluster ontologies*, representing a cluster ontology in a succinct way; and (iv) *community ontologies*, containing concepts and properties of a particular knowledge domain. Next, we provide a formal definition of these types of ontologies.

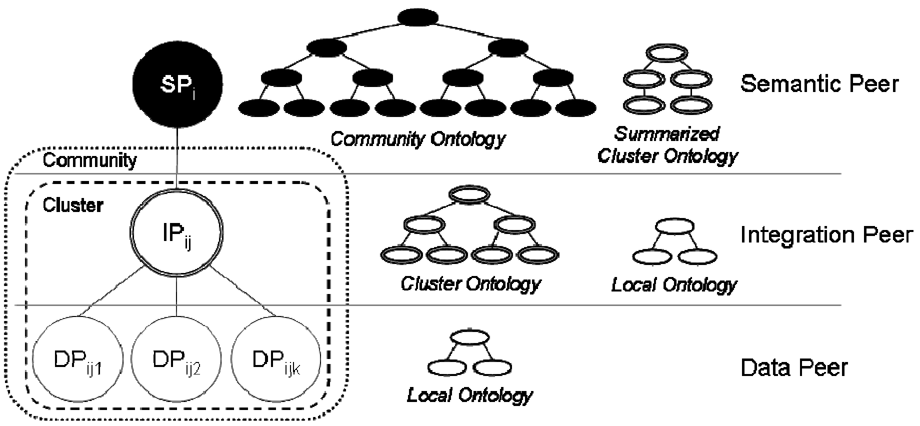


Figure 4.2. The different types of ontologies used in SPEED.

Definition 12 (Local Ontology). A local ontology LO_{ijk} corresponds to the exported schema describing the content shared by a data peer DP_{ijk}, an integration peer IP_{ij}, or a requesting peer RP_n.

Definition 13 (Cluster Ontology). A cluster ontology CLO_{ij} describes the content available in a semantic cluster CL_{ij}. A CLO_{ij} is obtained by merging the local ontologies (LO_{ij1},...,LO_{ijk}) of the data peers in a semantic cluster CL_{ij}. It is stored in an integration peer IP_{ij} which also maintains its local ontology LO_{ij}.

Definition 14 (Community Ontology). A community ontology CMO_i is an ontology describing the knowledge domain associated to a semantic community CM_i. A CMO_i is stored at a semantic peer SP_i.

Definition 15 (Summarized Cluster Ontology). A summarized cluster ontology OS_{ij} corresponds to an abridged version of a cluster ontology CLO_{ij}. An OS_{ij} is a subontology of CLO_{ij} containing the most relevant elements of CLO_{ij}. The process of summarizing a cluster ontology is described in Chapter 6.

In order to describe the different types of ontologies employed in SPEED, consider the setting illustrated in Figure 4.3 in which a semantic community is composed of two clusters. Each cluster contains an integration peer and two data peers. The setting is considered until the end of this chapter to explain the main ontology management processes used in SPEED.

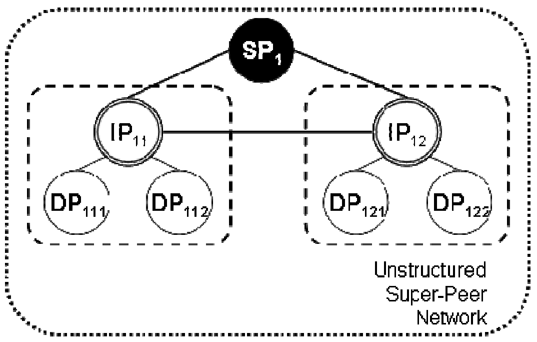


Figure 4.3. A setting of a semantic community containing two clusters.

In the described setting, the participating peers store content related to the *education* knowledge domain and wish to share data about universities and the corresponding activities occurring at them, e.g. given courses and publications. Thus, assume that each participating peer shares an exported schema as follows:

Cluster 1 (CL_{11})

Integration Peer IP_{11} (Object-Relational Model)
Publication (Title, PublicationDate, isAuthoredBy)
Author (Name, EmailAddress, isAuthorOf)
Article (Pages) isA Publication

Data Peer DP_{111} (Relational Model)
Researcher (ResID, FirstName, LastName, University)
Publication (PubID, Title, PubDate, Area, ResID)
Article (ArtID, FullPaper, PubID)
Book (BookID, ISBN, Chapters, Publisher, PubID)
Journal (PubID, Volume)
PublicationResearcher (PubID, ResID)

Data Peer DP_{112} (Object-Oriented Model)
Proceedings (ProceedingName, hasPublication)
Author (AuthorName, EmailAddress, isAuthorOf)
Publication (Title, Pages, isAuthoredBy)
Conference (Name, Year, Area, hasProceeding)

Cluster 2 (CL_{12})

Integration Peer IP_{12} (Object-Oriented Model)
Student (StudName, EmailAddress)
GraduateStudent (advisor) isA Student
UndergraduateStudent (ConclusionYear) isA Student
Professor (ProfName, EmailAddress, advices)

Data Peer DP_{121} (Object-Relational Model)
Course (CourseName) isA Work
Faculty (FacultyName, EmailAddress, isTeacherOf)
Student (StudentName, EmailAddress, takesCourse)
GraduateCourse (University) isA Course
Work (Manager)

Data Peer DP_{122} (Relational Model)
University (UnivID, UnivName, Homepage)
Faculty (FacID, FirstName, LastName, UnivID)
Student (StudID, StudentName, EmailAddress, UnivID, FacID)

Moreover, consider that, before joining the system, the participating peers have built an ontological description resembling the structure of their exported schemas. In other words, each peer has translated its exported schema, originally described in its data source metadata model, onto a **Local Ontology (LO)**. Such translation is needed because data sources can be heterogeneous. For the sake of correct query processing, the translation must preserve the structure and the integrity constraints (e.g. relational foreign keys) originally expressed on the exported schemas.

Figure 4.4 depicts the local ontologies of participating peers. All ontologies in this section are represented in OWL [Smith *et al.*, 2004] and depicted using the OntoViz (Protégé plug-in) notation[6]. In order to simplify the example, properties are not exhibited.

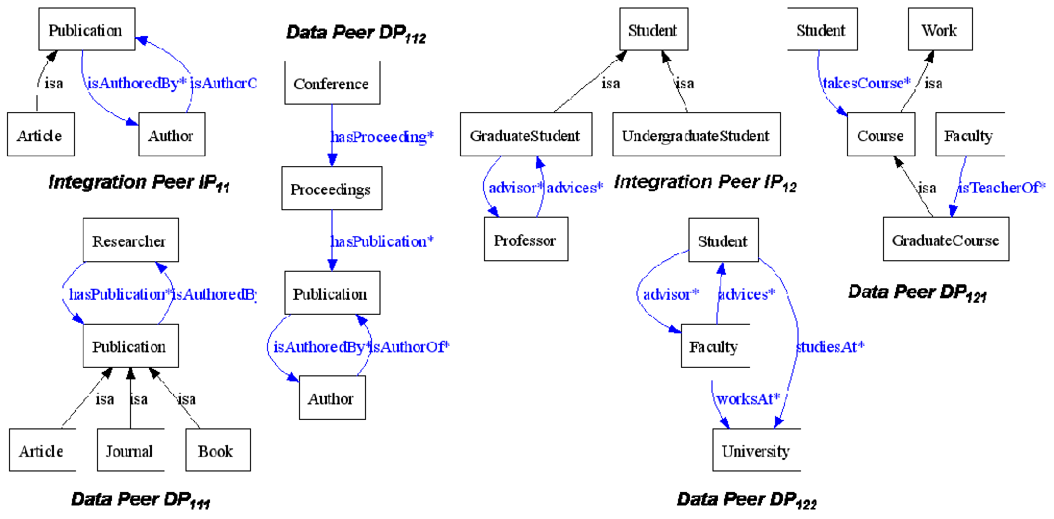


Figure 4.4. Local ontologies of participating peers.

In SPEED domain ontologies are used as a semantic reference at the community level. A **Community Ontology (CMO)** is a domain ontology offered by a semantic peer which is used as a semantic reference by all current clusters within the corresponding community. In our setting, we have considered the community ontology (*UnivCSCMO.owl*) described in the Appendix A.

[6] http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz#nid6CS

A **Cluster Ontology (CLO)** is obtained by merging the local ontologies of the peers participating in a particular cluster. In fact, a cluster ontology acts as a shared vocabulary inside a semantic cluster, inter-relating semantically similar ontological concepts. In our setting, considering the local ontologies of participating peers (Figure 4.4), we have the following two cluster ontologies diagrammed in Figure 4.5.

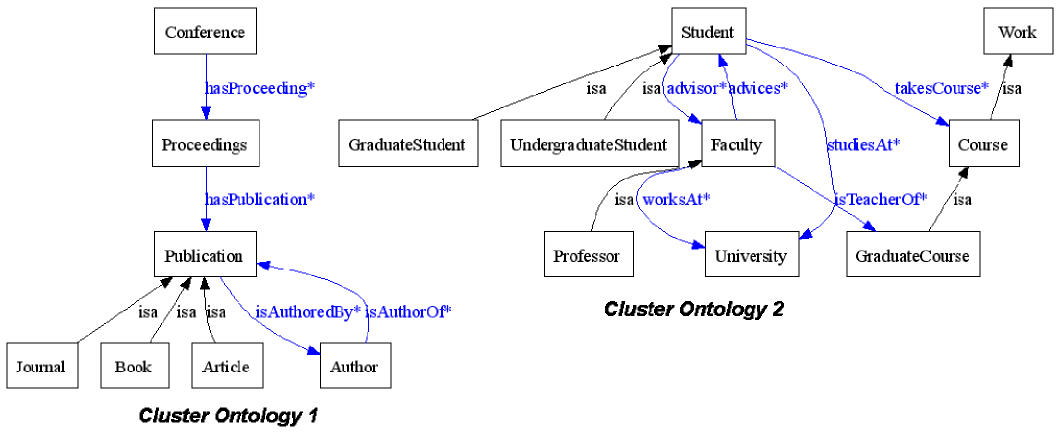


Figure 4.5. The cluster ontologies of Cluster 1 and Cluster 2.

To assist the connection of requesting peers, each cluster ontology is represented as a **Summarized Cluster Ontology (OS)**. The summaries are kept at the semantic index of a semantic peer. Figure 4.6 depicts the summarized cluster ontologies corresponding to the cluster ontologies of our example.

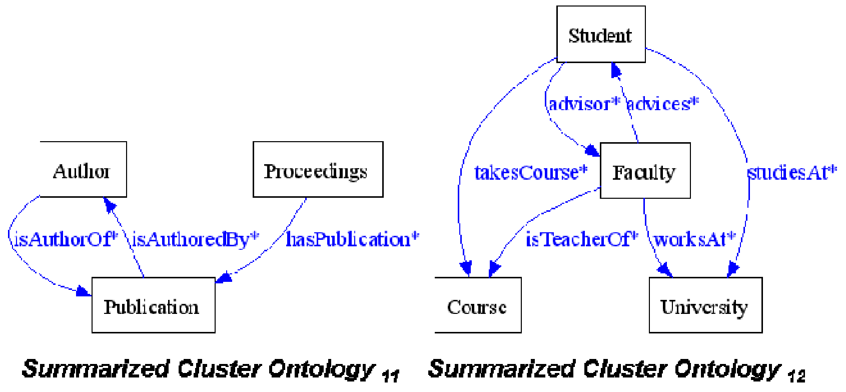


Figure 4.6. The summarized cluster ontologies of Cluster 1 and Cluster 2.

Ontologies are handled by appropriate modules located at the peers. In the following section, we detail the internal modules of each type of peer.

4.4 Peer Internal Modules

In the previous sections, we have seen that three types of peers are considered in SPEED: data peers, integration peers, and semantic peers. Since they can assume distinguishing roles in the system, they have different internal modules. Moreover, each module can be subdivided into several components as described in what follows.

4.4.1 Data Peer

A data peer is presumed to be any kind of peer (e.g. a simple computer or a server) which can frequently connect and disconnect from the P2P network. In fact, a data peer corresponds to a data source whose content is shared with other data peers through the establishment of semantic mappings. As illustrated in Figure 4.7, a data peer is composed of several modules and components.

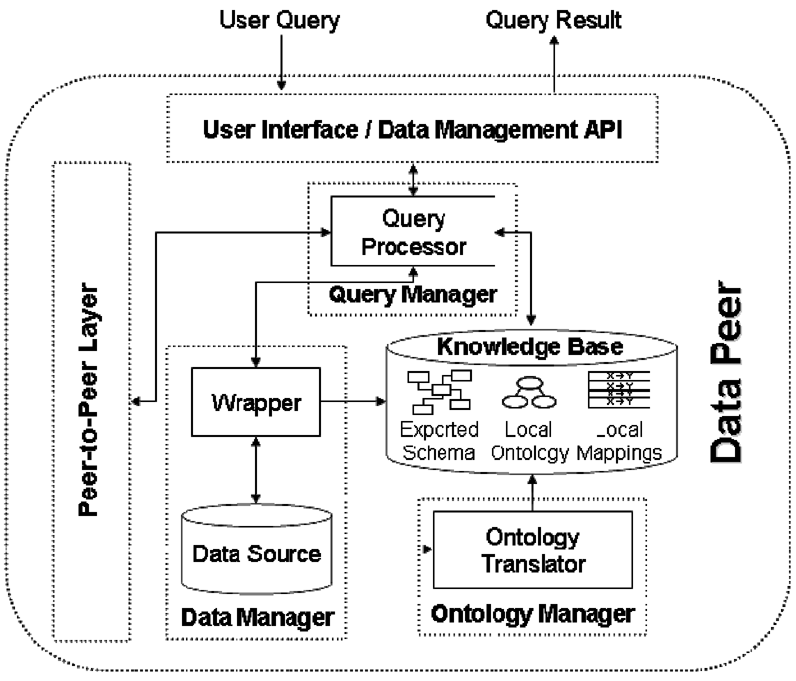


Figure 4.7. The internal modules of a data peer.

- *Peer-to-Peer Layer*: used for communication with an integration peer;
- *User Interface / Data Management API*: represents an interface for users to submit queries according to their exported schema;
- *Connectivity Manager*: responsible for managing data peer connectivity in the system. In a data peer, this module is composed of only one component:

    - *Ontology Translator*: translates the exported schema originally described using the data source metadata model to an ontology metadata model. As output, this component produces a local ontology along with local correspondences between elements of the exported schema and the local ontology.
- *Data Manager*: provides access to the content available in a data source through the exported schema. This module is subdivided into two components:
    - *Wrapper*: translates user queries from the common query language to the data source query language, and vice-versa. Such component is also responsible for translating query results from the source data model to the common data model adopted in the system, i.e. to an ontology metadata model;
    - *Data Source*: a data repository containing the content available in a data peer, e.g. a relational database. Only the content described by the exported schema can be accessed by the other peers in the network.
- *Query Manager*: responsible for managing user queries (i) submitted at the data peer or (ii) received from an integration peer. The participation of a data peer in query processing is discussed in Section 4.7. In a data peer, the query manager module is composed of only one component:
    - *Query Processor*: processes user queries at the data peer.
- *Knowledge Base*: stores an exported schema, a local ontology, and the local correspondences between them. Also, it maintains information (e.g. network address) used for communicating with its corresponding integration peer.

4.4.2 Integration Peer

The internal modules of data peers and integration peers are basically the same. However, when a peer acts as a data peer, some modules (those associated to an integration peer) are disabled. Since an integration peer is also a data peer, it may also store a data source. Thereby, its shared content is also considered during query processing. The internal modules of an integration peer are depicted in Figure 4.8.

In addition to the specific modules of a data peer described previously, the other modules of an integration peer include:

- *Peer-to-Peer Layer*: provides communication with (i) data peers in the same cluster; (ii) other integration peers in the same community; (iii) its corresponding semantic peer; and (iv) requesting peers;

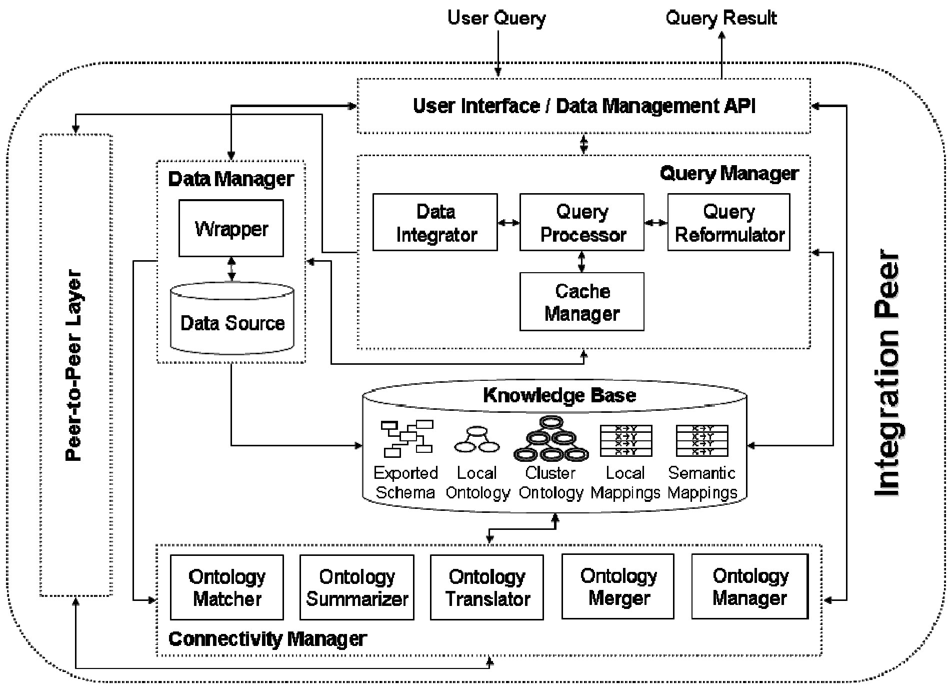


Figure 4.8. The internal modules of an integration peer.

- *Knowledge Base*: stores a cluster ontology, an exported schema, a local ontology, and the local correspondences. Besides, it contains semantic mappings which are essential for query processing. The knowledge base also maintains information (e.g. network address) for communicating with data peers, integration peers (i.e. the semantic neighbors), semantic peer, and requesting peers;
- *Query Manager*: responsible for managing user queries formulated (i) at an integration peer; (ii) received from the data peers; or (iii) received from other integration peers. The participation of an integration peer in query processing is discussed in Section 4.7. In an integration peer, the query manager module is subdivided into the following components:
  - *Query Processor*: processes user queries at the integration peer. It also determines the relevant peers to which a query must be sent;

  - *Query Reformulator*: reformulates a user query in such a way that the same query can be executed in data peers and other integration peers according to the terminology of their respective ontologies;
  - *Data Integrator*: integrates query results returned from data peers and other integration peers;
  - *Cache Manager*: maintains the local data cache of an integration peer.
- *Connectivity Manager*: manages integration peer connectivity in the network. It is responsible for assisting the connection of requesting peers and the disconnection of participating data peers. Since exported schemas are represented as ontologies peer connectivity is assisted by ontology processes, e.g. matching and merging. Thus, in an integration peer, this module is subdivided into the following components:
  - *Ontology Matcher*: automatically matches two peer ontologies and returns (i) an ontology alignment and (ii) a semantic similarity measure between the two (peer) ontologies. The matched ontologies can be a cluster ontology and a requesting peer's local ontology. In this case, the matching process determines if a requesting peer is able to join a cluster. The matched ontologies can also be two cluster ontologies. In this case, the matching process determines if two clusters are (still) semantic neighbors in the network. This component is detailed in Chapter 5;
  - *Ontology Merger*: takes as input an ontology alignment as well as the associated ontologies. As a result, it automatically produces a merged ontology containing the elements of both input ontologies as well as semantic correspondences between them;
  - *Ontology Summarizer*: automatically produces a summarized version of a cluster ontology. This component is invoked whenever a cluster ontology is modified. For instance, when a requesting peer joins a cluster. Its specification and implementation is detailed in Chapter 6;
  - *Ontology Manager*: updates a cluster ontology whenever a data peer disconnects from a cluster. The component removes from the cluster ontology the elements which were shared only by the disconnected data peer. Moreover, it eliminates the associated semantic correspondences. Considering that a data peer can reconnect to the same cluster, the

information is logically deleted. Physical deletion only occurs after a certain time interval.

4.4.3 Semantic Peer

A semantic peer acts as a knowledge (ontologies and metadata) server in the peer hierarchy offering a community ontology. Since a semantic peer does not store a data source, it does not participate in query processing. As illustrated in Figure 4.9, a semantic peer is composed of several modules and components described as follows.

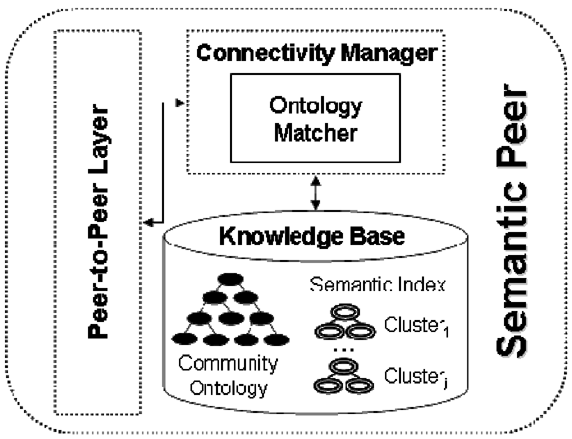


Figure 4.9. The internal modules of a semantic peer.

- *Peer-to-Peer Layer*: used for communication with (i) integration peers in the same community; (ii) other semantic peers; and (iii) requesting peers;
- *Knowledge Base*: stores a community ontology and a semantic index. Both structures are used as auxiliary information during the connection of requesting peers;
- *Connectivity Manager*: responsible for assisting the connection of requesting peers and updating the semantic index. In a semantic peer, this module is composed of a single component:
  - *Ontology Matcher*: matches a requesting peer's local ontology against the summarized cluster ontologies available at the semantic index.

4.5 Architectural Considerations

In SPEED, the connection of requesting peers starts by the DHT network. Such network is used to facilitate resource discovery by assisting requesting peers to "easily" find a semantically related community. DHT networks are characterized by efficient searches and sensibility to changes in their structure. They usually work with some kind of pointer tables, pointing to peers "closer" to the desired data. These pointer tables need to be updated at each connection and disconnection of a peer. Thus, SPEED's DHT network is only composed of semantic peers, i.e. peers with high reliability, network bandwidth, and availability. Excluding dynamic peers from the DHT network avoids unnecessary maintenance costs. In addition, the DHT network helps to forward requesting peers to corresponding communities which are more likely to be achieved with a smaller number of hops.

In cluster-based architectures, if a peer participates in more than one cluster, inefficiencies can be introduced [Vdovjak *et al.*, 2006]. For example, a query can be answered by all the clusters that include the (redundant) peer, resulting in duplication of query results and in an increase of communication effort. Therefore, in SPEED a peer takes part in only one cluster of a given semantic community.

If unrelated peers are neighbors in the network then semantic mappings can be incorrect or inconsistent. Clustering peers according to their exported schema (i.e. local ontology) provides an environment that is better suited to the establishment of schema mappings. Besides, as each integration peer maintains a description of its attached data peers, query routing is easily carried out. Furthermore, the physical heterogeneity of participating peers is also exploited in the selection of integration peers.

The construction of semantic clusters representing multiple data peers allows queries to navigate only among cluster ontologies, i.e. integrated schemas. A query received by an integration peer is processed in the relevant data peers inside the corresponding cluster. In this way, query results are more precise, since they come from semantically similar peers. Moreover, these peers receive a query version that has not been successively reformulated multiple times, and most importantly, through a cluster ontology that is as lossless as possible in terms of semantics. Therefore, the query that the integration peers receive is not degraded so much as it is through multiple successive reformulations [Tatarinov and Halevy, 2004; Kantere *et al.*, 2009].

Due to the different (and possibly numerous) cluster ontologies available in a semantic community, starting the search for a semantically similar cluster in an ad-hoc manner can be too costly in terms of time and network bandwidth. Ideally, the search should begin at a promising cluster and continue through other semantically similar clusters until a certain limit is reached. This strategy increases the probability that a requesting peer will quickly find a semantically similar cluster. In this sense, the basic idea is to search for the cluster that is probably more semantically similar to the requesting peer. To this end, a summarized version of each cluster ontology is used as an entry in a semantic index. We assume that a summary does not represent a cluster ontology in its entirety. Therefore, an initial cluster does not necessarily means the cluster to which a requesting peer will be connected. In the following sections, we describe how schema mappings and query processing are handled in SPEED.

4.6 Schema Mapping

Since exported schemas are represented by ontologies, schema mappings in SPEED are also named *semantic mappings* or *ontology mappings*. Semantic mappings describe the correspondences between elements of two distinct ontologies. According to the type of ontologies, semantic mappings can be subdivided into *cluster-to-local mappings* and *cluster-to-cluster mappings*.

Cluster-to-local mappings are semantic mappings between a cluster ontology and a local ontology. They are created when a requesting peer joins a cluster as a data peer and are removed when a data peer disconnects from a cluster. These mappings are stored at integration peers.

Cluster-to-cluster mappings are semantic mappings between the cluster ontology of two integration peers (semantic neighbors). They are created when a new cluster is formed. In this case, cluster-to-cluster mappings are defined between the new cluster and its corresponding semantic neighbors. They are updated when a cluster ontology is modified. In general, a cluster ontology is modified when a data peer joins a cluster, or a data peer (or integration peer) leaves the cluster. Such modification is needed to reflect the content shared in the referred cluster. Cluster-to-cluster mappings between any two clusters are stored at both integration peers.

4.7 Query Processing

In SPEED, queries can be posed and answered by data peers and integration peers. They are formulated according to the peer's exported schema and internally translated into the common query model. A query is disseminated only among the clusters of the semantic community where it was posed. Semantic peers do not participate in query processing. Consequently, if a query is submitted at a particular semantic community the query is not forwarded to other communities. During the navigation, a query is reformulated according to previously established semantic mappings. Integration peers are responsible for integrating query results received from its data peers and other integration peers.

To better explain how query processing is handled in SPEED, consider again the setting described in Figure 4.3 containing one semantic community and two clusters. Assume that queries and exported schemas are represented in a common data model. Suppose that a query Q_1 is submitted at the data peer DP_{111} (Figure 4.10). In this case, Q_1 is translated into a query Q_1' described in the common query model. Afterwards, Q_1' is sent to its corresponding integration peer IP_{11} in order to be propagated in the community. In parallel, Q_1 is executed at the data peer DP_{111} and the query result (R_1) is sent to the integration peer IP_{11} to be integrated with other query results possibly received from other clusters.

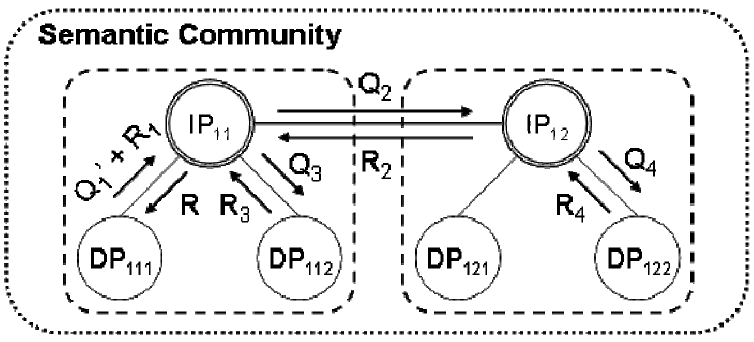


Figure 4.10. An example of query processing in SPEED.

Once the integration peer IP_{11} receives Q_1', it uses its semantic mappings to identify relevant peers for Q_1'. In the example, assume that the relevant peers are the data peer DP_{112} as well as the integration peers IP_{11} and IP_{12}. Thus, Q_1'

is reformulated into the queries Q_2 and Q_3 and forwarded to IP_{12} and DP_{112}, respectively. Particularly, Q_1' is executed at IP_{11} where the corresponding query result will be integrated with the ones obtained from DP_{111}, DP_{112}, and IP_{12}. Assume that IP_{12} has identified DP_{122} and IP_{12} as relevant peers (DP_{121} is not relevant). Thus, Q_2 is reformulated into Q_4 and forwarded only to DP_{122}.

When the reformulated query Q_3 arrives at the data peer DP_{112}, Q_3 is executed at the data source and the query result (R_3) is returned to the integration peer IP_{11} for data integration. Similarly, the data peer DP_{122} executes Q_4 in its local data source and returns its query result (R_4) to the integration peer IP_{12}. At IP_{12}, the query result R_3 is integrated with the one of IP_{12} and returned to IP_{11}.

At the integration peer IP_{11}, the query results R_1', R_2, and R_3 are integrated. The final query result (R) is then returned to the data peer DP_{111} where the initial query Q_1 was posed. At DP_{111}, the integrated query result is finally presented to the user who has formulated Q_1.

4.8 Comparative Analysis of SPEED and Related PDMS

In this section, we present a comparative analysis between SPEED and the three PDMS discussed in Chapter 3 (*OntSum*, *Sunrise*, and *Helios*). Similarly to SPEED, all of them employ a semantic-based approach to organize peers in their corresponding network. The comparison is illustrated in Table 4.2 which extends Table 3.1 with new a line summarizing SPEED.

Table 4.2. A comparison of PDMS employing a semantic-based approach to organize peers in the network (including SPEED).

PDMS	Schema Repres.	Network Topology	Network Population	Domains	Multiple Communities or Clusters	Neighborhood Search	Semantic Similarity Measure	Neighborhood Selection
OntSum	Ontologies	Unstructured	Not empty	Predefined semantic domains	No	Flooding; first short-distance links, then long-distance links; inter-cluster table	Semantic similarity measure (ontology matching)	Threshold
Sunrise	Generic (Ontologies, Relational, XML)	Unstructured	Empty	Non existing domains	Yes	Centralized Access Point Structure (APS) followed by SCI	Semantic distance between concepts (clustroids)	Range-based and kNN-based algorithms
Helios	Ontologies	Unstructured	Not empty	Non existing domains	Yes	Flooding	Ontology matching	Threshold
SPEED	Ontologies	Mixed (DHT, unstructured and super-peer)	Empty	Predefined semantic domains	No	Mixed (semantic index and flooding)	Ontology matching	Threshold

Except for Sunrise in which semantic similar peers are discovered through a centralized index (APS), all the other systems use a flooding strategy for such task. Particularly, the flooding strategy of OntSum first tries to find out similar peers using short-distance links and then long-distance links. In SPEED, a mixed search strategy is used to discover semantically similar peers. First, requesting peers search for a corresponding community in a DHT network. If such community is found, then a semantic index is used to forward a requesting peer to an initial cluster where a flooding strategy is employed to discover a semantically similar cluster (if it exists). The search strategy used in SPEED is detailed in Chapter 7. Differently from Sunrise, the semantic index proposed in SPEED cannot be seen as a structure that fully controls peer connectivity in the system. Instead, its goal is only to provide a promising initial cluster to a requesting peer. Moreover, in SPEED the frequency of index updates is minimized since summarized cluster ontologies contain the most frequent concepts of a cluster which are not constantly modified.

In Sunrise, the similarity between peers is calculated using the peer's representative concepts. Differently, the other two systems (Helios and OntSum) use ontology matching functions for the same task. Particularly, the ontology matching algorithm of Helios (H-Match) does not produce a global similarity measure between two (peer) ontologies. Besides, only a fragment of the peers' ontologies is considered during the member identification phase. Concerning OntSum, although a global similarity measure is employed, such measure is obtained through a simple and asymmetric function. Similarly to OntSum, SPEED uses an ontology matching function to produce a symmetric global similarity measure. Such function considers the complete peers' ontologies as well as the linguistic, structural, and semantic characteristics of the ontology elements. The ontology matching function used in SPEED is detailed in Chapter 5.

4.9 Considerations

In this chapter, we have proposed SPEED, a semantic-based PDMS whose overlay network is mainly designed to assist peer connection. We have also described the three types of peers (data peers, integration peers, and semantic peers), their internal modules, and the different types of ontologies used in the system (local ontologies, cluster ontologies, and community ontologies).

In the following chapters we detail the SPEED features specified in this work. In Chapter 5, we propose a global similarity measure between ontologies. The measure is obtained as a result of a semantic-based ontology matching process and is used to determine the similarity between peers in SPEED. In Chapter 6, we propose an automatic process to summarize cluster ontologies. The summaries are used to improve the ontology matching process. Instead of matching local ontologies against cluster ontologies which can be large in terms of the quantity of concepts, we match against their corresponding summaries. In Chapter 7, we propose an incremental process for clustering requesting peers into semantic clusters. The basic idea is to put together in the overlay network peers sharing semantically similar schemas. The clustering process makes use of the matching and summarization processes.

CHAPTER 5

SEMMATCH: A SEMANTIC-BASED ONTOLOGY MATCHING PROCESS

“Do not add days to your life but life to your days”
Harry Benjamin

In this chapter, we present **SemMatch**, a semantic-based ontology matching process [Pires *et al*., 2009b]. Basically, the process takes as input two (peer) ontologies and returns an ontology alignment as well as a global similarity measure between them. The resulting correspondences are generated as a combination of linguistic, structural and semantic alignments produced by existing ontology matching tools. Each correspondence in the resulting alignment is associated with (i) a *combined similarity* value which expresses the level of confidence between the elements; and (ii) the *semantic relationship* (e.g. equivalence and subsumption) between them.

In a PDMS such as SPEED in which peers are organized within semantic clusters according to their local ontologies, a global measure is needed for clustering semantically similar peers. Such global measure represents the overall similarity degree between two peer ontologies (and not only between their elements!). The measure is computed using the correspondences generated by our ontology matching process. To clarify matters, in this chapter we present a case study illustrating how the measure can be used. We also provide an experimentation of the ontology matching process with some obtained results.

5.1 Introduction

Traditional approaches to ontology matching mainly rely on linguistic and/or structural techniques. As a result, over the last years a high number of tools that employ a combination of these approaches have been developed [Aumüller *et al.*, 2005; Castano *et al.*, 2006; Hu and Qu, 2008]. However, there are cases in which linguistic and structural approaches fail to figure out relevant correspondences between ontology elements. In such cases, these approaches can be complemented by the semantic approach [Reynaud and Safar, 2007].

Besides producing a similarity value for each correspondence, semantic matching techniques are capable of identifying the semantic relationship (e.g. equivalence and subsumption) between ontology elements. To this end, they employ different kinds of solutions, e.g. a domain ontology or a thesaurus. Although semantic matching techniques can improve an ontology matching process they are still rarely exploited in the literature and thus only a few tools [Giunchiglia *et al.*, 2004] implement semantic matchers.

In this work, in order to determine the correspondences between ontology elements we use the semantic matcher proposed in [Souza, 2009]. Such matcher has been developed as part of the SPEED's project and therefore its description is useful to understand the proposed ontology matching process. Basically, in such matcher, the identification of correspondences between the elements of a source ontology with elements of a target one is assisted by a domain ontology which is used as background knowledge [Sabou *et al.*, 2006]. The basic idea is to first align the elements from both input ontologies with the corresponding domain ontology and then use the existing relationships of such domain ontology in order to derive semantic correspondences between the source and target ontologies' elements.

5.2 A Motivating Scenario

In SPEED, the task of clustering peers consists in grouping semantically similar peers into a semantic community and then into semantic cluster. To this end, a global similarity measure between peer ontologies is useful in several situations that characterize peer clustering. For instance, a global measure is needed to indicate (i) the initial cluster for a requesting peer from which it will start the search for a similar cluster; (ii) the semantic similarity between a requesting peer and a cluster; and (iii) the semantic neighbors of a cluster.

Considering such situations, we introduce a motivating scenario that reproduces the situation (ii) enumerated above. The scenario is used throughout the chapter and considers three peers: P_i, P_j, and P_k (Figure 5.1). All peers belong to the *Education* knowledge domain. Assume that P_i is an integration peer and P_j is a requesting peer. Moreover, consider that P_i and P_j have complementary data about academic people and their works (e.g. Research) from different institutions. The integration peer P_i is described by the cluster ontology O_i (*Semiport.owl*). The requesting peer P_j is described by the local ontology O_j (*UnivBench.owl*). Both are public ontologies which are available for download on the Web[7,8]. In addition, assume that P_k is a semantic peer offering a Domain Ontology (DO) as background knowledge. The domain ontology corresponds to a community ontology and is named *UnivCSCMO.owl.*

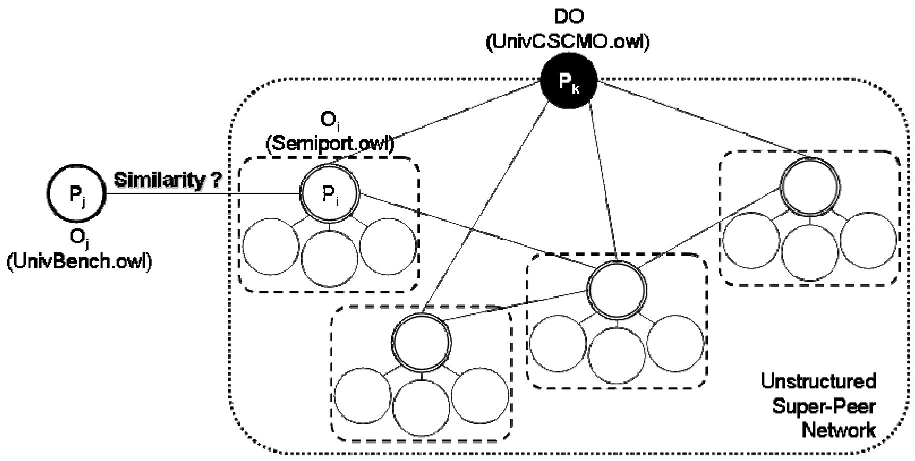


Figure 5.1. A motivating scenario for matching ontologies in SPEED.

Figure 5.2 shows excerpts from the three ontologies using OWLViz[9], a Protégé plug-in. A description of the ontologies *UnivBench.owl*, *Semiport.owl*, and *UnivCSCMO.owl* is provided in the Appendix A. Given the motivating scenario, we are interested in obtaining the semantic similarity between P_i and P_j in order to determine if P_j should join P_i's cluster. To this end, the ontologies

7 http://ontoware.org/frs/download.php/18/semiport.owl
8 http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl
9 http://www.co-ode.org/downloads/owlviz/

O_i (cluster ontology), O_j (local ontology), and DO (domain ontology) are considered.

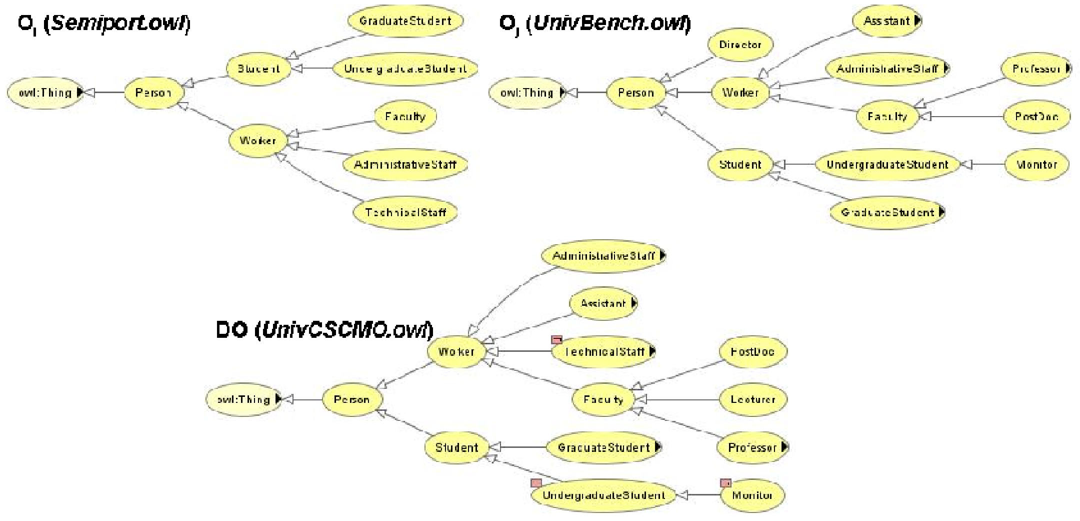


Figure 5.2. Excerpts from the ontologies of O_i, O_j, and DO. All of them belong to the *Education* knowledge domain.

To simplify matters, we assume that both ontologies O_i and O_j have been normalized [Rahm and Bernstein, 2001] to a uniform representation format according to the terms of the DO. In other words, element names from O_i and O_j have been adjusted to become compatible with the element names found in the DO. For instance, the concept *GraduateStudent* in the ontology O_j has been obtained from the original concept label (*GradStud*).

In the following section, we provide an overview of the semantic matcher used in the proposed ontology matching process. Basically, we enumerate the rules that can be applied to identify the semantic correspondences between ontology elements. The semantic matcher is not a contribution of this work and is described to facilitate the understanding of the proposed ontology matching process.

5.3 Using a Domain Ontology to Define Semantic Correspondences

Domain Ontologies (DO) contain concepts and properties of a particular knowledge domain and may be used as background knowledge in some important tasks. Particularly, we consider DO as reliable references that are available on the Web. In our scenario, a DO is used to bridge the conceptual differences or similarities between ontologies belonging to an integration peer (cluster ontology) and a requesting peer (local ontology).

In this sense, first concepts and properties from the two peer ontologies are mapped to equivalent concepts/properties in the DO and then their semantic correspondence is inferred based on the existing semantic relationship between the DO elements. Figure 5.3 shows an overview of the approach for specifying the semantics of the correspondences between peer ontologies. In this overview, O_i:x ≡ DO:k and O_j:y ≡ DO:z. Since *k* is subsumed by *z* in the DO, it is inferred that the same relationship occurs between *x* and *y*. Then, it is concluded that *x* in O_i is subsumed by *y* in O_j, denoted by $O_i.x \xrightarrow{\sqsubseteq} O_j.y$.

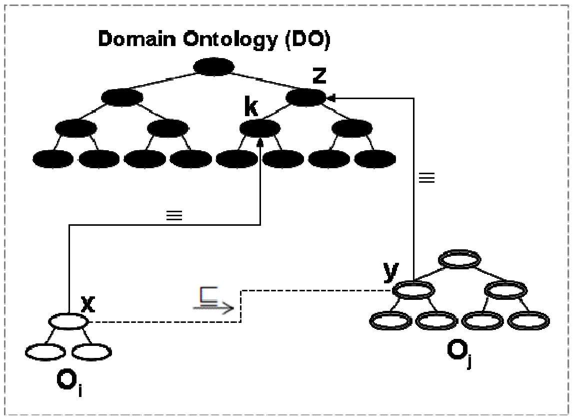


Figure 5.3. Specifying semantic correspondences between peer ontologies [Souza, 2009].

According to [Souza, 2009], a semantic correspondence can be one of the following expressions:

1. $O_i:x \xrightarrow{\equiv} O_j:y$, an *isEquivalentTo* correspondence
2. $O_i:x \xrightarrow{\sqsubseteq} O_j:y$, an *isSubConceptOf* correspondence
3. $O_i:x \xrightarrow{\sqsupseteq} O_j:y$, an *isSuperConceptOf* correspondence
4. $O_i:x \xrightarrow{\trianglerighteq} O_j:y$, an *isPartOf* correspondence
5. $O_i:x \xrightarrow{\triangleleft} O_j:y$, an *isWholeOf* correspondence
6. $O_i:x \xrightarrow{\approxeq} O_j:y$, an *isCloseTo* correspondence
7. $O_i:x \xrightarrow{\perp} O_j:y$, an *isDisjointWith* correspondence

where *x* and *y* are elements (i.e. concepts or properties) belonging to one ontology O_i and another ontology O_j respectively representing two semantic neighbor peer ontologies.

Considering the motivating scenario introduced in Section 5.2, in order to identify the semantic correspondences between O_i and O_j, first the semantic matcher finds out the equivalences between concepts of O_i and concepts in the DO, and the equivalences between concepts of O_j with their related ones in the DO. Then, the set of rules described in this section are applied. As a result, the set of semantic correspondences between O_i and O_j are identified. Examples of this set concerning the concept *Faculty* (from O_i) with some related concepts in O_j are presented in Table 5.1.

Table 5.1. Some semantic correspondences between O_i and O_j.

Correspondences for O_i:Faculty	
O_i:Faculty $\xrightarrow{\equiv}$ O_j:Faculty	O_i:Faculty $\xrightarrow{\sqsupseteq}$ O_j:PostDoc
O_i:Faculty $\xrightarrow{\sqsubseteq}$ O_j:Worker	O_i:Faculty $\xrightarrow{\approx}$ O_j:Assistant
O_i:Faculty $\xrightarrow{\sqsupseteq}$ O_j:Professor	O_i:Faculty $\xrightarrow{\approx}$ O_j:AdministrativeStaff

In this illustrative set, the equivalence correspondence between *Faculty* in O_i and O_j can be seen. Equivalence is an example of a commonly identified correspondence type in traditional ontology matching approaches. On the other hand, one can see that, taking into account the semantics underlying the DO, it is possible to identify other unusual correspondences. In this fragment, *Faculty* has been identified as: (i) sub-concept of *Worker*; (ii) super-concept of *Professor* and *PostDoc*; and (iii) close to *Assistant* and *AdministrativeStaff*.

5.4 SemMatch: a Semantic-based Ontology Matching Process

SemMatch is an ontology matching process that brings together a combination of already defined strategies with the semantic-based approach previously described. It is based on a composition strategy [Euzenat and Shvaiko, 2007] where linguistic-structural and semantic matchers are executed in parallel, and their individual similarity values are aggregated into combined similarity ones. As depicted in Figure 5.4, *SemMatch* receives as input two matching ontologies – O_i and O_j, as well as a domain ontology DO to be used as background knowledge. As output, it may produce one or two alignments (A_{CO} and/or A_{ij}), according to the following two possible objectives in the process instantiation:

- **Generating only the alignment A_{CO}**: this option corresponds to Phase 1. The global similarity measure is not calculated. Only the resulting set of correspondences identified by the linguistic-structural and semantic matchers is considered. In this set, a correspondence is defined between an element $e_i \in O_i$ and some matching elements $e_1,\ldots,e_j \in O_j$, considering the kind of semantic correspondence between them and its respective similarity value. Such alignment is useful for query processing purposes [Souza, 2009];

- **Calculating the global similarity measure**: in this option, Phases 1 and 2 are performed, i.e. both alignments A_{CO} and A_{ij} are generated. In order to generate A_{ij}, the correspondences in A_{CO} are ranked according to the combined similarity value and a filter strategy is applied to select the most suitable correspondences. Each correspondence in A_{ij} is defined between an element $e_i \in O_i$ and its best matching element $e_j \in O_j$, i.e. the element e_j having the highest similarity value with e_i. Based on the identified similarity value of each correspondence, the global measure is calculated.

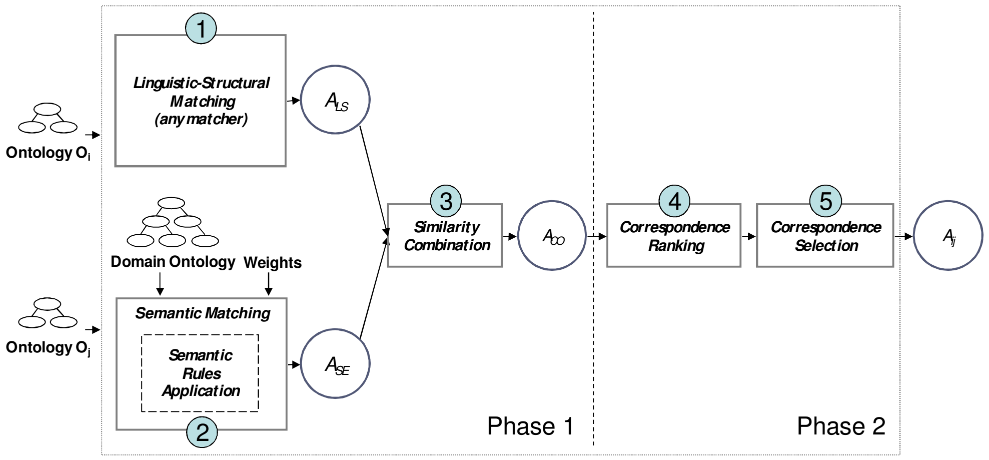
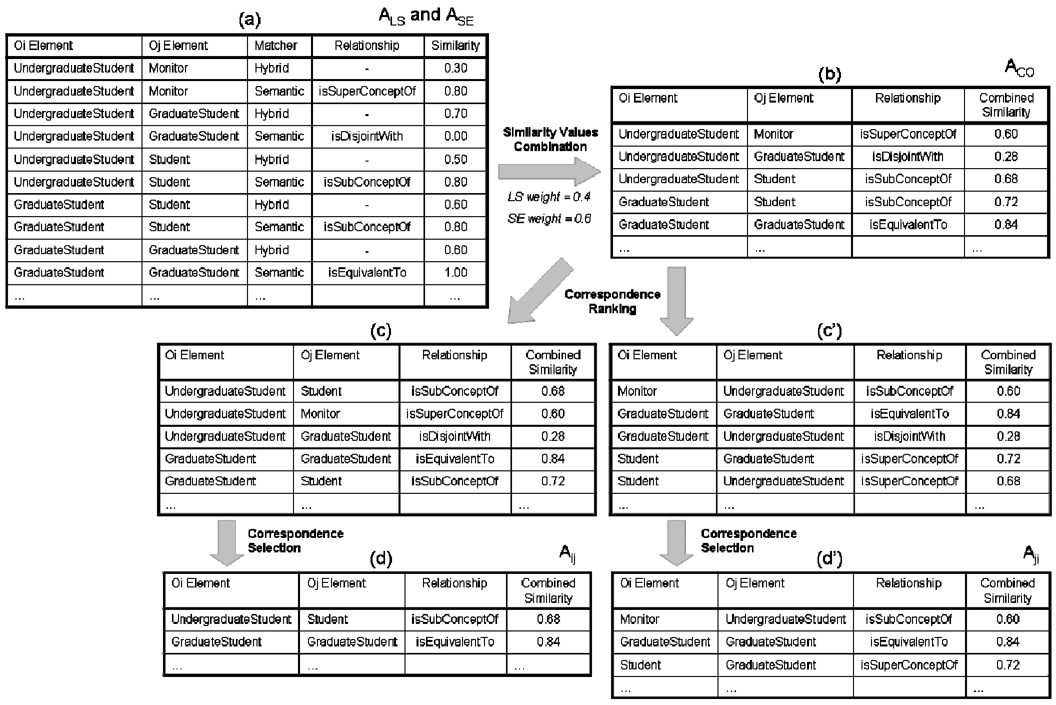


Figure 5.4. The general ontology matching process.

The main steps carried out by the semantic-based ontology matching process are described in the following.

Linguistic-Structural Matching

Since there are many available linguistic and structural matchers, we assume that any existing ontology matching tool including such category of matchers can be used. In *SemMatch*, the linguistic and structural matchers are handled as a hybrid matcher, i.e. as a fixed combination of simple matchers. The combination of their similarity values depends on the composition strategy of the ontology matching tool that has been used. The alignment produced by the hybrid matcher is denoted by A_{LS}. A correspondence in A_{LS} is a 3-tuple stated as $\langle e_i, e_j, n \rangle$. Figure 5.5 illustrates the overall process instantiation for the two ontologies O_i and O_j described in Section 5.2. To simplify, only a limited number of linguistic-structural correspondences are shown in Figure 5.5a. Among them, for instance, the similarity value generated by the hybrid matcher for the pair of elements (*UndergraduateStudent*, *Monitor*) is 0.30.



(a) A_{LS} and A_{SE}

Oi Element	Oj Element	Matcher	Relationship	Similarity
UndergraduateStudent	Monitor	Hybrid	-	0.30
UndergraduateStudent	Monitor	Semantic	isSuperConceptOf	0.80
UndergraduateStudent	GraduateStudent	Hybrid	-	0.70
UndergraduateStudent	GraduateStudent	Semantic	isDisjointWith	0.00
UndergraduateStudent	Student	Hybrid	-	0.50
UndergraduateStudent	Student	Semantic	isSubConceptOf	0.80
GraduateStudent	Student	Hybrid	-	0.60
GraduateStudent	Student	Semantic	isSubConceptOf	0.80
GraduateStudent	GraduateStudent	Hybrid	-	0.60
GraduateStudent	GraduateStudent	Semantic	isEquivalentTo	1.00
...	...	...		...

(b) A_{CO}

Oi Element	Oj Element	Relationship	Combined Similarity
UndergraduateStudent	Monitor	isSuperConceptOf	0.60
UndergraduateStudent	GraduateStudent	isDisjointWith	0.28
UndergraduateStudent	Student	isSubConceptOf	0.68
GraduateStudent	Student	isSubConceptOf	0.72
GraduateStudent	GraduateStudent	isEquivalentTo	0.84
...	...		...

(c)

Oi Element	Oj Element	Relationship	Combined Similarity
UndergraduateStudent	Student	isSubConceptOf	0.68
UndergraduateStudent	Monitor	isSuperConceptOf	0.60
UndergraduateStudent	GraduateStudent	isDisjointWith	0.28
GraduateStudent	GraduateStudent	isEquivalentTo	0.84
GraduateStudent	Student	isSubConceptOf	0.72
...	...		...

(c')

Oi Element	Oj Element	Relationship	Combined Similarity
Monitor	UndergraduateStudent	isSubConceptOf	0.60
GraduateStudent	GraduateStudent	isEquivalentTo	0.84
GraduateStudent	UndergraduateStudent	isDisjointWith	0.28
Student	GraduateStudent	isSuperConceptOf	0.72
Student	UndergraduateStudent	isSuperConceptOf	0.68
...	...		...

(d) A_{ij}

Oi Element	Oj Element	Relationship	Combined Similarity
UndergraduateStudent	Student	isSubConceptOf	0.68
GraduateStudent	GraduateStudent	isEquivalentTo	0.84
...	...		...

(d') A_{ji}

Oi Element	Oj Element	Relationship	Combined Similarity
Monitor	UndergraduateStudent	isSubConceptOf	0.60
GraduateStudent	GraduateStudent	isEquivalentTo	0.84
Student	GraduateStudent	isSuperConceptOf	0.72
...	...		...

Figure 5.5. An example of the ontology matching process.

Semantic Matching

A new kind of semantic matcher is proposed. Basically, such matcher uses a domain ontology as background knowledge and applies a set of semantic rules to derive the type of semantic correspondence for O_i and O_j elements, as explained in Section 5.3. Each type of semantic correspondence is associated with a given weight which corresponds to the level of confidence of such correspondence. The alignment generated by the semantic matcher is denoted by A_{SE}. A correspondence in A_{SE} is a 4-tuple stated as $\langle e_i, e_j, r, n \rangle$. Figure 5.5a illustrates some semantic correspondences for the two ontologies of our example. For instance, the semantic matcher has identified that the elements *UndergraduateStudent* and *Monitor* are related by the semantic correspondence *isSuperConceptOf*. The semantic correspondences have received the following weights: isEquivalentTo (1.00), isSubConceptOf (0.80), isSuperConceptOf (0.80), isPartOf (0.30), isWholeOf (0.30), isCloseTo (0.70), and isDisjointWith (0.00). The weights are attributed according to the semantic relevance of each correspondence. For instance, the semantic similarity value of (*UndergraduateStudent*, *Monitor*) is 0.80.

Similarity Combination

For each correspondence, the individual similarity values produced by the hybrid matcher and the semantic matcher are associated in a combined similarity one. The combined value is obtained through a weighted average of the similarity values generated by the individual matchers. The weights are attributed according to the relevance of each matcher. The combined alignment set generated in this step is denoted by A_{CO}. A correspondence in A_{CO} is a 4-tuple stated as $\langle e_i, e_j, r, n \rangle$.

A weighted average is used because matchers may produce opposing similarity values. For example, a linguistic matcher can find a low similarity value for two elements because their labels are completely different. On the other hand, a semantic matcher can detect that the same elements are related by a strong relationship (e.g. equivalence) and assign a high similarity value. Regarding our example, the similarity values generated by the hybrid and the semantic matchers for the pair of elements (*UndergraduateStudent*, *Monitor*) are 0.30 and 0.80, respectively. We assume that the weights associated to the hybrid and semantic matchers are 0.4 and 0.6, respectively. Thus, the combined similarity value produced for the pair (*UndergraduateStudent*, *Monitor*) is 0.60 (Figure 5.5b). Since we are interested in obtaining the global measure, the matching example continues in the next steps.

Correspondence Ranking

The correspondences involving each element of O_i are ranked (in descending order) based on the combined similarity values. In Figure 5.5c, the O_i element *UndergraduateStudent* is ranked in descending order according to the O_j elements *Monitor*, *GraduateStudent*, and *Student*.

Correspondence Selection

Finally, a filter strategy is applied to choose the most suitable correspondence for each O_i element. The strategy consists in selecting the correspondence with the highest combined similarity. Such decision represents the natural choice to guarantee at most one correspondence per O_i and O_j element. As a result of this step, an alignment A_{ij} is generated. A correspondence in A_{ij} is a 4-tuple: $\langle e_i, e_j, r, n\rangle$. In Figure 5.5c, for the concept *UndergraduateStudent*, the correspondence (*UndergraduateStudent*, *Student*) is preferred instead of (*UndergraduateStudent*, *Monitor*) or (*UndergraduateStudent*, *GraduateStudent*) because the combined similarity value of the first (0.72) is higher than the combined similarity of the last two (0.60 and 0.28). A fragment of the alignment A_{ij} is illustrated in Figure 5.5d.

Steps 4 and 5 are needed to define correspondences which are used to measure the global similarity measure between O_i and O_j (to be explained in Section 5.5). Such steps are also executed in the opposite direction, i.e. from O_j to O_i. The elements of O_j are ranked according to the elements of O_i (Figure 5.5c') and the same filter strategy is applied (Figure 5.5d'). An alignment A_{ji} is produced as output. Correspondences in A_{ji} have the same format of the ones in A_{ij}.

For the final alignment set, an O_i and an O_j element are only accepted as a matching correspondence if they are identified as such in both directions. In other words, a correspondence is included in the final alignment set if the correspondence is contained in the alignment sets A_{ij} and A_{ji}. In the following section, we will present our method for determining the global similarity measure.

5.5 Calculating the Global Similarity Measure

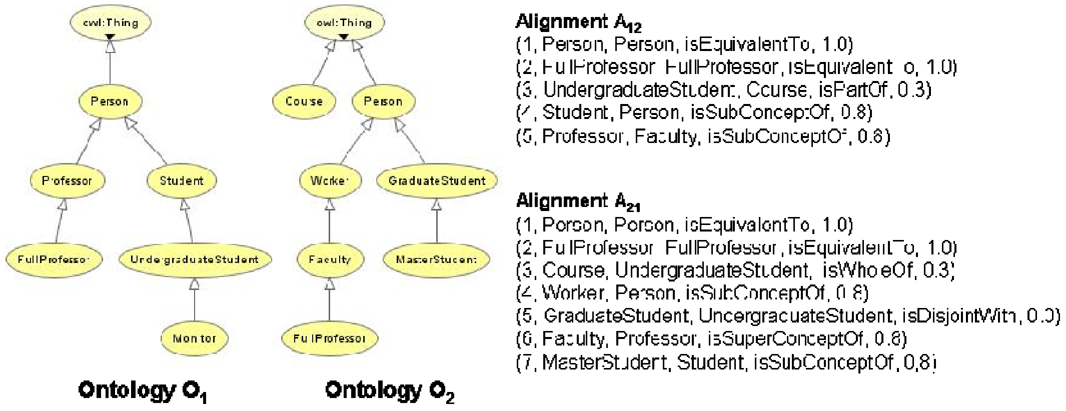
The evaluation of the overall similarity between the two ontologies O_i and O_j is an additional step in the proposed ontology matching process. Such step uses the output of the *Correspondence Selection* step (Step 5 of Section 5.4) which produces the alignment sets A_{ij} and A_{ji}. Both alignment sets are taken as input to calculate an overall similarity value between O_i and O_j. Such value indicates the global similarity measure between the two input ontologies.

There are several similarity measures available in the literature which can be adapted in order to calculate the global similarity measure between O_i and O_j [Castano *et al.*, 1998, Aumüller *et al.*, 2005, David and Euzenat, 2008]: *dice*, *weighted*, and *overlap*. All of them take into account the size of the input ontologies. The *dice measure* is derived from the dice coefficient [Rijsbergen, 1979]. It refers to the ratio between the number of matching elements of both alignments and the number of elements of the input ontologies. The *weighted average measure* refers to the ratio between the sums of the similarity values (*n*) of all matching elements of both alignments and the number of elements of the input ontologies. The *overlap measure* is derived from the overlap coefficient [Rijsbergen, 1979]. It refers to the ratio between the size of the intersection between both alignments and the number of elements of the shortest input ontology. In this work, the size of an ontology is determined by the number of its concepts and denoted by |O|.

As opposed to the *dice measure*, the global similarity degree computed by the *weighted average measure* is influenced by the individual similarity values. Hence, the *dice measure* returns higher similarity values than the *weighted average measure*. With all element similarities set to 1.0, both measures return the same similarity. However, in general not all correspondences are evaluated to the maximum level of confidence (1.0). Regarding the *overlap measure*, it is mostly used when the input ontologies are close to each other and have similar sizes. In practice, it is common to match ontologies with different sizes. For these reasons, in this work we use the *weighted average measure* to evaluate the global similarity degree between O_i and O_j. In this sense, the selected measure is determined as follows:

$$Weighted\ Average(Oi, Oj) = \frac{\sum_{i=1}^{|Aij|} n + \sum_{j=1}^{|Aji|} n}{|Oi| + |Oj|}$$

In order to demonstrate how the global measure is computed, consider the two ontologies (O_1 and O_2) illustrated in Figure 5.6 as well as the corresponding alignments A_{12} and A_{21} between them produced by *SemMatch*.



Alignment A_{12}

(1, Person, Person, isEquivalentTo, 1.0)
(2, FullProfessor, FullProfessor, isEquivalentTo, 1.0)
(3, UndergraduateStudent, Course, isPartOf, 0.3)
(4, Student, Person, isSubConceptOf, 0.8)
(5, Professor, Faculty, isSubConceptOf, 0.8)

Alignment A_{21}

(1, Person, Person, isEquivalentTo, 1.0)
(2, FullProfessor, FullProfessor, isEquivalentTo, 1.0)
(3, Course, UndergraduateStudent, isWholeOf, 0.3)
(4, Worker, Person, isSubConceptOf, 0.8)
(5, GraduateStudent, UndergraduateStudent, isDisjointWith, 0.0)
(6, Faculty, Professor, isSuperConceptOf, 0.8)
(7, MasterStudent, Student, isSubConceptOf, 0.8)

Figure 5.6. The ontologies O_1 and O_2 as well as the alignments A_{ij} and A_{ji} between them.

The sizes of O_1 and O_2 are 6 and 7 (concepts), respectively. In this sense, the global similarity measure between them is calculated as follows:

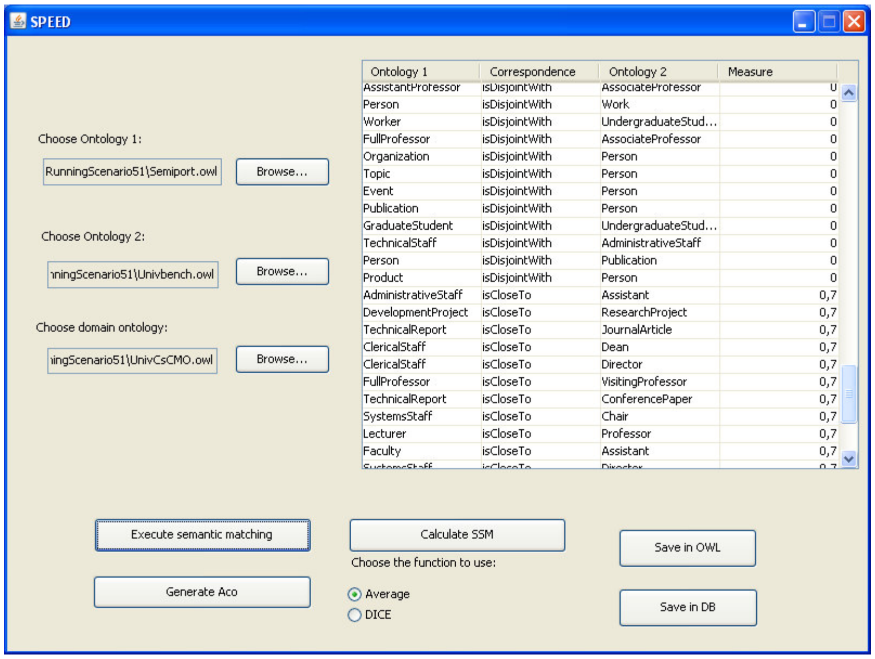
$$Weighted\ Average(O_1, O_2) = \frac{(1.0+1.0+0.3+0.8+0.8)+(1.0+1.0+0.3+0.8+0.0+0.8+0.8)}{|6|+|7|} = 0.66$$

Particularly, the global similarity measure between *Semiport.owl* and *UnivBench.owl* is 0.77. A complete description of the calculus considering the alignments A_{ij} and A_{ji} between them is available in the Appendix B.

5.6 Experiments and Results

An ontology matching tool implementing the semantic-based process was developed in Java. In order to provide ontology manipulation and reasoning, we used Jena [Jena, 2009]. In this current version, H-Match [Castano *et al.*, 2006] was used as the hybrid matcher and we restricted the correspondence identification to concepts (not including properties). Figure 5.7 shows a screenshot of the tool's main window that is split into three areas: (i) one for choosing the matching ontologies; (ii) another for depicting the resulting semantic correspondences and their respective weights; and (iii) the other one for executing the main options, concerned with identifying the semantic correspondences, generating the A_{CO} alignment, and calculating the global similarity measure.

The goal of our experiments is twofold. First, we want to show that the use of background knowledge allows producing semantically richer correspondences between two matching ontologies than using only a linguistic-syntactic approach. Also, we want to investigate if we can obtain a higher precision/recall [Rijsbergen, 1979] considering our process which is a combination of some existing approach (linguistic-syntactic) with our semantic one. Consider that R is a reference alignment and A is an alignment produced by an ontology matching tool. In our work, precision and recall are defined as the ratio of the number of true positive ($|R \cap A|$) and retrieved correspondences ($|A|$) or those to be retrieved ($|R|$), respectively.



Ontology 1	Correspondence	Ontology 2	Measure
AssistantProfessor	isDisjointWith	AssociateProfessor	0
Person	isDisjointWith	Work	0
Worker	isDisjointWith	UndergraduateStud...	0
FullProfessor	isDisjointWith	AssociateProfessor	0
Organization	isDisjointWith	Person	0
Topic	isDisjointWith	Person	0
Event	isDisjointWith	Person	0
Publication	isDisjointWith	Person	0
GraduateStudent	isDisjointWith	UndergraduateStud...	0
TechnicalStaff	isDisjointWith	AdministrativeStaff	0
Person	isDisjointWith	Publication	0
Product	isDisjointWith	Person	0
AdministrativeStaff	isCloseTo	Assistant	0,7
DevelopmentProject	isCloseTo	ResearchProject	0,7
TechnicalReport	isCloseTo	JournalArticle	0,7
ClericalStaff	isCloseTo	Dean	0,7
ClericalStaff	isCloseTo	Director	0,7
FullProfessor	isCloseTo	VisitingProfessor	0,7
TechnicalReport	isCloseTo	ConferencePaper	0,7
SystemsStaff	isCloseTo	Chair	0,7
Lecturer	isCloseTo	Professor	0,7
Faculty	isCloseTo	Assistant	0,7
SystemsStaff	isCloseTo	Director	0,7

[✓ Average] DICE

Figure 5.7. The semantic matching tool interface.

Our experiments were conducted considering the ontologies introduced in Section 5.2. To verify the first mentioned goal, we matched O_i against O_j using only the H-Match and then using the DO as background knowledge. As a result, regarding the O_i concept *FullProfessor*, H-Match has produced 41 correspondences (1:n) to target O_j concepts. Most of them have been stated with similarity measures around 0.5, what is meaningless in terms of trying to identify the semantics underlying them. Only three of them have been stated with measures higher than 0.8, meaning some kind of subsumption or equivalence, although this result is not made explicitly. Still regarding *FullProfessor*, our semantic matcher produced five semantic correspondences.

Table 5.2 depicts some of the correspondences identified by H-Match and all the ones found out by the proposed semantic matcher.

Table 5.2. O_j target concepts obtained for the O_i concept *FullProfessor*.

H-Match		Semantic Matcher	
Target Concept	**Measure**	**Target Concept**	**Correspondence Type**
FullProfessor	1.00	FullProfessor	isEquivalentTo (1.00)
VisitingProfessor	0.82	VisitingProfessor	isCloseTo (0.70)
Student	0.50	Professor	isSubConceptOf (0.80)
Faculty	0.50	AssociateProfessor	isDisjointWith (0.0)
Manual	0.50	ResearchProject	isPartOf (0.30)

In order to check the second mentioned goal, we invited two expert users, which are knowledgeable about the *Education* domain, to produce a manual alignment between O_i and O_j. This "gold standard" alignment was used against our produced alignments. We used the ontology matching tools COMA++, H-Match, and Falcon-AO, discussed in Chapter 2, to match the ontologies *Semiport.owl* and *UnivBench.owl*. Afterwards, we calculated recall and precision to determine the agreement between the resulting alignments and the manual alignment. Next, we combined the (linguistic-structural) alignments produced by each matching tool with the semantic alignments produced by our matching tool. Again, recall and precision were used to measure the agreement between the resulting alignments and the manual alignment.

For both comparisons, we used the filter strategy which consists in selecting, for each concept of O_i, the correspondence with the highest combined similarity. The comparison results are illustrated in Figure 5.8.

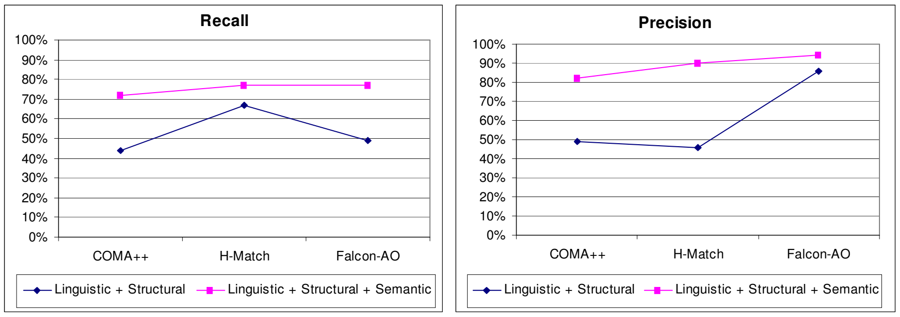


Figure 5.8. Evaluation of resulting alignments.

According to the results, we can see that when the semantic matcher is applied both measures (recall and precision) are increased. The reason for such improvement is that incorrect correspondences are removed from the resulting alignments while missing but still relevant correspondences are introduced. For instance, none of the three ontology matching tools (COMA++, H-Match, and Falcon-AO) was capable of identifying a candidate element in O_i for the element *AssociateProfessor* of O_j. Differently, *SemMatch* has identified that *AssociateProfessor* has a certain degree of similarity with *ResearchProject* in O_i. Our tool has identified that both elements are related by an *isPartOf* semantic relationship.

In addition, H-Match and COMA++ has incorrectly identified that the elements *worker* (O_i) and *work* (O_j) are equivalent. In fact, these tools make strong use of linguistic matchers and, since the strings describing both elements are similar to each other, a correspondence between the elements has been detected. During our tests with *SemMatch*, the semantic matcher did not detect such correspondence. Since we have given more importance to the semantic matcher than to the hybrid matcher (matcher weights: *hybrid* = 0.4; *semantic* = 0.6), the correspondence was not well evaluated and, consequently, has not been included in the resulting alignment.

5.7 Related Work

Most of the work dealing with ontology similarity (or distance) [Mädche and Staab, 2002; Hu *et al.*, 2006] is in reality concerned with concept similarities (or distances). However, there are many situations where it is useful to know if two ontologies are close to each other or not, or what is the closest ontology to another one. For instance, in semantic P2P systems, it would be easier to find information if queries can be sent to peers using similar ontologies because query transformation will miss less information [Ehrig *et al.*, 2005].

The work of Castano and her group [Castano *et al.*, 1998] proposes a kind of global similarity measure, but concerned with ER schemas. COMA++ [Aumüller *et al.*, 2005] is a tool which argues that calculates a global measure between schemas, but, considering the version we had performed our tests, we were not able to find out such feature explicitly. Recently, [David and Euzenat, 2008] have presented a review of several concept and ontology distance measures as well as an evaluation of their qualities. Basically, the authors have analyzed the speed of distance computation and the accuracy with regard to asserted similarity. Our work provides a global similarity measure as an additional feature in a semantic-based ontology matching process. The global measure produced by the matching process is used for clustering semantically similar peers in a PDMS.

5.8 Considerations

This chapter presented an ontology matching process which tries to overcome the limitations of traditional approaches by using a combination of linguistic, structural, and semantic matchers. Particularly, the semantic matcher is capable of identifying, besides the traditional types of correspondences (equivalence and subsumption), some other ones (e.g. closeness and disjointness). Furthermore, as a result of the overall process, we introduced the determination of a global similarity measure between the matching ontologies which is calculated considering the identified similarity value of each correspondence. Such measure is used for clustering semantically similar peers in SPEED.

Experiments carried out has shown that the combination of the proposed semantic matcher with linguistic-structural matchers can improve the alignments produced by existing ontology matching tools, by taking out incorrect or meaningless correspondences and including some relevant ones. These additional correspondences are useful for query answering and for the determination of the global measure.

CHAPTER 6

ONTOLOGY SUMMARIZATION

"Mestre não é quem sempre ensina, mas quem de repente aprende"
Guimarães Rosa

In this chapter, we propose an automatic process to summarize ontologies representing an individual schema or multiple schemas. An ontology summary is defined as a subontology of the initial ontology under a specific size. Particularly, in SPEED, the process is used to summarize cluster ontologies. The structure of the chapter is described as follows. First, we present an overview of the proposed summarization process and our formalism to represent an ontology. Next, we describe *centrality* [Freeman, 1979; Mika, 2007], the main criterion used to determine the relevance of concepts in an ontology. It is measured considering the relationships of a concept with other ones in an ontology. Particularly, if an ontology represents multiple schemas, as in SPEED, then *frequency* (i.e. the number of occurrences of a concept in local ontologies) is also used as another criterion to measure the relevance of concepts. Afterwards, we describe the proposed process to summarize ontologies, the summarization algorithm, and an illustrative example. We also expose the results of applying the proposed process to real world ontologies according to different criteria and discuss some related work.

6.1 Introduction

In the peer clustering process proposed for SPEED (Chapter 7), cluster ontologies are used by requesting peers to identify other semantically similar peers and, consequently, join a cluster. Such identification is assisted by an ontology matching process between a requesting peer's local ontology and (a subset of) the current cluster ontologies. As more requesting peers join a cluster, new elements are introduced in the corresponding cluster ontology which can reach a size that overburdens development and quality control procedures. In this sense, schema summarization techniques [Castano *et al.*, 1998; Moody and Filtman, 1999; Yu and Jagadish, 2006] can be used to produce succinct versions of cluster ontologies. These summarized ontologies can be of great help since not all elements shared inside a cluster need to be considered during the ontology matching process used for peer clustering.

An ontology summary provides a succinct overview of the entire ontology, making it possible to explore only the relevant elements [Zhang *et al.*, 2007]. However, creating a good summary is a non-trivial task. Ideally, the summary should be concise enough for requesting peers to comprehend the initial ontology quickly, yet it needs to convey enough information for requesting peers to obtain a decent understanding of the whole ontology. Manual ontology summarization is labor-intensive and impractical especially in situations where a high number of ontologies need to be summarized as in PDMS. In addition, leaving summary generation to a manual process let open the possibility that the summary will not be updated when the cluster ontology evolves, resulting in a supposed summary that is actually outdated and misleading [Yu and Jagadish, 2006]. In a PDMS context, the need for automatic tools to summarize cluster ontologies is mainly due to scalability and consistency reasons.

6.2 General Overview

As illustrated in Figure 6.1, the proposed summarization process consists in, given an ontology *O*, generating an abridged version of *O*, named ontology summary (denoted *OS*) [Pires *et al.*, 2009a]. The relevant concepts of *O* (depicted in grey) are initially identified and *OS* corresponds to the subontology of *O* concentrating the maximum number of relevant concepts. Since relevant concepts can be non-adjacent in *O*, non-relevant concepts (white color) may be also introduced in an *OS*. Such "undesired" concepts are needed to maintain the original relationships among relevant concepts. If the relevant concepts are simply identified and added to an ontology summary (ignoring their relationships), then a human intervention would be necessary to (re)link them. Therefore, *OS* also corresponds to the subontology of *O* containing the minimum number of non-relevant concepts.

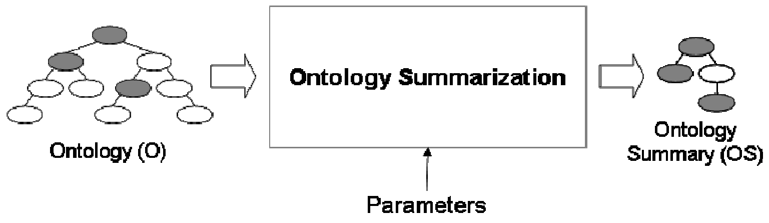


Figure 6.1. An overview of the proposed ontology summarization process.

Although we focus on ontological schemas, the proposed summarization process can be adapted to other kinds of schemas (e.g. XML and relational), considering that the schema can be mapped onto a graph representation. Therefore, most of the principles presented here are applicable to a wide variety of schemas.

6.3 Ontology Formalism

According to OWL syntax, an ontology can contain different constructs such as classes (i.e. concepts), properties, instances, and axioms of atomic class (property). In this work, we assume that terms at conceptual level (concepts and properties) are enough to provide an understandable ontology summary. OWL constructs such as ontology header and instances are ignored during ontology summarization.

The proposed graph formalism to represent an OWL ontology enables us to focus only on the OWL constructs which are important to the summarization process. In this light, an **ontology** *O* is modeled as a connected directed labeled graph $O = (C, R)$, where $C = \{c_1,...,c_n\}$ is a finite set of vertices (concepts) and $R = \{r_1,...,r_n\}$ is a finite set of edges (relationships between concepts).

A relationship $r_k \in R$ represents a directed relation between two adjacent concepts c_i and $c_j \in C$; i.e. $r_k = (c_i \times c_j)$. Two concepts c_i, $c_j \in C$ are **adjacent** in O if $\exists\ r_k \in R$ / $r_k = (c_i \times c_j)$ or $r_k = (c_j \times c_i)$. A directed labeled edge is defined from c_i to c_j if c_i is a direct subconcept of c_j. Similarly, if c_i is a domain concept and c_j its range concept then a directed labeled edge is added from c_i to c_j. The number of concepts in C indicates the **size of an ontology** O, denoted $|O|$. Particularly, we assume that in O there are no self-references or multiple edges between two distinct concepts. Edges from a concept vertice to its datatype property vertices and from each subproperty to its superproperty are ignored. Similarly, we define an **ontology summary** OS as a proper subgraph of O since $OS \subset O$ (or O is a supergraph of OS). Notice that $OS \neq O$, otherwise OS is not a summary of O. Since OS is a subgraph of O, the same formalism is valid for OS. Formally, $OS = (CS, RS)$, where $CS \subset C$ and $RS \subset R$.

6.4 Relevance Measures

The relevance of an ontology concept c_n is measured considering the relationships of c_n with other concepts in an ontology O (*centrality*) and the occurrences of c_n in local ontologies $O_1,\ldots,O_n$ that compose O (*frequency*). In our approach, centrality is used to capture the importance of a given concept within an ontology, while frequency is used when an ontology results from an integration process and captures the importance of this concept in the set of underlying local ontologies. In the following, we detail these two relevance measures.

6.4.1 The Centrality Measure

Centrality [Freeman, 1979] is one of the most important and widely used ways for identifying relevant vertices within a graph. The notion of relevance is subjective since it depends on what is considered important for a vertice. In [Freeman, 1979], the authors categorized centrality measures into three basic categories (degree, closeness, and betweenness) and presented canonical measures for each category. As a result, these measures have come to dominate empirical usage, along with the eigenvector-based measure [Bonacich, 1972].

The *degree centrality* [Mika, 2007] is based on the idea that a vertice v with a large number of links to other vertices has wider and more efficient access to the other vertices in the graph. The *eigenvector centrality* [Zhang *et al.*, 2007] acknowledges that the centrality of a vertice v does not only depend on the number of its links to other vertices, but also on their value of centrality.

The other two centrality measures are based on the notion of graph paths [Diestel, 2005]. A path in a graph is a sequence of consecutive edges. A geodesic path is the shortest path, in terms of number of edges traversed, between two vertices. The *closeness centrality* [Mika, 2007] of a vertice *v* means the geodesic distance between *v* and all its reachable vertices. The *betweenness centrality* [Mika, 2007] of a vertice *v* is the number of geodesic paths between other vertices that *v* falls on.

In this work, we extend the original definition of the degree centrality measure not only to consider the number of relationships between ontology concepts but also the types of relationships between them. In this light, two types of relationships are identified: standard (e.g. *is-a*, *part-of*, and *same-as*) and user-defined (e.g. *hasItems* and *authorOf*). The normalized formula for the extended degree centrality is:

$$centrality(c_n) = \frac{nr \times \left(\frac{n_s \times w_s}{max_s} + \frac{n_{ud} \times w_{ud}}{max_{ud}} \right)}{|C| - 1}$$

where n_s and n_{ud} are respectively the number of standard and user-defined relationships maintained by an ontology concept c_n. w_s and w_{ud} are respectively the weights of the standard and user-defined relationships. max_s and max_{ud} indicate respectively the maximum number of standard and user-defined relationships maintained by a particular concept in *O*. *nr* represents the number of distinct concepts with which a concept c_n maintains relationships. In addition, (i) centrality(c_n) $\in$ [0,1]; (ii) $w_s + w_{ud} = 1$; and (iii) $n_s + n_{ud} = n_r$.

6.4.2 The Frequency Measure

Frequency is a measure that can be used when the ontology to be summarized is a cluster ontology obtained as a result of merging several local ontologies $LO_1,\ldots,LO_n$. Ontology merging [Noy and Musen, 2000] is the process in which two (or more) local ontologies are merged into one target ontology. In general, the local ontologies remain, along with ontology mappings between each local ontology and the merged ontology. Different types of ontology mappings can be defined between a target ontology and local ontologies, e.g. concept mappings and property mappings. Figure 6.2 represents an excerpt from an XML file describing concept mappings in a cluster of peers of SPEED [Pires, 2007b]. For instance, the concept *faculty* contained in the target ontology CLO_1 is mapped to the concepts *phd*, *professor*, and *lecturer* located at the local ontologies LO_1, LO_2, and LO_3, respectively.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TARGET clo="CLO1">
<TARGETCLASS>
<LABEL>faculty</LABEL>
     <LOCALCLASS>
        <LABEL>phd</LABEL>
        <LOCAL>L01</LOCAL>
     </LOCALCLASS>
     <LOCALCLASS>
        <LABEL>professor</LABEL>
        <LOCAL>L02</LOCAL>
     </LOCALCLASS>
     <LOCALCLASS>
        <LABEL>lecturer</LABEL>
        <LOCAL>L03</LOCAL>
     </LOCALCLASS>
</TARGETCLASS>
...
</TARGET>
```

Figure 6.2. An example of concept mappings.

In the proposed summarization process, we assume that O can be a merged ontology. Thus, a concept $c_n \in C$ can be mapped to one or more concepts contained in $O_1,\dots,O_n$. In this sense, the frequency of c_n is defined as the ratio between the number of concept mappings involving c_n (denoted $|\text{mappings}(c_n)|$) and the number of distinct local ontologies (denoted $|O_1,\dots,O_n|$). Both information can be extracted from the ontology mappings. Formally,

$$frequency(c_n) = \frac{|mappings(c_n)|}{|O_1,\dots,O_n|}$$

where $\text{frequency}(c_n) \in [0,1]$. For instance, given the concept mappings illustrated in Figure 6.2, the concept *faculty* is involved in three concept mappings. Assuming that the number of distinct local ontologies is also three then frequency(*faculty*) = 1.0.

6.5 Building an Ontology Summary

In this section, we offer a detailed description of the proposed ontology summarization process including its several steps and input parameters. Then, we present the summarization algorithm and illustrate the process with an example.

6.5.1 The Summarization Process

The main steps of the ontology summarization process are: (i) calculate the relevance of ontology concepts; (ii) determine the relevant concepts; (iii) group adjacent relevant concepts; (iv) identify paths between groups of concepts; (v) analyze the identified paths; and (vi) determine the ontology summary. Figure 6.3 corresponds to an UML activity diagram [Booch *et al.*, 2005] depicting all the steps involved in the summarization process. Next, we provide a detailed description of each step.

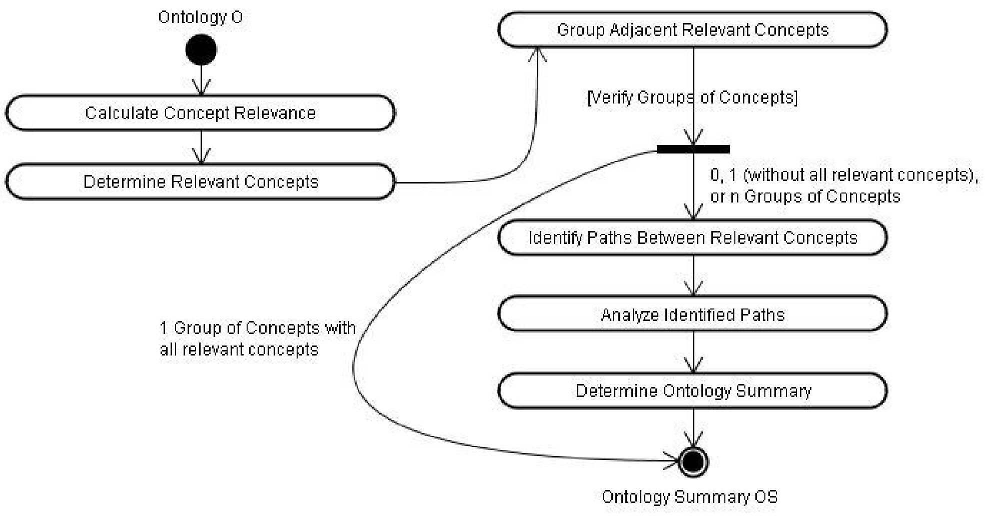

Ontology O
Calculate Concept Relevance
Determine Relevant Concepts
Group Adjacent Relevant Concepts
[Verify Groups of Concepts]
0, 1 (without all relevant concepts), or n Groups of Concepts
Identify Paths Between Relevant Concepts
Analyze Identified Paths
Determine Ontology Summary
1 Group of Concepts with all relevant concepts
Ontology Summary OS

Figure 6.3. The several steps of the proposed summarization process.

Step 1: Calculate relevance of concept

Centrality and frequency are two criteria inherently different. It is certainly possible to find two distinct ontology summaries where one has more centrality but the other has better frequency. In fact, the most relevant measure depends on the application needs. However, in some occasions both measures need to be considered together. Our proposal to combine centrality and frequency consists in using a weighted formula in which the weights are defined by a user according to the importance of each measure to the application purposes (λ: centrality weight; β: frequency weight). This formula is used to calculate the relevance of an individual concept c_n in an ontology O:

$$relevance(c_n) = \lambda \times centrality(c_n) + \beta \times frequency(c_n)$$

where relevance(c_n) $\in$ [0,1] and $\lambda + \beta = 1$.

Step 2: Determine relevant concepts

This step consists in identifying the *set of relevant concepts* (denoted *RC*, where $RC \subseteq C$) of an ontology *O*. Ideally, the concepts in the identified set should be contained in the ontology summary *OS*. Several options can be used to determine *RC*. The first one considers that *RC* has a fixed size which is determined by the user (*suggested size*). In this light, all concepts are firstly classified in descending order according to their respective relevance. Afterwards, the top *k* concepts are selected, where *k* corresponds to *suggested size*. In general, *suggested size* is defined as a percentage of the ontology size ($|C|$).

The second option assumes that *RC* can have a variable size. The concepts to be included in *RC* are those concepts whose relevance is above a *relevance threshold* which is also informed by the user. Formally,

$$\forall c_n \in C,\ if\ relevance(c_n) \geq relevance\ \ threshold\ \Rightarrow c_n \in RC$$

Finally, the third option also enables *RC* to assume a variable size. However, it determines *RC* automatically. In order to select the concepts to be included in *RC*, the *average relevance* (*AR*) of all individual concepts in *C* is calculated:

$$AR(C) = \sum_{i=1}^{n} \frac{relevance(c_i)}{|C|}$$

The most relevant concepts are those concepts in which the individual relevance is higher than or equal to the average relevance. Formally,

$$\forall c_n \in C,\ if\ relevance(c_n) \geq AR(C) \Rightarrow c_n \in RC$$

The last two options to determine *RC* should be carefully used. For instance, if *relevance threshold* is configured to a small value or if each concept maintains relationships with the majority of concepts in *O*, a high number of relevant concepts can be included in *RC*.

Step 3: Group adjacent relevant concepts

This step consists in forming *groups of concepts* containing only relevant concepts which are adjacent in the initial ontology *O*. A group of concepts corresponds to a subontology of *O*. Such groups are created in order to facilitate the identification of paths between relevant concepts (Step 4).

When building groups of concepts the following situations can occur: (i) each group is formed by a single relevant concept: this means that all relevant concepts are non-adjacent in *O*; (ii) several groups are formed, and at least one of the groups has more than one relevant concept: this means that some relevant concepts are not adjacent in *O*; (iii) only one group is formed, containing all the relevant concepts: this means that each relevant concept has at least one distinct relevant concept that is adjacent in *O*. In the first two situations, the ontology summarization process proceeds with Steps 4, 5, and 6. Differently, in the last situation, the summarization process finishes and the ontology summary corresponds to the group of concepts that is formed.

Step 4: Identify paths between groups of concepts

This step is executed if there are at least two groups of concepts which are not adjacent in the initial ontology *O* (situations i and ii of Step 3). It consists in detecting all paths (denoted OS_n) between groups of concepts in *O*. For such task each group of concepts is treated as a single concept. Similarly to a group of concepts, a path OS_n also corresponds to a subontology in *O*. Multiple paths between two groups of concepts can be detected. In order to minimize computation efforts, only paths with the requested summary size (i.e. $|RC|$) should be considered. However, notice that this step is executed if at least two relevant concepts are separated by a non-relevant concept. Thus, none of the identified paths can contain the entire set of relevant concepts *RC*.

Groups of concepts can be separated by both a large or a few number of non-relevant concepts in *O*. In the first case, discarding some relevant concepts (e.g. a relevant concept that is far from a group of concepts) would allow the identification of a path containing only relevant concepts. In the second case, introducing only some non-relevant concepts to a path (e.g. adding a non-relevant concept which separates two groups of concepts) would allow the identification of a path including all relevant concepts.

Both cases can only be satisfied if the requested size for an ontology summary (initially defined as $|RC|$) can be relaxed, i.e. if summary size can vary in a specific interval. Such interval is determined by a *size variation* denoted by

Δ. For example, assuming a summary size and a size variation of 6 and 2 respectively, then all paths whose size is between 4 and 8 are considered as candidate summaries. Formally,

$$summary\ \ size - \Delta \leq |OS_n| \leq summary\ \ size + \Delta$$

Step 5: Analyze identified paths

Since multiple paths between relevant concepts (and groups of concepts) can be identified, it is necessary to analyze each path individually. The classical metrics *recall* and *precision*, commonly used in Information Retrieval [Baeza-Yates and Ribeiro-Neto, 1999], are used to determine the level of coverage and conciseness of each path *OS_n*, respectively. Recall means that a path should be an extraction of *O* reflecting as many relevant concepts as possible. Precision determines if a path is succinct enough to facilitate the analysis of the entire ontology *O*. Formally,

$$Recall = \frac{|OS_n \cap RC|}{|RC|} \qquad Precision = \frac{|OS_n \cap RC|}{|OS_n|}$$

Paths cannot be compared based solely on precision and recall. The path which has high recall may have a low precision and vice-versa. For this purpose, F-measure [Baeza-Yates and Ribeiro-Neto, 1999] is used to aggregate precision and recall.

$$F - measure = \frac{Precision \times Recall}{(1-\alpha) \times Precision + \alpha \times Recall}$$

Notice that, if $\alpha = 1$, then the F-measure is equal to precision and if $\alpha = 0$, the F-measure is equal to recall. In between, the higher the value of α, the more importance is given to precision.

Step 6: Determine the ontology summary

Among the identified candidate paths (OS_n), the selection of an ontology summary is determined by choosing the best candidate path. In this sense, the selection occurs according to the following priority order:

(i) *F-measure*: the path should be the one having the maximum number of relevant concepts and the minimum number of non-relevant concepts. In other words, the path with the highest value of F-measure should be selected;

(ii) *Average relevance*: since it is possible to find two distinct paths with the same value of F-measure, a second criterion is needed. In this case, the path with highest average relevance should be chosen. The average relevance of a path corresponds to the ratio between the sum of the individual concept relevance in a path and the number of concepts in a path;

If multiple candidate paths have identical F-measure and average relevance values, other strategies can be used to select a path: choose randomly one of the paths or select the one leading to the summary with the smallest size (considering that $\Delta > 0$).

6.5.2 The Ontology Summarization Algorithm

Figure 6.4 illustrates the proposed summarization algorithm. It accepts as input the ontology to be summarized (mandatory), a set of ontology mappings (optional), and a set of parameter values supplied by the user (mandatory), e.g. suggested size. If ontology mappings are not informed, only the centrality measure is used to calculate the relevance of concepts. An ontology summary is generated as output. In order to meet diverse user requirements, the algorithm can accept various types of parameters. Depending on the parameter values provided, different ontology summaries *OS* can be generated for the same ontology *O*.

```
SummarizeOntology (in: Ontology; in: Mappings; in: Parameters; out: Ontology)
{
CalculateConceptRelevance(Ontology, λ, β, centrality measure);
RC ← DetermineRelevantConcepts(Ontology, relevance criteria);
Ontology ← GroupAdjacentRelevantConcepts(Ontology);
If Ontology.Groups = 1 and RC ⊆ Ontology.Groups[1].Concepts then
 OntologySummary ← Ontology.Groups[1];
Else
 Paths ← IdentifyPaths(Ontology, Δ);
 AnalyzePaths(Paths, α);
 OntologySummary ← GetBestPath(Paths);
End if
Return(OntologySummary);
}
```

Figure 6.4. The ontology summarization algorithm.

6.5.3 An Example

Our example considers a public ontology[10] describing nodes in a local area network (Figure 6.5). Assume that an ontology summary containing 6 concepts (i.e., *suggested size* = 6) with size variation of 1 ($\Delta = 1$) must be generated. To simplify matters, only centrality is used to determine the relevance of concepts. Moreover, assume that recall and precision have the same importance (α =0.5). In this sense, *RC* = {*ServerSoftware* (0.231), *NetworkNode* (0.192), *SwitchEquipment* (0.192), *Computer* (0.192), *Software* (0.192), *Cable* (0.192)}. The first five concepts are adjacent in the *NetworkA* ontology. Thus, they are combined into the group of concepts $Group_1$. The other group of concepts ($Group_2$) is composed solely by *Cable*. Since more than one group of concepts has been identified, the summarization process proceeds. All paths between $Group_1$ and $Group_2$ are identified. There are only two paths whose size is in the interval defined by Δ. The first path ($Path_1$) is: $Group_1 \rightarrow Equipment \rightarrow Group_2$. The second path ($Path_2$) is: $Group_1 \rightarrow NodePair \rightarrow Group_2$. The value of F-measure is identical for both paths (92.5). However, the average relevance of $Path_1$ (0.187) is higher than the average relevance of $Path_2$ (0.181). As a result, $Path_1$ is chosen as the ontology summary. The summary is shown in the left lower part of Figure 6.5.

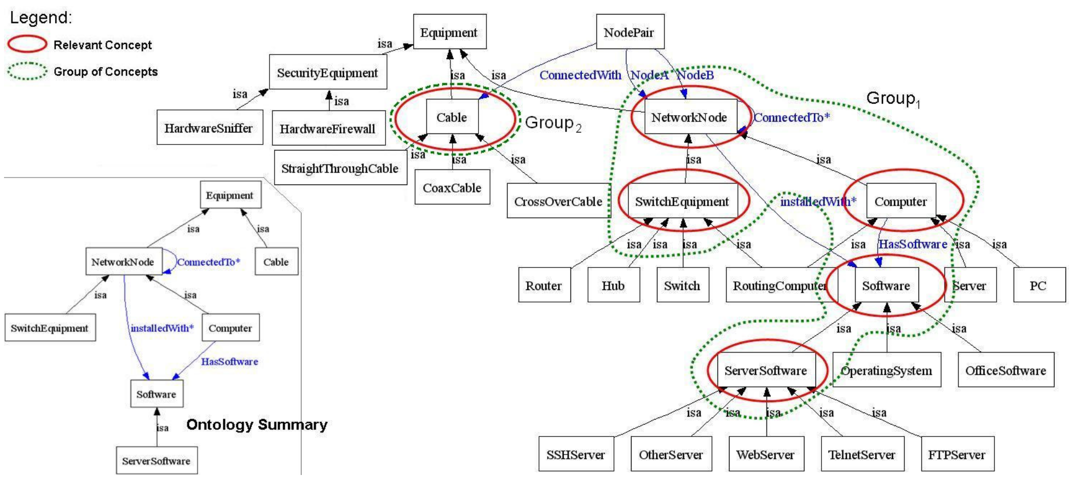


Figure 6.5. The networkA ontology and its corresponding summary.

[10] http://www.atl.lmco.com/projects/ontology/ontologies/network/networkA.owl

6.6 Generating Ontology Summaries

In this section, we present an evaluation of the proposed ontology summarization process. Basically, we asked expert users, which are knowledgeable about specific ontologies, to produce manual summaries. This created a "gold standard" set of summaries against which our automatic summaries can be compared and analyzed.

6.6.1 Implementation

We have developed an ontology summarization tool to produce automatic summaries of OWL ontologies. The tool is implemented in Java [Java, 2009] and uses the OWL API [OWL API, 2009] to manipulate ontologies. A first version of the summarization tool is available for download[11]. The tool can be invoked at command line and accepts a set of input parameters, e.g. *owl_filename* and *owl_summarized_filename*. The parameters can be initialized at the *summary.properties* file. After each successful execution, a log file is generated. The log file contains processing information produced by the tool during an execution, e.g. elapsed time and identified candidate paths. Such information can be useful for users in order to adjust the initialization parameters.

6.6.2 Case Study

We have selected four OWL ontologies belonging to distinct knowledge domains as test cases: a conference ontology[12], a network ontology (Section 6.5), an office ontology[13], and an university ontology[14]. The conference ontology is an ontology draft for events and, specifically, conferences. The office ontology models an office environment. The university ontology describes a computer science academic department. All of them are public ontologies which are available for download. The four ontologies are selected as test case since they are rather small and can be reviewed by human to produce "gold standards". Some statistical data of the chosen ontologies are illustrated in Table 6.1, including the number of concepts and properties.

[11] The Ontology Summarizer site, http://www.cin.ufpe.br/~speed/OWLSummarizer/
[12] http://ebiquity.umbc.edu/ontology/conference.owl
[13] http://ise.icu.ac.kr/Ontologies/office-env2.owl
[14] http://www.cs.toronto.edu/semanticweb/maponto/MapontoExamples/univ-cs.owl

Table 6.1. Ontology statistics.

	conference.owl	networkA.owl	office-env2.owl	univ-cs.owl
# Concepts	18	27	35	53
# Properties	18	06	08	25

6.6.3 Comparison with Expert Summaries

We invited three expert users to generate "gold standard" summaries for the conference, network, office, and university ontologies. Summaries of different sizes were requested: 4, 8, and 12 concepts. No size variation was allowed. Experts were oriented to produce summaries containing only adjacent concepts. Correspondingly, we generated automatic summaries at the same sizes and measured the agreement between the automatic summaries and the expert summaries. Since frequency is not an intrinsic characteristic of ontologies, in order to be comparable with expert summaries only centrality was considered when generating the automatic summaries. The agreement between two ontology summaries is defined as the percentage of the number of concepts selected by both the expert users and the summarization tool over the requested summary size. An agreement summary of a particular summary size is generated by combining all expert summaries and retaining only the concepts selected by a majority of the experts (in this case, at least two experts). We have also compared the expert summaries against the summaries produced by OntoSum[15], a demo tool for summarizing small ontologies in real-time on the Web. Particularly, for OntoSum, we have used the Weighted PageRank measure since in [Zhang *et al.*, 2007] the authors affirm that have obtained the best evaluation for ontology summaries.

Table 6.2 illustrates the results of our experiments. Summaries containing 12 concepts were not generated for the conference and network ontologies. In our opinion, this size is too high to represent a summary for the two ontologies. Except for the office ontology, our system was in reasonable consonance with human experts. The results for the office ontology are explained because a high relevant concept was positioned far from the other relevant concepts. Consequently, candidate summaries containing this concept were very well

[15] http://iws.seu.edu.cn/services/falcon-f/ontosum/

evaluated (F-measure), even with some non-relevant concepts. Obviously, such non-relevant concepts were not chosen by the expert users. Experts do not always agree on what is the best summary. In general, the percentage of agreement between expert summaries and automatic summaries increases as the summary size augments. Briefly, Table 6.2 shows that the automated summarization tool was able to produce summaries at different sizes that appear to be similar to what an expert may have produced.

Table 6.2. Comparison between the automatic summaries and the expert summaries.

conference.owl	**4-Concept**	**8-Concept**	**12-Concept**
Expert 1 against Automatic	75%	88%	-
Expert 2 against Automatic	50%	75%	-
Expert 3 against Automatic	75%	75%	-
User agreement against Automatic	75%	75%	-
User agreement against OntoSum	50%	50%	-
networkA.owl	**4-Concept**	**8-Concept**	**12-Concept**
Expert 1 against Automatic	50%	100%	-
Expert 2 against Automatic	50%	75%	-
Expert 3 against Automatic	50%	50%	-
User agreement against Automatic	50%	75%	-
User agreement against OntoSum	75%	75%	-
office-env2.owl	**4-Concept**	**8-Concept**	**12-Concept**
Expert 1 against Automatic	100%	75%	67%
Expert 2 against Automatic	75%	63%	58%
Expert 3 against Automatic	75%	63%	58%
User agreement against Automatic	100%	63%	58%
User agreement against OntoSum	50%	50%	75%
univ-cs.owl	**4-Concept**	**8-Concept**	**12-Concept**
Expert 1 against Automatic	-	75%	92%
Expert 2 against Automatic	-	88%	83%
Expert 3 against Automatic	-	50%	67%
User agreement against Automatic	-	75%	92%
User agreement against OntoSum	-	63%	75%

During the experiments with the summarization tool, we have observed some particular situations which are important to be stated: (i) as the summary size increases, the probability of forming only one group of concepts containing all relevant concepts is also increased. Consequently, the possibility of introducing non-relevant concepts in the summary decreases; (ii) at most one group of concepts was formed for the chosen ontologies; and (iii) in general, the use of a fixed summary size ($\Delta = 0$) does not allow the identification of the best summary. For a certain summary size, there were cases in which no summary was identified, e.g. a 4-Concept summary for the university ontology (Table 6.2). We believe that better summaries could have been obtained if a variable summary size had been used ($\Delta > 0$).

6.7 Related Work

The first studies on schema summarization have focused on entity-relationship (ER) model abstraction. In such model, since data is not available, only the structural characteristics of ER diagrams are exploited [Castano *et al.*, 1998; Moody and Filtman, 1999]. The authors of [Castano *et al.*, 1998] use clustering techniques to produce a summarized version of an ER diagram. They present an algorithm for performing schema clustering, and then discuss criteria for representing clusters by means of abstract elements and for abstracting links between elements. The technique rely heavily on the semantics embedded in the relationships to guide the process and is therefore not truly automated. The amount of human effort required is significant, especially to define links between abstract elements. In [Yu and Jagadish, 2006], a summarization process for relational and XML schemas is proposed. The authors demonstrate that while schema structure is of vital importance in summarization, data distribution often provides important insights that significantly improve the summary quality. One consequence of using data distributions is that a generated summary may evolve when a database is updated even though the schema stays the same.

We have also analyzed other summarization processes in which ontologies do not represent schemas. In [Zhang *et al.*, 2007], the authors propose a novel process to automatic ontology summarization based on RDF Sentence Graph. Summaries are customizable, i.e. users can specify the length of summaries and their navigational preferences. A notion of RDF sentence is proposed as the basic unit of summarization. An RDF Sentence Graph is proposed to characterize the links between RDF sentences derived from a given ontology. The salience of each RDF sentence is assessed in terms of its centrality in the graph. An ontology is summarized by extracting a set of salient RDF sentences according to a re-ranking strategy.

In [Stuckenschmidt and Klein, 2004], an automatic method for structure-based ontology partitioning is proposed. The method is applicable to large ontologies and consists in dividing an ontology into smaller and disjoint modules based solely on the structural properties of the ontology. Each module contains information about a coherent subtopic of the ontology and can be used independently of the other modules. Concepts inside each module are stronger related among them than with concepts outside the module. The output produced by the method is a connected graph where each node corresponds to a subtopic (or module) of the overall ontology. Although the set of modules can be considered as a summary for a given ontology, some important considerations must be made: (i) a module is not a concept; (ii) since the modules are not too close to each other in the graph, no information is provided to explicit the kind of relationship between them; thus, the result is considered a set of independent modules; (iii) the number of modules is arbitrarily predefined, as if the number of relevant concepts was; and (iv) the semantics of the relationships is not exploited in order to determine the level of dependency between concepts during the partitioning process. In [Schlicht and Stuckenschmidt, 2008], the authors present a tool for partitioning OWL ontologies that implement the described process.

Some notion of centrality is used to calculate the relevance of concepts in all the discussed works. However, none of them exploits the type of relationships between concepts. Although the works of [Yu and Jagadish, 2006; Zhang *et al.*, 2007] affirm that their summarization process is fully automatic, the size of summaries is still manually provided. In [Stuckenschmidt and Klein, 2004], the number and the size of modules also need to be informed. Moreover, using frequency as a criterion for determining relevant concepts to be included in a summary is not considered by any of the presented works. The main reason for that is because existing solutions do not consider merged ontologies in the summarization processes.

6.8 Considerations

In this chapter, we proposed an automatic process to summarize cluster ontologies representing multiple local schemas. To determine the relevance of concepts a combination of two measures was used. Centrality is calculated using an extended definition of the degree centrality measure. Frequency is used as a distinguishing criterion when the ontologies to be summarized are merged ontologies. A detailed description of the summarization process was presented as well as an algorithm for ontology summarization. Experiments have shown that the summarization process is able to find good summaries compared to the ones manually generated by expert users.

In the next chapter, we propose an ontology-based process for clustering peers in SPEED. To this end, the clustering process makes use of the ontology matching process described in Chapter 5 and the automatic process for summarizing ontologies described in this chapter.

CHAPTER 7

ONTOLOGY-BASED PEER CLUSTERING IN SPEED

"Words build bridges to unexplored regions"
Adolf Hitler

In this chapter, we describe an ontology-based process for clustering peers in SPEED. Although the proposed process aims to cluster peers in a PDMS, it can be applied to a data integration system or any other distributed system in which data sources communicate via some network protocol. In this sense, first an overview of the clustering process is introduced. Next, a demonstration of how a requesting peer searches for a corresponding semantic community in the DHT network is shown. Also, the main clustering characteristics and the algorithm for inserting a requesting peer into a semantically similar cluster are presented. The steps to connect a requesting peer to an existing cluster and to create a new cluster are detailed. Considerations about the maintenance of semantic clusters are also provided. Finally, experimental results are presented and discussed.

7.1 An Overview of SPEED's Clustering Process

In SPEED, the connection of a requesting peer is performed in a twofold way (Figure 7.1). First, a corresponding semantic community is searched in the DHT network. If the community is found, then a semantically similar cluster is searched in the unstructured network of the identified community. In both cases, the requesting peer's local ontology is of great importance since it is used to associate the requesting peer to an appropriate community as well as to a semantically similar cluster.

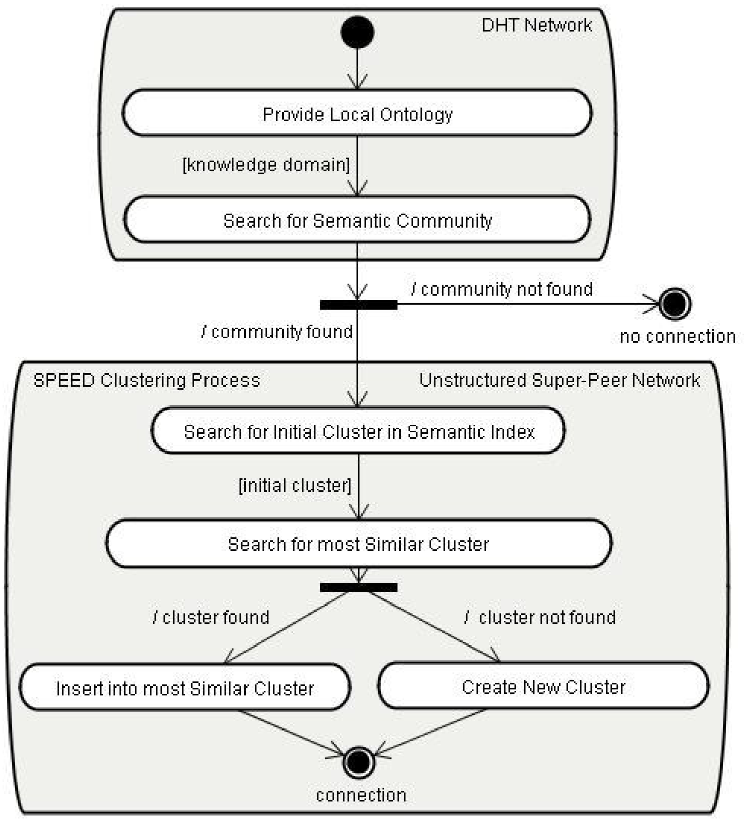


Figure 7.1. The several steps involved in the connection of a requesting peer.

Assuming that a community has been found by a requesting peer, the search for a semantically similar cluster begins when an initial cluster is provided to the requesting peer. Such initial cluster is obtained from the semantic index of the identified community. The search for a semantically similar cluster starts at the initial cluster and continues by visiting the semantic neighbors of the initial cluster located in the unstructured network.

At each visited cluster, the semantic similarity between the cluster and the requesting peer is computed. To this end, the ontology matching function *SemMatch* described in Chapter 6 is used. *SemMatch* takes as arguments two peer ontologies (i.e. a cluster ontology CLO_{ij} and a local ontology LO_n) and returns a global similarity measure which indicates the degree of similarity between the two ontologies.

Two peers are **semantically similar** if the global similarity measure between their ontologies is above a certain threshold, called **cluster threshold** (denoted *ct*). The integration peer of each visited cluster returns a global similarity measure to the requesting peer RP_n. If RP_n identifies a semantically similar cluster among the visited clusters, then RP_n joins that cluster. Otherwise, RP_n creates a new cluster. In both cases the requesting peer RP_n is connected to the community.

Once a requesting peer RP_n is connected, it can assume different roles in the system. For instance, if RP_n joins an existing cluster, it is connected as a data peer. Otherwise, if RP_n creates a new cluster, then it is connected as an integration peer. Figure 7.2 is an UML statechart diagram illustrating the two possible states of a requesting peer.

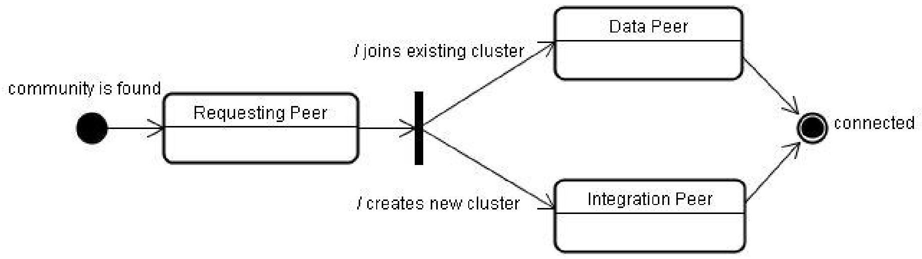


Figure 7.2. The possible states of a requesting peer.

In the following sections, we detail each step of the proposed ontology-based process for clustering peers in SPEED. We begin describing how a semantic community is searched in the DHT network.

7.2 Search for a Semantic Community

SPEED's DHT network is composed of multiple semantic peers. Each semantic peer represents a particular community and therefore a distinct knowledge domain. A semantic community is described by a set of keywords associated to its corresponding knowledge domain. For instance, the set of keywords related to the *education* knowledge domain can be *education*, *university*, and *professor*. Such keywords are defined by a system administrator and are used to locate a semantic community in the DHT network.

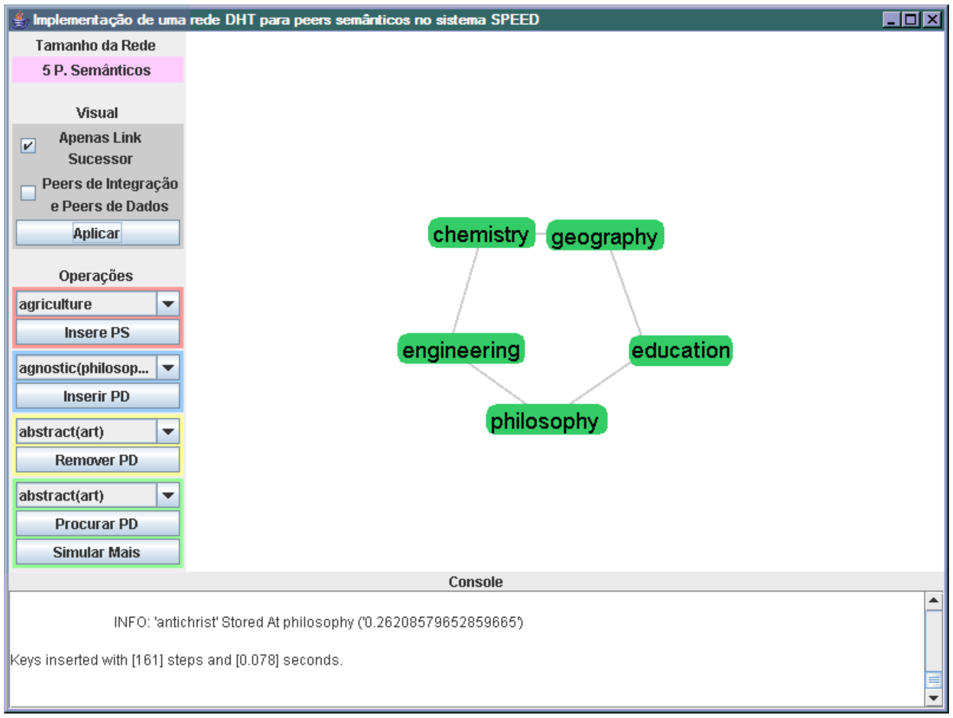
A semantic community is created when a group of peers wish to share data about a specific knowledge domain that is not available in the DHT network yet. For instance, different universities are interested in sharing data about their research projects as well as in producing more complete results that involve data located at other universities. In this case, the creation of an *education* community would fulfill such need.

SPEED's DHT network can be built according to any structured P2P protocol, e.g. Chord [Stoica *et al.*, 2001]. Particularly, if Chord is chosen then consistent hashing [Karger *et al.*, 1997] is used to map semantic peers and their corresponding keywords onto an identifier circle. An identifier is associated to each semantic peer and keyword. Each semantic peer implements a *successor* function. The *successor* of a semantic peer (or keyword) is the next semantic peer in the identifier circle.

Each semantic peer maintains a routing table called *finger table*. This table stores information about some other semantic peers in the DHT network. Basically, each entry in the finger table contains a semantic peer identifier and its network address. When a new semantic peer joins the network it must initialize its finger table. As a consequence, existing semantic peers are notified about this event and must also update their finger table to reflect the existence of the new semantic peer.

The management of semantic communities is not the main focus of the proposed clustering process. In fact, we are interested in discovering an appropriate community to insert a requesting peer. Therefore, in this thesis we assume the existence of a predefined DHT network representing multiple semantic communities. Figure 7.3 illustrates a Chord network formed by five semantic peers representing the following knowledge domains: *chemistry*, *geography*, *education*, *philosophy*, and *engineering*. The DHT network was generated by the SPEED's simulator. The simulator was used during our experiments and is better described in Section 7.6 and in the Appendix C.

In order to search for a semantic community, a requesting peer RP_n must first provide an *interest theme*, i.e. an abstract description of the requesting peer's knowledge domain. An interest theme corresponds to a keyword that can either be extracted automatically from the requesting peer's local ontology or manually informed by the user. *Education*, *health*, and *bioinformatics* are examples of interest themes.



```
INFO: 'antichrist' Stored At philosophy ('0.26208579652859665')
Keys inserted with [161] steps and [0.078] seconds.
```

Figure 7.3. An instantiation of the SPEED's DHT network.

Consistent hashing is also used to map an interest theme onto an identifier *i* which is sent to an arbitrary semantic peer in the DHT network. The search for a semantic peer is done progressively: at each step the successor of a semantic peer is identified until the closest semantic peer is found. As seen in the UML sequence diagram of Figure 7.4, a search for the successor of an identifier *i* (interest theme) initiated at the semantic peer SP_1, begins by determining if *i* is between SP_1 and the immediate successor of SP_1 (i.e. SP_2). If so, the search terminates and the successor of SP_1 is returned. Otherwise, SP_1 forwards the search request to the latest semantic peer (in its finger table) that precedes *i* (in Figure 7.4, SP_i). The procedure is repeated by SP_i until the search terminates.

According to such search strategy, if the provided interest theme *i* is contained in the set of keywords of a semantic community then the corresponding semantic peer will necessarily be found. In this case, a message containing the corresponding semantic peer's address is sent back to the requesting peer (Figure 7.4). If the interest theme is not found, it probably means that the referred community does not exist in the DHT network or the provided interest theme is not used to describe any semantic community. In this case, the requesting peer should try another interest theme. In SPEED's current version, a peer is able to participate in only one community. Besides, we assume that a keyword cannot be used to describe more than one community.

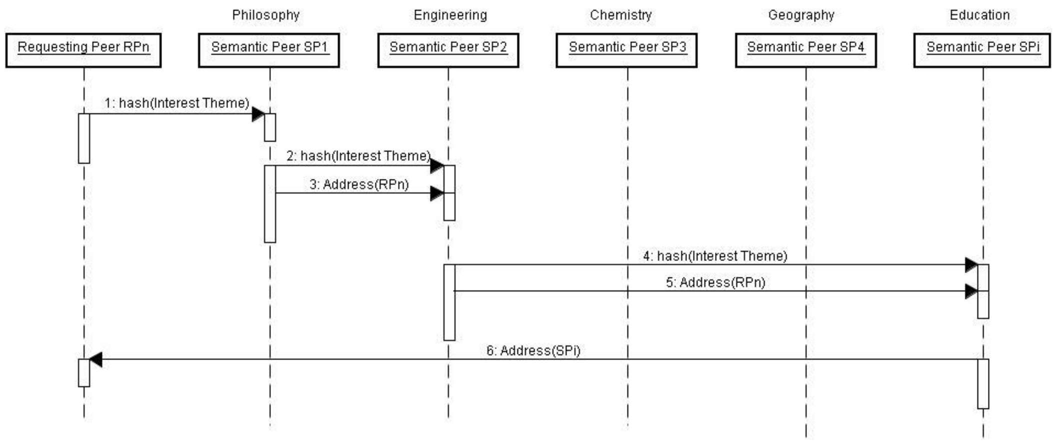


Figure 7.4. Sequence diagram describing how a semantic community is found in SPEED's DHT network.

In the following sections, we describe the steps to insert a requesting peer into a semantically similar cluster. It is assumed that a requesting peer has already discovered a corresponding semantic community in the DHT network.

7.3 SPEED's Clustering Process

Basically, the problem we are interested in can be sketched as follows: how to form semantic clusters in a semantic community considering as much as possible the semantics of the peers? Since it is not possible to predict the nature and the semantics of requesting peers, the clusters cannot be formed a priori. Moreover, due to the dynamism of participating peers it is not possible to assume a static disposal of the semantic clusters in the unstructured network. In short, the main goal of the clustering process is to minimize the semantic similarity between peers attached to distinct clusters and to maximize the semantic similarity between peers located in each cluster.

The SPEED's clustering process has the following characteristics:

- **Incremental insertion**: peers arrive one at a time;
- **Unique assignment**: peers are allocated in only one cluster (exclusive);
- **Ontology-based representation**: peers are represented by ontologies which, in turn, are represented by a collection of concepts and properties. Each

cluster is represented by a cluster ontology which corresponds to the integration of the local ontologies of the peers that compose it;

- **Parameterization**: The parameters that need to be provided are *similarity function*, *cluster threshold, neighbor threshold*, and *connect TTL*. In this work, *SemMatch* is used as a similarity function between two ontologies however any other function that returns a global measure is supported. Particularly, the last two parameters are explained in Section 7.4.2.

In the next section, we discuss the algorithm for inserting a requesting peer into a semantically similar cluster and describe its main steps.

7.4 Clustering Algorithm

In SPEED, clustering is mainly an incremental process. Peers are added to semantic clusters one at a time depending on some criterion, e.g. semantic similarity between a requesting peer and current clusters. In this sense, our clustering process is inspired in the Leader algorithm [Hartigan, 1975] described in Chapter 2 since it supports all the stated clustering characteristics. However, the Leader algorithm presents some drawbacks when applied to a P2P environment such as SPEED. Such drawbacks are listed in Table 7.1 as well as the proposed adaptations made in SPEED.

Table 7.1. Drawbacks of the Leader algorithm and proposed adaptations to SPEED.

Drawbacks of the Leader algorithm	**Introduced adaptations**
Assume that a centralized view of the clusters is available.	The clusters of each community are connected in an unstructured P2P network and should be searched accordingly. Thus, flooding is used to find clusters in a community.
Clusters are isolated, i.e. there are no links between them.	The clusters are connected through semantic correspondences which are needed to enable query processing. In this sense, we consider the definition of *semantic neighbors* to link clusters only with the most similar ones.
Since clusters are searched in a fixed order, the initial clusters tend to concentrate a high number of peers.	An initial cluster is indicated at each time a requesting peer is to be inserted. The initial cluster is obtained from a semantic index.
The comparison with all clusters may cause scalability and/or performance problems.	We limit the number of clusters to be searched in the unstructured network. Besides the initial cluster, the other clusters to be searched include the *semantic neighbors* of the initial cluster.

The introduction of the proposed adaptations results in a new clustering algorithm whose pseudo-code is described as follows. Similarly to the Leader algorithm, the proposed algorithm is also order-dependent.

```
Let ct (cluster threshold) be a similarity threshold
Let connectTTL be a search bound
Let the first requesting peer $RP_1$ be assigned to cluster $CL_1$
For each requesting peer $RP_{n+1}$
    **Search for initial cluster in Semantic Index**
    Start at the initial cluster and while connectTTL > 0 do
        simClust ← **Search for most similar semantic cluster**
        connectTTL ← connectTTL – 1
    maxSim ← GetMaximumSimilarity(simClust)
    If maxSim ≥ ct, **connect $RP_{i+1}$ to the corresponding cluster $CL_j$**
    Else, **connect $RP_{i+1}$ to a new cluster $CL_k$**
    Determine the semantic neighbors of $CL_j$ (or $CL_k$)
```

In the following subsections, we describe how semantic clusters are handled in a semantic community as requesting peers arrive. Each step of the algorithm is detailed.

7.4.1 Search for Initial Cluster in Semantic Index

In SPEED, the connection of requesting peers is continuous and unlimited. Matching a requesting peer's local ontology LO_n against all cluster ontologies CLO_{ij} of a semantic community CM_i is a costly and time-consuming task and therefore should be avoided. The main reasons for that are: (i) the size of cluster ontologies can be large since they integrate multiple local ontologies; and (ii) the number of clusters varies and cannot be predicted.

In order to provide an initial cluster to RP_n, we have a semantic index located at each semantic peer SP_i. In this semantic index, each cluster CL_{ij} of a corresponding community CM_i is represented by its summarized cluster ontology OS_{ij}.

When a requesting peer RP_n finds a semantic community CM_i, its local ontology LO_n is sent to the corresponding semantic peer SP_i. The search in the semantic index is done by matching LO_n against the summarized cluster ontologies OS_{ij} (Figure 7.5). For each index entry a global similarity measure between OS_{ij} and LO_n is produced by the ontology matching function *SemMatch*. Afterwards, SP_i determines the initial cluster by ranking in descending order the computed global similarity measures. The initial cluster will be the one associated with the highest global measure. Finally, the corresponding integration peer's address is returned to RP_n. Particularly, if no initial cluster is identified (for example, if the semantic community is empty) then RP_n creates a new cluster. In this case, RP_n connects as an integration peer. The steps required to connect a requesting peer as an integration peer will be described in the Section 7.4.3.

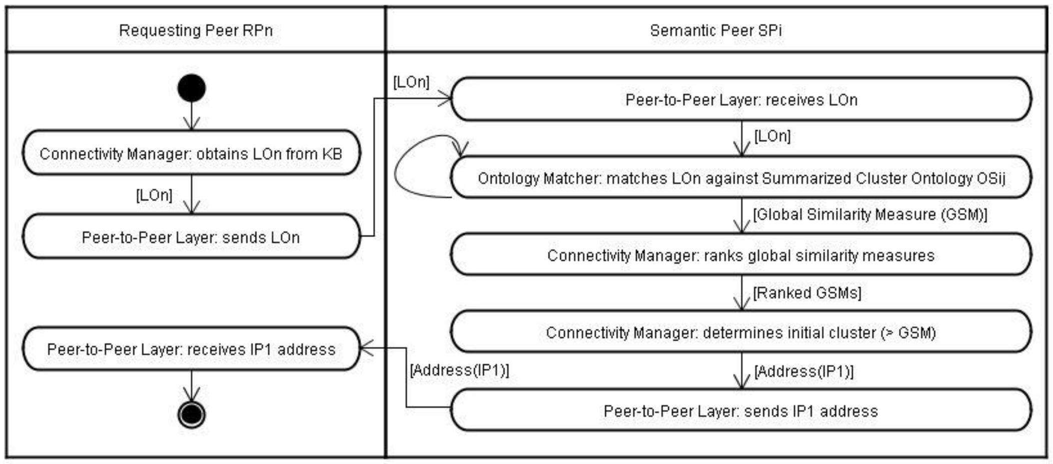

Requesting Peer RPn
Semantic Peer SPi
Connectivity Manager: obtains LOn from KB
[LOn]
Peer-to-Peer Layer: sends LOn
[LOn]
Peer-to-Peer Layer: receives LOn
[LOn]
Ontology Matcher: matches LOn against Summarized Cluster Ontology OSij
[Global Similarity Measure (GSM)]
Connectivity Manager: ranks global similarity measures
[Ranked GSMs]
Connectivity Manager: determines initial cluster (> GSM)
[Address(IP1)]
Peer-to-Peer Layer: sends IP1 address
[Address(IP1)]
Peer-to-Peer Layer: receives IP1 address

Figure 7.5. A semantic peer SP_i determines the initial cluster of a requesting peer RP_n.

7.4.2 Search for the Most Similar Semantic Cluster

Given an initial cluster, the problem now is to determine the clusters in the community CM_i that should be visited in order to search for a semantically similar cluster. To this end, the semantics of the involved peers is taken into account by extending the definition of semantic neighbors presented in Chapter 4. According to such definition, one of the conditions to consider two distinct clusters CL_{ij} and CL_{ik} as semantic neighbors is that they must share similar content, i.e. cluster ontologies. In this sense, a cluster CL_{ij} is a semantic neighbor of CL_{ik}, if the global similarity measure between CLO_{ij} and CLO_{ik} is above a certain threshold called **neighbor threshold** (denoted *nt*). Thus, given a semantic cluster CL_{ij} and its semantic neighborhood N_{ij}, a semantic cluster $CL_{ik} \in N_{ij}$ is such that $SemMatch(CLO_{ij}, CLO_{ik}) \geq nt$.

In addition, given the neighborhood $N_{ij} = \{CL_{i1}, CL_{i2},\ldots,CL_{ik}\}$ of a cluster CL_{ij}, all the clusters in N_{ij} are considered **direct neighbors** of CL_{ij}. If a cluster CL_{in} is not included in N_{ij} but is contained in N_{ik} (i.e. the neighborhood of CL_{ik})

then we say that CL_{in} is an **indirect neighbor** of CL_{ij}. In Figure 7.6, CL_{i3} and CL_{i4} are direct neighbors of CL_{i2} and indirect neighbors of CL_{i1}.

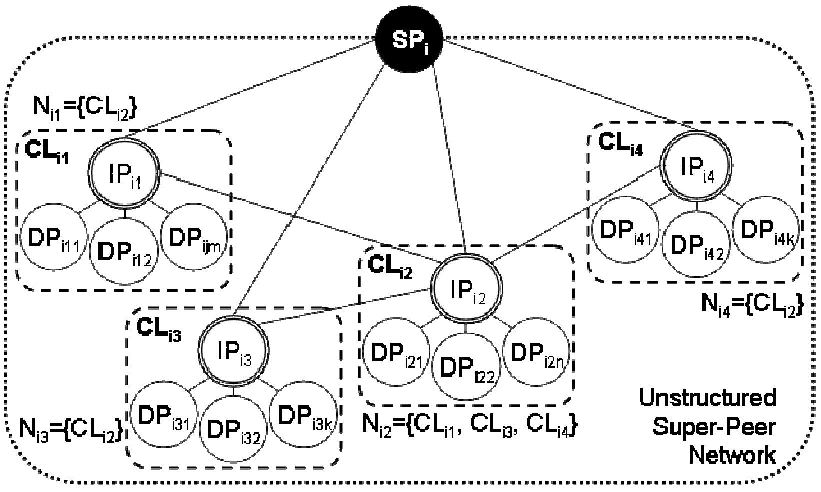


Figure 7.6. An example of direct and indirect neighbors.

Based on the extended definition of semantic neighbors, several possible search strategies can be derived in order to limit the number of clusters to be searched. All of them can be controlled by a TTL limit (denoted *connect TTL*). For instance, if:

- *connect TTL* = 1, the search scope is resumed to the initial cluster;
- *connect TTL* = 2, the search scope includes the initial cluster and its direct semantic neighbor(s);
- *connect TTL* ≥ 3, the search scope includes the initial cluster as well as its direct and indirect semantic neighbor(s).

The search is started when RP_n sends its local ontology LO_n to the integration peer corresponding to the initial cluster. At the integration peer, *SemMatch* is executed by taking as arguments the current cluster ontology and LO_n. The resulting global similarity measure is returned to RP_n. According to the defined search strategy, LO_n can be propagated to the direct and/or indirect semantic neighbors of the initial cluster. At each visited cluster, *connect TTL* is decreased and the search process continues. The search finishes when *connect TTL* reaches zero. To avoid waiting indefinitely for matching results a timeout is set by RP_n when RP_n sends its LO_n to the initial cluster.

Figure 7.7 is the UML sequence diagram representing the community instantiation of Figure 7.6. Since *connect TTL* is set to 3, the four clusters (CL_{i1}, CL_{i2}, CL_{i3}, and CL_{i4}) are visited in order to determine the most similar cluster for the requesting peer RP_n. The clusters CL_{i1}, CL_{i2}, CL_{i3}, and CL_{i4} are represented by their corresponding integration peers IP_{i1}, IP_{i2}, IP_{i3}, and IP_{i4}, respectively. The first visited integration peer is IP_{i1} that corresponds to the initial cluster provided by the semantic peer SP_i. The search scope comprises the direct (IP_{i2}) and indirect (IP_{i3} and IP_{i4}) semantic neighbors of IP_{i1}. The global similarity measures returned to RP_n are: 0.5 (IP_1), 0.6 (IP_2), 0.2 (IP_3), and 0.3 (IP_4). These measures are used by RP_n to decide whether to join one of the visited clusters or create a new one (in this case, CL_{i5}). Such process is described in the next section.

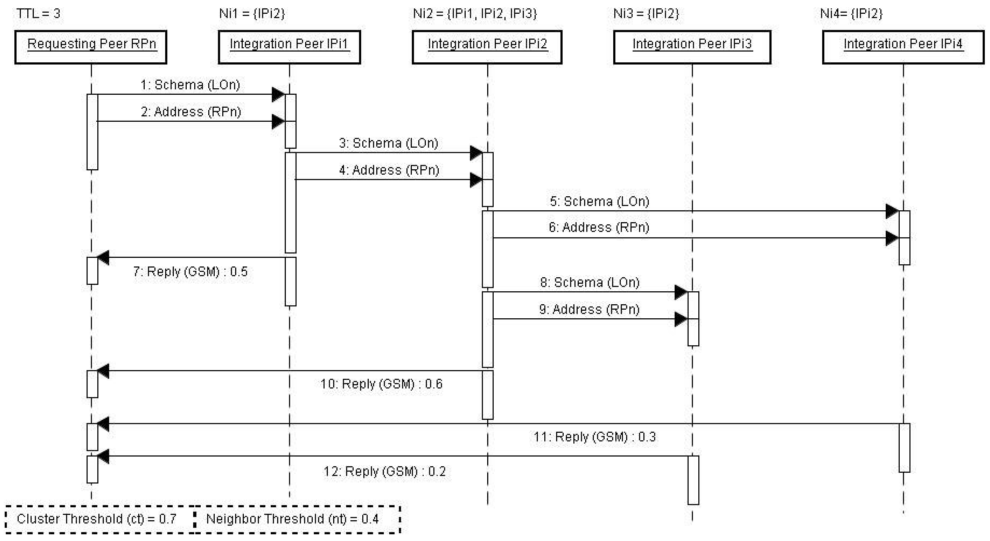


Figure 7.7. A requesting peer RP_n searches for a semantically similar cluster.

7.4.3 Connection of a Requesting Peer

Once RP_n receives the global similarity measures from the visited clusters, RP_n must select the highest global similarity measure. In this case, two possible cases can occur:

Case 1

If the selected measure is equal or higher than *cluster threshold* (*ct*), then RP_n joins the corresponding cluster as a data peer. In this case, a new version of CLO_{ij} is produced by merging the current CLO_{ij} and the local ontology LO_n of the new data peer. To this end, an ontology merging process **Merge** is considered. *Merge* takes as arguments the two peer ontologies and the ontology alignment between them. Such alignment was produced by *SemMatch* when both ontologies were matched during the search process. As a result, *Merge* builds a new version of CLO_{ij} as well as a set of semantic correspondences between CLO_{ij} and LO_n which are needed for query processing. The new CLO_{ij} includes all the elements contained in both input ontologies. In addition, a new summary of CLO_{ij} (OS_{ij}) is built and the semantic index is updated accordingly.

Case 2

If the selected measure is lower than *cluster threshold* (*ct*), RP_n creates a new cluster and joins that cluster as an integration peer. In this case, the cluster ontology CLO_{ij} of the new cluster corresponds to the local ontology LO_n describing RP_n. The semantic neighborhood of the new cluster is composed of all the visited clusters CL_{ik} such that $nt \leq SemMatch(CLO_{ij}, CLO_{ik}) < ct$. A summarized version of CLO_{ij} (OS_{ij}) is built and a new entry is added to the semantic index. In Figure 7.7, *neighbor threshold* (*nt*) and *cluster threshold* (*ct*) are set to 0.4 and 0.7, respectively. Since the highest global similarity measure returned by the searched clusters (i.e. 0.6, returned by CL_{i2}) is lower than *ct*, the requesting peer RP_n will create a new cluster (CL_{i5}). The semantic neighborhood of CL_{i5} is defined as $N_{i5} = \{CL_{i1}, CL_{i2}\}$. Next, we make some considerations about cluster maintenance.

7.5 Maintenance Considerations

Although cluster maintenance is not the main focus of this work, it is necessary to make some important considerations about this issue. In order to reflect the content available in a semantic cluster, cluster ontologies should be created and maintained dynamically and in an automatic way [Haase and Stojanovic, 2005; Konstantinidis *et al.*, 2008]. They should be adaptable to the changes in the semantics of the peers that participate in the cluster. A cluster ontology should be able to evolve not only when a requesting peer joins the cluster (as seen in Section 7.4.3) but also when a participating peer leaves it. In this section, we present some considerations about peer disconnection, another event that requests a cluster ontology to be updated. In addition, we discuss what might happen in the system when a cluster ontology evolves.

7.5.1 Disconnection of Participating Peers

The disconnection of a data peer DP_{ijk} implies in updating the cluster ontology CLO_{ij} of its corresponding cluster CL_{ij}. The elements that are shared only by DP_{ijk} are logically removed from CLO_{ij}, along with the associated semantic correspondences. Logical deletion is preferred because DP_{ijk} can reconnect to the same cluster CL_{ij} in a near future. Physical deletion occurs only after a certain time interval. Once CLO_{ij} is updated, a new summary of it (OS_{ij}) is built and the semantic index is updated accordingly.

The procedure to update CLO_{ij} as the result of an integration peer disconnection is basically the same. However, the disconnection of IP_{ij} requires an additional effort: the selection of a new integration peer for the corresponding cluster. Next, we present two consequences of evolving a cluster ontology.

7.5.2 Update of Cluster Neighborhood

When a cluster ontology evolves, there might be some changes in the neighborhood N_{ij} of a cluster CL_{ij}. Thus, the global similarity measures between CLO_{ij} and each cluster ontology in N_{ij} needs to be recomputed. If the global similarity measure between CLO_{ij} and a cluster ontology $CLO_{ik} \in N_{ij}$ decreases to a value that is below *neighbor threshold* (*nt*), then the cluster CL_{ik} is removed from N_{ij}. On the other hand, if the similarity value increases and becomes higher than or equal to *cluster threshold* (*ct*), then the two (neighbor) clusters CL_{ij} and CL_{ik} need to be transformed into a single cluster.

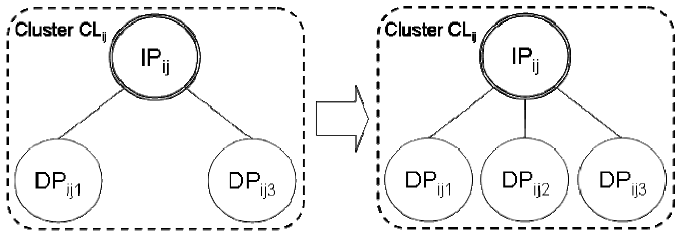
7.5.3 Recalculation of Global Similarity Measure

Another consequence of a cluster ontology evolution is described as follows. Assume that a requesting peer RP_n has joined a cluster CL_{ij} as the data peer DP_{ijk}. Thus, its LO_n is now referred as to LO_{ijk}. During the interval DP_{ijk} remains connected to CL_{ij}, CLO_{ij} may evolve in such a way that the similarity between CLO_{ij} and LO_{ijk} can be increased or decreased. In this sense, the assumption that, given a semantic cluster CL_{ij}, each data peer DP_{ijk} in CL_{ij} is such that *SemMatch*(CLO_{ij}, LO_{ijk}) $\geq$ *ct* can only be considered when a requesting peer joins a cluster.

The similarity increases when data peers sharing some dissimilar content to DP_{ijk} leave CL_{ij}. In this case, no cluster maintenance is needed and DP_{ijk}

should continue attached to CL_{ij}. Otherwise, if the global similarity measure between CLO_{ij} and LO_{ijk} decreases then some cluster maintenance is needed. The similarity decreases when data peers sharing some additional (dissimilar) content join CL_{ij} after DP_{ijk}. For example, this can be caused if the *cluster threshold* (*ct*) is initialized to a low value.

For instance, consider the scenario described in Figure 7.8. It illustrates a cluster CL_{ij} containing one integration peer (IP_{ij}) and two data peers (DP_{ij1} and DP_{ij3}). The values in the tables represent the global similarity measures between each distinct pair of ontologies in CL_{ij}. Particularly, in SPEED the measure between a CLO_{ij} and a local ontology LO_{ijk} is computed before a data peer joins a cluster CL_{ij}. The other measures (between the local ontologies) are not calculated during the clustering process and are presented only to illustrate the example. The table of Figure 7.8a shows that the data peers DP_{ij1} and DP_{ij3} share a high similar content since the global similarity measure between their local ontologies (LO_{ij1} and LO_{ij3}) is 0.9.



	CLO_{ij}	LO_{ij1}	LO_{ij3}
CLO_{ij}	-	0.8	0.9
LO_{ij1}	0.8	-	0.9
LO_{ij3}	0.9	0.9	-

(a)

	CLO_{ij}	LO_{ij1}	LO_{ij2}	LO_{ij3}
CLO_{ij}	-	0.2	0.7	0.6
LO_{ij1}	0.2	-	0.5	0.8
LO_{ij2}	0.7	0.5	-	0.6
LO_{ij3}	0.6	0.8	0.6	-

(b)

Figure 7.8. Cluster maintenance as a result of the cluster ontology evolution.

In Figure 7.8b, a new data peer (DP_{ij2}) joins CL_{ij}. DP_{ij2} not only shares similar content with DP_{ij1} and DP_{ij3} but also has some additional content that is not available in the other two data peers. As a result, new ontology elements are introduced in CLO_{ij} and the similarities between CLO_{ij} and DP_{ij1} as well as between CLO_{ij} and DP_{ij3} are decreased. In this case, cluster maintenance can be performed in two ways: (i) the most dissimilar data peer in CL_{ij} (in this case,

DP_{ij1}) must move to a more semantically similar cluster; or (ii) CL_{ij} must be split into two new clusters.

7.6 Experiments

In this section we discuss implementation issues and provide a selection of the experiments that we have performed to verify the effectiveness of the proposed clustering process.

7.6.1 Implementation

For our experiments we have developed a simulator through which we were able to reproduce the main conditions characterizing the SPEED's environment. The simulator was implemented in Java [Java, 2009] using the Eclipse Integrated Development Environment (IDE) [Eclipse, 2009]. Through the simulator we were able to generate scenarios corresponding to networks of peers, each with its own schema describing a particular reality. In the simulator, we assume that there exists a communication facility among the peers that enables sending and receiving information, i.e. queries, data, and schema information. A more detailed description of the simulator is provided in the Appendix C. In this current version, the tool is able to simulate requesting peer connection and the formation of clusters in a given semantic community. Concerning the maintenance considerations discussed in Section 7.5, only the update of a cluster neighborhood is implemented.

In order to execute the experiments, we included in the simulator the ontology management tools (ontology matching and ontology summarization) previously described.

Concerning ontology merging, we have not found any automatic tool that could be integrated with the simulator. Therefore, we also had to develop a new one. The merging tool was also developed in Java and the OWL API [OWL API, 2009] was used to handle ontologies. Basically, the tool can be invoked from command line and accepts as input two OWL ontologies. As a result, the merging tool generates a new OWL ontology containing all elements (concepts and properties) of the input ontologies. Repeated elements are not included.

7.6.2 Experimental Setting

Our simulation tests were conducted considering the *education* knowledge domain. Therefore, we have built an ontology library containing dozens of local ontologies to be used by requesting peers during the tests. Each local ontology contains about six concepts on average. The local ontologies were derived from the real-world ontology *UnivCSCMO.owl* illustrated in the Appendix A. The *UnivCSCMO.owl* ontology describes a computer science academic department and was also used as the community ontology. During our tests, we have assumed that the element names of the local ontologies were normalized according to the element names of the chosen community ontology.

7.6.3 Validation

All tests were performed in an Intel Pentium M 1.60GHz, 1GB of RAM. The operating system was Windows XP®. In our experiments, the SPEED's DHT network was first created with some semantic peers. Afterwards, we started the connection of requesting peers one at a time. Each requesting peer has searched for a corresponding semantic community (*education*) in the DHT network and then for a semantically similar cluster in the unstructured network of the discovered community.

In the following, we demonstrate the effectiveness of the proposed clustering process from two different points of view. First, we measure the generation of semantic clusters using the clustering indices presented in Chapter 2. Afterwards, we evaluate the resulting network by executing query processing simulations.

Clustering Indices

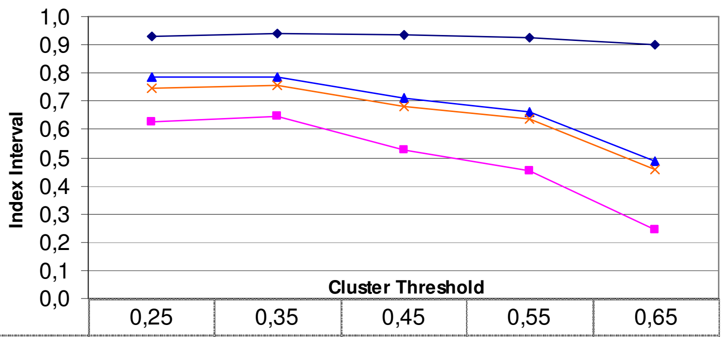
We have evaluated the clustering results using the classical *external* and *internal* cluster validity approaches (Chapter 2). External validity was measured using the classical statistical indices: *Rand Index* [Theodoridis and Koutroumbas, 2003], *Jaccard Coefficient* [Batistakis *et al.*, 2002a], *Fowlkes-Mallows (FM) Index* [Fowlkes and Mallows, 1983], and *Hubert's statistic* [Batistakis *et al.*, 2002b]. The indices were computed through a comparison between the clustering results obtained from the simulator against an ideal one generated by a hierarchical clustering algorithm [Jain *et al.*, 1999]. The hierarchical algorithm follows the *batch approach* (Chapter 2) for clustering a set of peers. In other words, it considers the set as a whole and begins to organize peers into meaningful clusters.

SPEED's clustering algorithm is incremental and order-dependent. Particularly, due to the second characteristic, the indices were calculated multiple times considering different orders of requesting peers. Then, for each of the statistical indices an average of the index results was calculated.

The values of these statistical indices are between 0 and 1. However, a requirement for achieving the maximum value is to have the same number of clusters in both clustering results, which, as we observed, is not always possible. For all the used indices, the larger their value the higher the agreement between the two clustering results.

Two types of experiments were performed. In the first one, we considered the search strategy proposed in SPEED (denoted *limitedClusters*). According to such strategy, a requesting peer receives an initial cluster and visits only a limited number of clusters, i.e. the direct and indirect semantic neighbors of the initial cluster. In the second experiment (denoted *allClusters*), we considered a different search strategy. Each requesting peer visits all current clusters before connecting to the system. In this case, the semantic index is discarded.

For the *limitedClusters* strategy, the following parameters (with respective values) were considered: *summary size* (6), *neighbor threshold* (0.10), and *connectTTL* (3). Figure 7.9 depicts the index results for different values of *cluster threshold*.

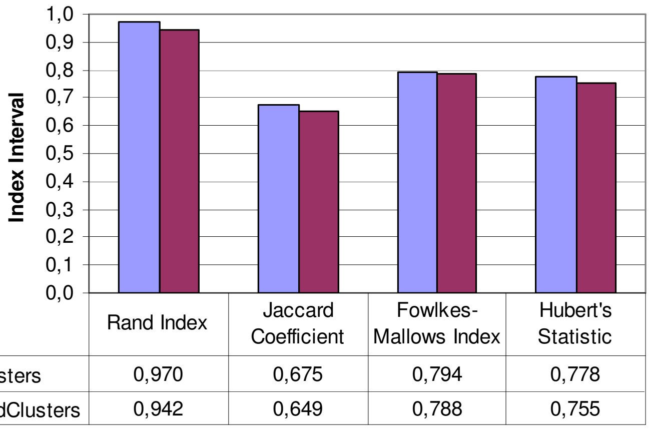


	0,25	0,35	0,45	0,55	0,65
◆ Rand Index	0,928	0,942	0,935	0,928	0,901
■ Jaccard Coefficient	0,629	0,649	0,530	0,454	0,246
▲ Fowlkes-Mallows Index	0,785	0,788	0,713	0,664	0,486
× Hubert's Statistic	0,748	0,755	0,682	0,636	0,458

Figure 7.9. Clustering evaluation: external indices.

The results indicate that the agreement between the clustering result produced by the simulator and the ideal one is degraded as the value of *cluster threshold* increases. For the set of local ontologies used in the tests, the highest agreement has been obtained when *cluster threshold* was adjusted to 0.35. The highest agreement could have been obtained for a different value of cluster threshold if a distinct set of local ontologies had been used.

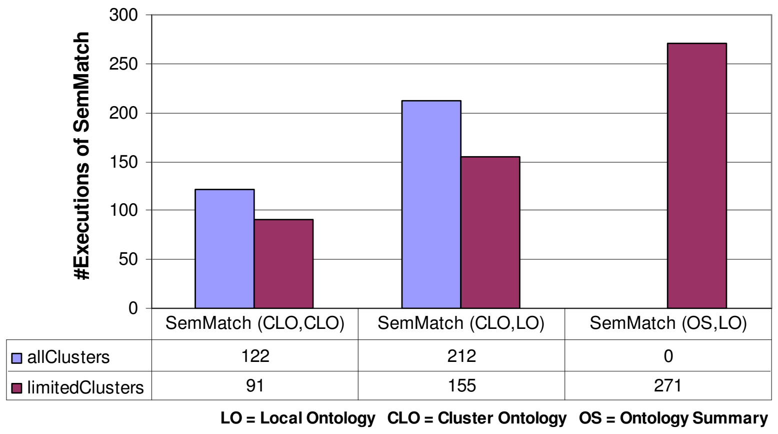
For the *allClusters* strategy, the clustering result was also compared with the ideal one. To guarantee that all clusters were visited we have modified the parameters *connectTTL* (999) and *neighbor threshold* (0). *Summary size* remained unaltered. Afterwards, we compared the obtained index results against the best ones of *limitedClusters*, i.e. when *cluster threshold* was set to 0.35. Naturally, *allClusters* tends to produce better index results than *limitedClusters*. However, we have obtained similar index results (Figure 7.10) with fewer executions of *SemMatch* (Figure 7.11) and less number of messages transmitted among peers in the simulated network (Figure 7.12).



	Rand Index	Jaccard Coefficient	Fowlkes-Mallows Index	Hubert's Statistic
■ allClusters	0,970	0,675	0,794	0,778
■ limitedClusters	0,942	0,649	0,788	0,755

Figure 7.10. A comparison of search strategies using internal statistical indexes.

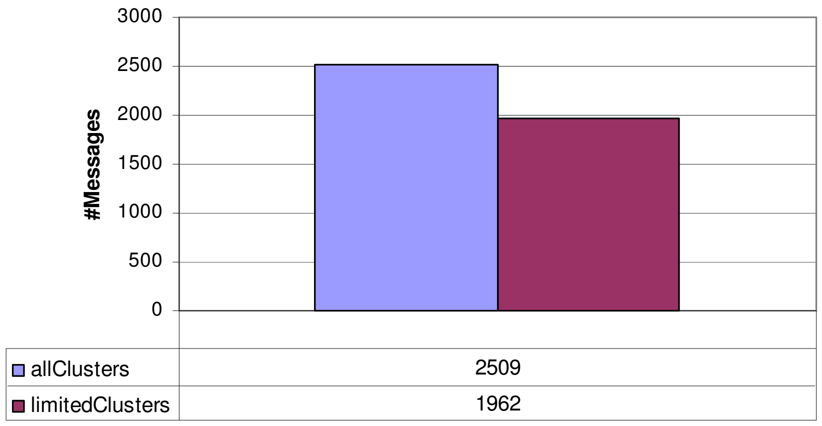
Such decrease is explained because when a requesting peer arrives at a semantic community, only a limited number of clusters are visited in order to determine the most similar cluster for the requesting peer. Consequently, the number of ontology matching executions is minimized. The numbers available in Figure 7.11 indicate a reduction of 27% of matchings involving local ontologies and cluster ontologies, and a reduction of 25% of matchings between cluster ontologies.



	SemMatch (CLO,CLO)	SemMatch (CLO,LO)	SemMatch (OS,LO)
🟦 allClusters	122	212	0
🟥 limitedClusters	91	155	271

Figure 7.11. A comparison of search strategies considering the number of executions of *SemMatch*.

Since a requesting peer's local ontology is propagated only among a limited number of semantic neighbors of the initial cluster, the quantity of messages transmitted among integration peers is also decreased. The numbers illustrated in Figure 7.12 indicate a reduction of 22%.

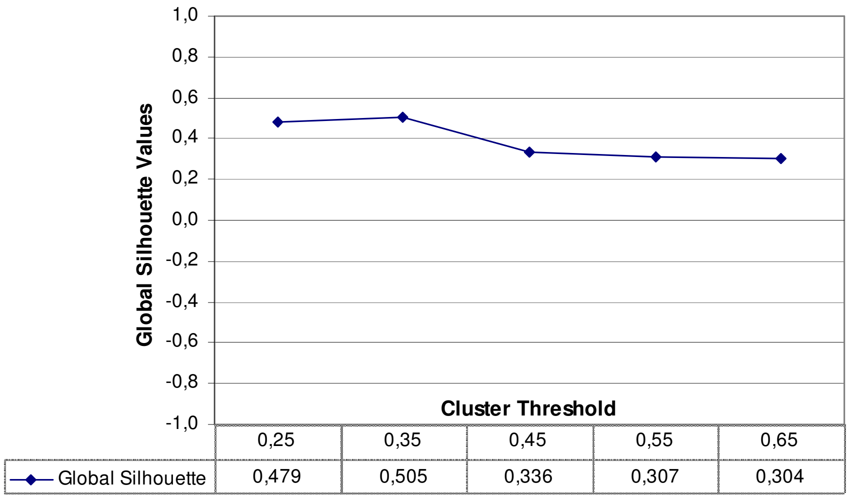


🟦 allClusters	2509
🟥 limitedClusters	1962

Figure 7.12. A comparison of search strategies considering the number of messages transmitted among peers.

Internal validity was evaluated using the Silhouette indices [Rousseeuw, 1987]. Such indices are useful when it is seeking compact and clearly separated clusters. In this case, there are two interesting issues to be analyzed in a clustering result: the homogeneity of each cluster and the degree of separation between the obtained clusters. The higher their homogeneity and the separation the better is the clustering result. Both aspects can be captured in a global

Silhouette value between –1 (bad clustering) and 1 (very good clustering). Figure 7.13 illustrates the global Silhouette values for different values of *cluster threshold*. Again, the best clustering result (0.505) for *limitedClusters* has been obtained when *cluster threshold* was set to 0.35.



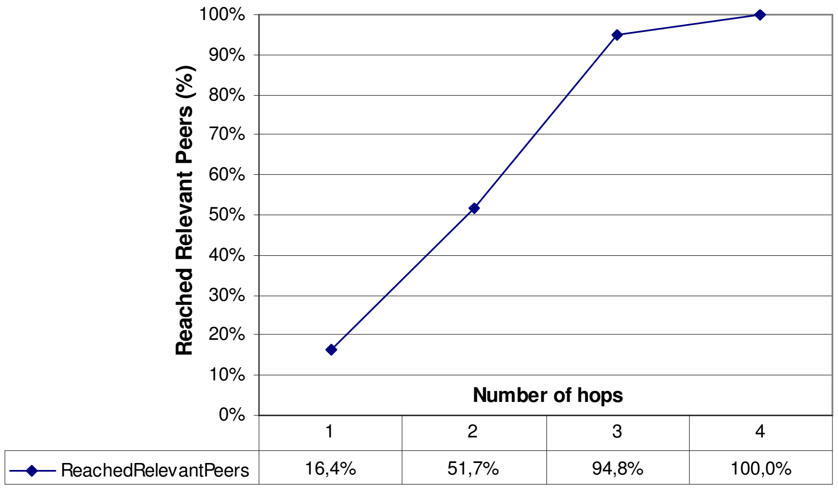
	0,25	0,35	0,45	0,55	0,65
—◆— Global Silhouette	0,479	0,505	0,336	0,307	0,304

Figure 7.13. Clustering validity: internal indices.

Query Processing

In order to evaluate the impact of the proposed clustering process on query answering, we simulated query routing on the networks produced by the proposed clustering algorithm. Basically, query routing was simulated by posing five different queries on randomly selected peers. Each query was a combination of small number of predicates specifying conditions on concepts. The set of relevant peers in the network that were able to answer each query was previously identified manually, considering as a relevant peer a peer that can answer a query integrally. We quantified the advantages on query processing by propagating each query until a stopping condition was reached, i.e., a TTL. Queries were propagated among semantic neighbor clusters. In this sense, we evaluated the effectiveness improvement by measuring the percentage of relevant peers that were reached for different number of hops.

The results illustrated in Figure 7.14 correspond to the network generated when cluster threshold was configured to 0.35. In this network, the 45 requesting peers were clustered into seven clusters in the *education* community and each cluster contained six peers on average. The results indicate that one hop was needed to reach 17% of the relevant peers. This means routing a query to relevant peers participating in the same cluster of the peer in which the query was posed. In addition, three hops were necessary to reach almost all relevant peers. This means routing a query to the direct and indirect neighbors (three at maximum) of the cluster in which the query was posed.



	1	2	3	4
ReachedRelevantPeers	16,4%	51,7%	94,8%	100,0%

Figure 7.14. Percentage of reached relevant peers for a given number of hops.

7.7 Considerations

The proposed clustering process can bring several benefits to the organization of peers in a PDMS. Semantically similar peers are clustered according to their knowledge domain (communities) and local ontologies (clusters). If a semantic cluster is discovered by a requesting peer, several semantic similar peers are also found. By using a semantic index the search for a semantic cluster is not started randomly in a community. Thus, the probability of finding a semantically similar cluster for RP_n increases. Furthermore, since the search for clusters is guided by the semantics of the participating peers, irrelevant semantic clusters are discarded. If there is a semantic cluster which is similar to RP_n, then such cluster can be found by RP_n in a shorter number of hops.

Our experimental evaluation has shown the effectiveness of the proposed clustering process. We are aware that limiting the number of clusters to be visited when requesting peer arrives can lead to situations in which the most semantically similar cluster is not found. However, the use of a semantic index has shown that satisfactory clustering results can be obtained. In addition, limiting the number of visited clusters minimizes the number of ontology matching executions as well as the number of messages transmitted in the network. Finally, queries posed at peers can reach relevant peers in a few hops. In the next chapter, we present our conclusions and suggestions for future work.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

"When we think we know all the answers, life comes and changes all the questions"
Anonymous

In this thesis, we have proposed an incremental process to cluster semantically similar peers in a PDMS. Peers are organized in the network according to a mixed P2P topology (DHT, unstructured, and super-peer). Ontologies are employed in the PDMS to improve some of its main services, e.g. to represent the exported schema shared by (a set of) peers. Exported schemas (ontologies) are used to group semantically similar peers into communities and clusters.

Peer clustering is assisted by two other processes: ontology matching and ontology summarization. An ontology matching process produces a global measure which is mainly used to determine the similarity between peers. An ontology summarization process produces summaries of clusters ontologies. The summaries are used as a semantic index to indicate an initial cluster for requesting peers. The initial cluster serves as a starting point in order to locate other semantically similar peers.

8.1 Research Contributions

The main contributions of this work are summarized as follows.

PDMS Architecture

We have extended the original definition of OPDMS arguing that ontologies can be used in a broader way to enhance PDMS services. Based on our analysis of the state-of-the-art on PDMS, we have identified six high-level requirements that an OPDMS should fulfill in order to take advantage of using ontologies to enhance its services: R_1) Exported schema representation; R_2) Global conceptualization; R_3) Support for correspondences identification; R_4) Support for query processing; R_5) Semantic index; and R_6) Semantic matching capabilities. None of the discussed PMDS (OntSum, Sunrise, and Helios) satisfied the requirements completely. In this sense, we have proposed a semantic-based PDMS which fulfills all the identified requirements. Table 8.1 illustrates the different components of SPEED which are used to satisfy each one of the requirements.

Table 8.1. The resources used in SPEED to satisfy the identified high-level requirements of an OPDMS.

R_1	R_2	R_3	R_4	R_5	R_6
Ontology Translator; Local Ontologies	Ontology Merger; Cluster and Community Ontologies	Community Ontologies	Cluster Ontologies	Ontology Summarizer; Summarized Cluster Ontologies	Ontology Matcher (SemMatch)

The internal modules of the three types of peers used in SPEED (data peers, integration peers, and semantic peers) have been described as well as the different types of ontologies used in the system (local ontologies, cluster ontologies, and community ontologies). A simulator has been developed through which we were able to reproduce the main conditions characterizing the SPEED's environment.

Ontology Matching Process

We have proposed a semantic-based ontology matching process (*SemMatch*). Differently from other matching processes, *SemMatch* produces, besides an ontology alignment between two ontologies, a global similarity measure representing the overall similarity degree between them (and not only between their elements!). Particularly, in SPEED, a global measure is needed in many situations, e.g. to determine the similarity between a requesting peer and an existing cluster. We have developed a tool implementing the semantic-based ontology matching process. The tool has been submitted to experimental evaluation and integrated to our PDMS simulator. Experimental results have shown that the combination of linguistic, structural and semantic matchers can improve ontology alignments.

Ontology Summarization Process

We have proposed an automatic process for building summaries of cluster ontologies. The process is divided into several steps and is based on the notions of centrality and frequency. Particularly, the use of frequency has not been investigated before in other works. In this thesis, the use is motivated by the fact that a cluster ontology is obtained by merging several different local ontologies. The use of frequency as a measure to determine the relevance of concepts of an ontology minimizes the need to update the semantic index. If the most frequent concepts are included in the summary that represents a cluster, when the associated cluster ontology evolves the most relevant concepts tends to remain unaltered. Consequently, there is no need to generate a new summary. The main contributions of this topic include a preliminary implementation and evaluation of an ontology summarization tool which has also been integrated to the PDMS simulator.

Peer Clustering Process

We have proposed an incremental process for clustering peers in a PDMS. According to the clustering process, peers are first grouped into a corresponding semantic community and then into a semantically similar cluster. Instead of visiting all current clusters when a requesting peer arrives at the semantic community, the basic idea is to start at a promising cluster and visit only a subset of the clusters. The proposed process was implemented in the simulator and submitted to experimental evaluation. Validation has been performed using clustering indices and by executing query processing simulations. The results have shown that homogeneous and well-separated clusters can be generated if an ontology-based clustering process is used to organize peers as soon as they connect to the system.

8.2 Future Work

This work has raised a large spectrum of new problems to be solved, which are listed as follows. The problems are organized according to each one of the research contributions. Although not directly related to this work, other types of problems are also indicated, e.g. load balancing and fault tolerance.

Ontology Matching Process

As further work, the ontology matching tool can be extended by considering the properties of the concepts both in the correspondences identification and in the determination of the global similarity measure. Furthermore, new global similarity measures between ontologies can be proposed as well as other existing ones can be implemented.

Ontology Summarization Process

There are a number of ongoing research issues concerned with the proposed summarization process which will be the goal of our future activity. An issue to be studied in deep detail regards the application of transitivity rules to identified paths in order to eliminate non-relevant concepts. In some situations, instead of adding non-relevant concepts in the summary, some relationships between relevant concepts could be inferred. The main idea is to automatically derive new relationships between relevant concepts which are separated by a non-relevant concept, and then remove the non-relevant concept and its relationships.

Another research activity is devoted to executing experiments with the other types of centrality measures (closeness, betweenness, and eigenvector). We think that more accurate sets of relevant concepts can be produced if different centrality measures are applied.

Peer Clustering Process

Regarding the ontology-based clustering process, there are some interesting issues that can be the goal of future research. For instance, during the clustering process, the fact that participating peers can leave the system may be considered. In this case, the Ontology Manager component, located at the integration peers, needs to be implemented. The component is responsible for updating a cluster ontology whenever a data peer leaves the cluster, reflecting the current content shared inside a cluster. Also, the simulator can be adapted to consider parallel connection and disconnection of peers. This modification will probably require the use of threads [Oaks and Wong, 2004].

Clearly, the search for an initial cluster in a semantic index needs to be improved. The basic idea is to avoid full index scans and minimize the number of matching between a local ontology (requesting peer) and the summarized cluster ontologies. To this end, the summaries should be organized in such a way that the initial cluster (i.e. the one with the highest global measure) should be determined with as few matching as possible.

Load Balancing

The dynamic behavior of data peers and integration peers can lead to situations where the overlay network of a community may need to be reorganized. For instance, if most of the connected peers are integration peers, the system is more like an unstructured P2P network and several peers will participate in query processing. On the other hand, if too few integration peers are available, the system is more like a centralized network.

In this sense, a graph-based clustering algorithm [Steiner and Biersack, 2005; Hammouda and Kamel, 2007] can be used to avoid the previously mentioned situations. Such algorithm can be adapted to periodically balance current clusters and still maintain the semantic organization of peers in the overlay network. Clearly, some operations to be considered in the algorithm are: (i) the redistribution of data peers between the semantic neighbors of an overloaded cluster; and (ii) the merging of two clusters or the split of an existing cluster into two new clusters.

Fault Tolerance

When an integration peer fails or disconnects, a fault tolerance approach must be available in order to maintain the corresponding data peers connected. A pro-active solution can be used in such a way that one of the data peers of a particular cluster should be previously elected as a candidate integration peer. In this case, the candidate acts as a redundant integration peer and keeps a copy of the actual integration peer's knowledge base. The knowledge base needs to be periodically replicated from the actual integration peer to the candidate integration peer. If the actual integration peer fails, then the candidate integration peer assumes its role and another data peer is chosen as candidate integration peer.

Since integration peers are responsible for executing important issues within a cluster, several characteristics need to be considered so a data peer can become an integration peer candidate. Such characteristics include physical resources available such as physical memory, disk space, CPU powerfulness, and network bandwidth. Additionally, the behavior of a data peer, while it is connected to the system, should be an essential factor when determining an integration peer candidate. Thus, subjective characteristics are also taken into account, for example, availability, accuracy, response time, completeness, and amount of data.

REFERENCES

Aberer et al., 2002 Aberer, K., Cudre-Mauroux, P., and Hauswirth, M.: A Framework for Semantic Gossiping. In: SIGMOD Record, 31(4) (2002)

Arenas et al., 2003 Arenas, M., Kantere, V., Kementsietsidis, A., Kiringa, I., Miller, R. J., and Mylopoulos, J.: The Hyperion Project: From Data Integration to Data Coordination. In: SIGMOD Record, Special Issue on Peer-to-Peer Data Management, 32(3):53-58 (2003)

Aumüller et al., 2005 Aumüller, D., Do, H. H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: International Conference on Management of Data (SIGMOD), Software Demonstration (2005)

Baader et al., 2003 Baader, F., Calvanese, D., McGuinness, D., Nardi D., and Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)

Baeza-Yates and Ribeiro-Neto, 1999 Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Harlow, England, ACM Press (1999)

Barrasa et al., 2004 Barrasa, J., Corcho, O., and Gómez-Pérez, A.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: 2nd Workshop on Semantic Web and Databases (SWDB'04), pages 1069-1070, Toronto, Canada (2004)

Batini et al., 1986 Batini, C., Lenzerini, M., and Navathe, S.: A comparative analysis of methodologies for database schema integration. In: ACM Computing Surveys, 18(4):323-364 (1986)

Batistakis et al., 2002a Batistakis, Y., Halkidi, M., and Vazirgiannis, M.: Cluster validity methods: Part I. In: Sigmod Record, 31(12) (2002)

Batistakis et al., 2002b Batistakis, Y., Halkidi, M., and Vazirgiannis, M.: Clustering validity checking methods: Part II. In: Sigmod Record, 31(3) (2002)

Bellahsène and Roantree, 2004 Bellahsène, Z., Roantree, M. Querying Distributed Data in a Super-Peer Based Architecture. In: 15th Int. Conf. on Database and Expert Systems Applications (DEXA'04), Volume 3180 of LNCS, pages 296-305, Zaragoza, Spain (2004)

Berkhin, 2002 Berkhin, P.: Survey of Clustering Data Mining Techniques. Accrue Software, Inc. San Jose, CA, USA (2002)

Berners-Lee et al., 2001 Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web. Scientific American (2001)

Bernstein et al., 2002 Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I.: Data management for peer-to-peer computing: A vision. In: 5th International Workshop on the Web and Databases (WebDB), pages 89-94, Madison, USA (2002)

Bizer, 2003	Bizer, C.: D2R MAP – A Database to RDF Mapping Language. In: 12th Int. World Wide Web Conference (Posters), Budapest, Hungary (2003)
Bonacich, 1972	Bonacich, P.: Factoring and Weighting Approaches to Status Scores and Clique Identification. In: Journal of Mathematical Sociology, Volume 2, Number 1, pages 113-120 (1972)
Booch et al., 2005	Booch, G., Rumbaugh, J., and Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley, 2nd Edition (2005)
Castano et al., 1998	Castano, S., Antonellis, V., Fugini, M. G., Pernici, B. : Conceptual Schema Analysis : Techniques and Applications. In: ACM Transactions on Database Systems, Volume 23, Number 3, pages 286-333 (1998)
Castano et al., 2003	Castano, S., Ferrara, A., Montanelli, S., Pagani, E., Rossi, G. P.: Ontology-Addressable Contents in P2P Networks. In: 1st WWW International Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID'03), in conjunction with 12th International World Wide Web Conference (WWW'03), pages 55-68, Budapest, Hungary (2003)
Castano et al., 2004	Castano, S., Ferrara, A., Montanelli, S., and Racca, G.: Semantic Information Interoperability in Open Networked Systems. In: International Conference on Semantics of a Networked World (ICSNW'04), pages 215-230, Paris, France (2004)
Castano and Montanelli, 2005	Castano, S., Montanelli, S.: Semantic Self-Formation of Communities of Peers. In: ESWC Workshop on Ontologies in Peer-to-Peer Communities, Heraklion, Greece (2005)
Castano et al., 2006	Castano S., Ferrara, A., and Montanelli, S.: Matching ontologies in open networked systems: Techniques and applications. In: Journal on Data Semantics, V:25-63 (2006)
Cerbah, 2008	Cerbah, F.: Learning Highly Structured Semantic Repositories from Relational Databases – The RDBtoOnto Tool. In: 5th European Semantic Web Conference (ESWC'08), pages 777-781, Tenerife, Spain (2008)
Cheeseman and Stutz, 1996	Cheeseman, P., Stutz, J.: Bayesian Classification (AutoClass): Theory and Results. In: U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 153-180, AAAI/MIT Press (1996)
Crespo and Garcia-Molina, 2002	Crespo, A., Garcia-Molina, H. Semantic Overlay Networks for P2P Systems. Technical Report, Stanford University (2002)
Cullot et al., 2007	Cullot, N., Ghawi, R., and Yétongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping". In: 15th Italian Symposium on Advanced Database Systems (SEBD'07), pages 491-494, Torre Canne di Fasano (BR), Italy (2007)
David and Euzenat, 2008	David, J., Euzenat, J.: Comparison between ontology distances (preliminary results). In: 7th International Conference on the Semantic Web (ISWC'08), pages 245-260, Karlsruhe, Germany (2008)

Davies et al., 2006 Davies, J., Studer, R., and Warren, P.: Semantic Web Technologies: Trends and Research in Ontology-based Systems. John Wiley & Sons Ltd (2006)

de Laborda and Conrad, 2005 de Laborda, C. P. and Conrad, S. Relational.OWL A Data and Schema Representation Format Based on OWL. In: 2nd Asia-Pacific Conf. on Conceptual Modelling (APCCM'05), Volume 43 of CRPIT, pages 89-96, Newcastle, Australia (2005)

Diestel, 2005 Diestel, R.: Graph Theory. 3rd Edition, Springer-Verlag, Heidelberg (2005)

Do and Rahm, 2002 Do, H. H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: 28th International Conference on Very Large Data Bases (VLDB), ACM Press, Hong Kong. pages 610-621 (2002)

Doan et al., 2003 Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A.: Learning to match ontologies on the semantic web. In: VLDB Journal 12, 4, 303-319 (2003)

Doan and Halevy, 2005 Doan, A.-H., Halevy, A.: Semantic integration research in the database community: A brief survey. In: AI Magazine, 26(1):83–94. Special issue on Semantic integration. (2005)

Dou et al., 2005 Dou, D., McDermott, D., and Qi, P.: Ontology translation on the semantic web. In: Journal on Data Semantics, II:35-57 (2005)

Doulkeridis et al., 2006 Doulkeridis, C., Nørvag, K., and Vazirgiannis, M.: Scalable Semantic Overlay Generation for P2P-Based Digital Libraries. In: 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'06), pages 26-38, Alicante, Spain (2006)

Doval and O'Mahony, 2003 Doval, D., O'Mahony, D.: Overlay Networks: A Scalable Alternative for P2P. In: IEEE Internet Computing, Volume 07, No. 4, pages 79-82 (2003)

Eclipse, 2009 The Eclipse Integrated Development Environment (IDE), http://www.eclipse.org/ (2009)

Ehrig et al., 2005 Ehrig, M., Haase, P., Hefke, M., and Stojanovic, N.: Similarity for ontologies – a comprehensive framework. In: 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy (ECIS'05), Regensburg, Germany (2005)

Ehrig, 2007 Ehrig, M.: Ontology alignment: bridging the semantic gap. Semantic web and beyond: computing for human experience. Springer, New-York (NY US) (2007)

Ester et al., 1996 Ester, M., Kriegel, H.-P., Sander, J., and Xu, X.: A density-based algorithm for discovering clusters in large spatial data sets with noise. In: 2nd Int. Conf. on Knowledge Discovery and Data Mining, pages 226-231, Portland, USA (1996)

Euzenat, 2001 Euzenat, J.: Towards a principled approach to semantic interoperability. In: IJCAI Workshop on Ontologies and Information Sharing, pages 19-25, Seattle, USA (2001)

Euzenat, 2004 Euzenat, J.: An API for ontology alignment. In: 3rd International Semantic Web Conference (ISWC), volume 3298 of Lecture notes in computer science, pages 698-712, Hiroshima, Japan (2004)

Euzenat and Shvaiko, 2007	Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag Berlin Heidelberg (2007)
Faye et al., 2007	Faye, D., Nachouki, G., and Valduriez, P. Semantic Query Routing in SenPeer, a P2P Data Management System. In: 18th International Conference on Database and Expert Systems Applications, pages 365-374, Regensburg, Germany (2007)
Fensel, 2004	Fensel, D.: Ontologies: a silver bullet for knowledge management and electronic commerce. Springer, Heidelberg (DE), 2nd edition (2004)
Fiorano, 2003	Fiorano Software: Super-Peer Architectures for Distributed Computing. White Paper, Fiorano Software, Inc. Available at http://www.fiorano.com/docs/superpeer.pdf (2003)
Fisher, 1987	Fisher, D. H.: Knowledge acquisition via incremental conceptual clustering. In: Machine Learning, 2(2):139-172 (1987)
Fisher et al., 1992	Fisher, D. H., Xu, L., and Zard, N.: Ordering effects in clustering. In: 9th International Conference on Machine Learning, pages 163-168, Aberdeen, Scotland (1992)
Fowlkes and Mallows, 1983	Fowlkes, E., Mallows, C.: A method for comparing two hierarchical clusterings. In: Journal of the American Statistical Association, 78 (1983)
Freeman, 1979	Freeman, L. C.: Centrality in Networks: I. Conceptual clarification. Social Networks 1, 215-39 (1979)
Freenet, 2009	The Free Network Project. http://freenetproject.org/ (2009)
Fuxman et al., 2006	Fuxman, A., Hernández, M.A., Ho, H., Miller, R, Papotti, P., and Popa, L.: Nested Mappings: Schema Mapping Reloaded. In: VLDB 2006 Conference, pages 67-78, Seoul, Korea (2006)
Gan et al., 2007	Gan, G., Ma, C., Wu, J.: Data Clustering: Theory, Algorithms, and Applications, ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA (2007), Society for Industrial and Applied Mathematics (2007)
Ganti et al., 1999	Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A., and French, J.: Clustering Large Datasets in Arbitrary Metric Spaces. In: 15th ICDE Conf., pages 502-511, Sydney, Australia (1999)
Gennari et al., 1989	Gennari, J., Langley, P., and Fisher, D.: Models of incremental concept formation. In: Journal of Artificial Intelligence, Volume 40, pages 11-61 (1989)
Ghawi and Cullot, 2007	Ghawi, R., and Cullot, N.: Database-to-Ontology Mapping Generation for Semantic Interoperability. In: 3rd International Workshop on Database Interoperability (InterDB'07), held in conjunction with VLDB 2007, Vienna, Austria (2007)
Giunchiglia and Zaihrayeu, 2002	Giunchiglia, F., Zaihrayeu, I.: Making peer databases interact – a vision for an architecture supporting data coordination. In: 6th Int. Workshop on Cooperative Information Agents (CIA), pages 18-35, Madrid, Spain (2002)
Giunchiglia and Shvaiko, 2003	Giunchiglia, F., Shvaiko, P.: Semantic matching. The Knowledge Engineering Review, 18(3):265-280 (2003)
Giunchiglia et al., 2004	Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: S-Match: an algorithm and an implementation of semantic matching. In: 1st European Semantic Web Symposium (ESWS), Volume 3053 of LNCS, pages 61-75, Hersounisous, Greece (2004)

Giunchiglia et al., 2005	Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: Semantic schema matching. In: 13th Int. Conference on Cooperative Information Systems (CoopIS), Volume 3761 of LNCS, pages 347-365, Agia Napa, Cyprus (2005)
Giunchiglia et al., 2006	Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: Discovering missing background knowledge in ontology matching. In: 16th European Conference on Artificial Intelligence (ECAI'06), pages 382-386, Riva del Garda, Italy (2006)
Giunchiglia et al., 2007	Giunchiglia, F., Yatskevich, M., and Shvaiko, P.: Semantic matching: Algorithms and implementation. In: Journal on Data Semantics, IX (2007)
Gruber, 1993	Gruber, T. R.: A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):199-220 (1993)
Guarino, 1998	Guarino, N.: Formal Ontology and Information Systems. In: 1st International Conference on Formal Ontologies in Information Systems, pages 3-15, Trento, Italy (1998)
Haase and Stojanovic, 2005	Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: 2nd European Semantic Web Conference (ESWC'05), Volume 3532 of LNCS, pages 182-197 (2005)
Hai, 2005	Hai, D. H.: Schema Matching and Mapping-Based Data Integration. Ph.D. Thesis. University of Leipzig, Germany (2005)
Halevy et al., 2003a	Halevy, A. Y., Ives, Z. G., Mork, P. and Tatarinov, I.: Piazza: Data Management Infrastructure for Semantic Web Applications. In: World Wide Web Conference, pages 556-567 (2003)
Halevy et al., 2003b	Halevy, A. Y., Ives, Z., Suciu, D., and Tatarinov, I.: Schema Mediation in Peer Data Management Systems. In: International Conference on Data Engineering (ICDE'03), pages 505-516, Bangalore, India (2003)
Halevy et al., 2006	Halevy, A., Rajaraman, A., and Ordille, J.: Data Integration: The Teenage Years. In: VLDB'06, pages 9-16 (2006)
Halkidi et al., 2001	Halkidi, M., Batistakis, Y., and Vazirgiannis, M.: On Clustering Validation Techniques. In: Journal of Intelligent Information System, 17(2):107-145 (2001)
Hammouda and Kamel, 2007	Hammouda, K. M., Kamel, M. S.: HP2PC: Scalable Hierarchically-Distributed Peer-to-Peer Clustering. In: 7th SIAM International Conference on Data Mining, Minneapolis, Minnesota, USA (2007)
Hartigan, 1975	Hartigan, J. A.: Clustering Algorithms. John Wiley and Sons, Inc., New York, NY (1975)
Hartigan and Wong, 1979	Hartigan, J., Wong, M.: A k-means clustering algorithm. In: Journal of Applied Statistics, Volume 28, pages 100-108 (1979)
Heese et al., 2005	Heese, R., Herschel, S., Naumann, F., Roth, A.: Self-Extending Peer Data Management. In: GI-Fachtagung fur Datenbanksysteme in Business, Technologie und Web (BTW'05), Karlsruhe, Germany (2005)

Hose et al., 2007 Hose, K., Lemke, C., Quasebarth, J., Sattler, K.-U.: SmurfPDMS: A Platform for Query Processing in Large-Scale PDMS. In: Business, Technology, and Web (BTW'07), pages 621-624, Aachen, Germany (2007)

Hu et al., 2006 Hu, B., Kalfoglou, Y., Alani, H., Dupplaw, D., Lewis, P., and Shadbolt, N.: Semantic Metrics. In: 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW'06), Vol. 4248 of LNCS, pages 166-181, Prague, Czech Republic (2006)

Hu and Qu, 2007 Hu, W., Qu, Y.: Discovering Simple Mappings Between Relational Database Schemas and Ontologies. In: 6th International Semantic Web Conference (ISWC'07), 4825:225-238, Busan, South Korea, (2007)

Hu and Qu, 2008 Hu, W., Qu, Y.: Falcon-AO: a practical ontology matching system. In: Journal of Web Semantics: Science, Services and Agents on the World Wide Web archive. Volume 6, Issue 3, pages 237-239 (2008)

Jain et al., 1999 Jain, A. K., Murty, M. N., and Flynn, P. J.: Data clustering: a review. In: ACM Computing Survey, 31(3):264-323 (1999)

Java, 2009 Java, Sun Developer Network, http://java.sun.com/ (2009)

Jena, 2009 Jena, a Semantic Web Framework for Java, http://jena.sourceforge.net/ (2009)

Kantere et al., 2003 Kantere, V., Kiringa, I., Mylopoulos, J., Kementsietsidis, A., and Arenas, M.: Coordinating Peer Databases using ECA Rules. In: International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P'03), Volume 2944 of LNCS, pages 108-122, Berlin, Germany (2003)

Kantere et al., 2008 Kantere, V., Tsoumakos, D., and Sellis, T.: A Framework for Semantic Grouping in P2P Databases. In: The Information Systems Journal, Volume 33, Issue 7-8, pages 611-636 (2008)

Kantere et al., 2009 Kantere, V., Tsoumakos, D., Sellis, T., and Roussopoulos, N.: GrouPeer: Dynamic clustering of P2P databases. In: The Information Systems Journal, Volume 34, Issue 1, pages 62-86 (2009)

Karger et al., 1997 Karger, D., Lehman, E., Leighton, F., Levine, M., Lewin, D., and Panigrahy, R.: Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In: 29th Annual ACM Symposium on Theory of Computing, pages 654-663, El Paso, Texas, USA (1997)

Kashyap and Sheth, 1996 Kashyap, V., Sheth, A.: Semantic and schematic similarities between database objects: a context-based approach. In: The VLDB Journal, 5(4):276-304 (1996)

Kashyap and Sheth, 1998 Kashyap, V., Sheth, A.: Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. In: Michael Papazoglou and Gunter Schlageter, editors, Cooperative information systems, pages 139-178. Academic Press, New York, USA (1998)

Katchaounov, 2003 Katchaounov, T.: Query Processing for Peer Mediator Databases. PhD thesis, Department of Information Technology, Uppsala University, Sweden (2003)

Konstantinidis et al., 2008	Konstantinidis, G., Flouris, G., Antoniou, G., Christophides, V.: A Formal Approach for RDF/S Ontology Evolution. In: 18th European Conference on Artificial Intelligence (ECAI'08), pages 405-409, Patras, Greece (2008)
Kotsiantis and Pintelas, 2004	Kotsiantis, S. B. and Pintelas, P. E.: Recent advances in clustering: a brief survey. In: WSEAS Transactions on Information Science and Applications, Volume 1, Number 1, pages 73-81 (2004)
Larson et al., 1989	Larson, J., Navathe, S., and Elmasri, R.: A theory of attributed equivalence in databases with application to schema integration. In: IEEE Transactions on Software Engineering, 15(4):449–463 (1989)
Leibowitz et al., 2003	Leibowitz, N., Ripeanu, M., and Wierzbicki, A.: Deconstructing the KaZaA Network. In: 3rd IEEE Workshop on Internet Applications (WIAPP'03), San Jose, CA, USA (2003)
Lenzerini, 2004	Lenzerini, M.: Principles of P2P Data Integration. In: 3rd International Workshop on Data Integration over the Web (DIWeb'04), pages 7-21, Riga, Latvia (2004)
Li and Vuong, 2005	Li, J., Vuong, S.: Ontology-Based Clustering and Routing in Peer-to-Peer Networks. In: 6th Int. Conference on Parallel and Distributed Computing, Applications and Technologies, pages 791-795, Dalian, China (2005)
Li and Vuong, 2007	Li, J., and Vuong, S.: OntSum: A Semantic Query Routing Scheme in P2P Networks Based on Concise Ontology Indexing. In: 21st International Conference on Advanced Networking and Applications, pages 94-101, Niagara Falls, Canada (2007)
Lodi et al., 2008	Lodi, S., Penzo, W., Mandreoli, F., Martoglia, R., and Sassatelli, S.: Semantic Peer, Here are the Neighbors You Want! In: 11th Extending Database Technology (EDBT'08), pages 26-37, Nantes, France. (2008)
Löser et al., 2003	Löser, A., Naumann, F., Siberski, W., Nejdl, W., and Thaden, U.: Semantic Overlay Clusters within Super-Peer Networks. In: International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03), pages 33-47, Berlin, Germany (2003)
Mädche and Staab, 2002	Mädche, A., Staab, S.: Measuring similarity between ontologies. In: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02), Vol. 2473 of LNCS, pages 251-263, Siguenza, Spain (2002)
Madhavan and Halevy, 2003	Madhavan, J., Halevy, A. Y.: Composing mappings among data sources. In: 29th VLDB Conference, pages 572-583, Berlin, Germany (2003)
Mandreoli et al., 2006a	Mandreoli, F., Martoglia, R., Penzo, W., and Sassatelli, S.: Semantic Query Routing Experiences in a PDMS. In: 3rd Italian Semantic Web Workshop (SWAP'06), Pisa, Italy (2006)
Mandreoli et al., 2006b	Mandreoli, F., Martoglia, R., Penzo, W., and Sassatelli, S.: SRI: Exploiting Semantic Information for Effective Query Routing in a PDMS. In: WIDM, (2006)

Mandreoli et al., 2007	Mandreoli, F., Martoglia, R., Penzo, W., Sassatelli, S., and Villani, G.: SUNRISE: Exploring PDMS Networks with Semantic Routing Indexes. In: 4th European Semantic Web Conference (ESWC'07), Innsbruck, Austria (2007)
Melnik et al., 2003	Melnik, S., Rahm, E., and Bernstein, P.: Rondo: A programming platform for model management. In: 22nd International Conference on Management of Data (SIGMOD), pages 193-204, San Diego, USA (2003)
Mika, 2007	Mika, P.: Social Networks and the Semantic Web. Springer-Verlag New York, Inc. (2007)
Miller, 1995	Miller, A. G.: WordNet: A lexical database for English. In: Communications of the ACM, 38(11):39-41 (1995)
Milojicic et al., 2002	Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-Peer Computing. In: Technical Report HPL-2002-57, HP Labs. Available at http://www.hpl.hp.com/techreports/2002/HPL-2002-57R1.pdf (2002)
Mitra et al., 1999	Mitra, P., Wiederhold, G., and Jannink, J.: Semi-automatic integration of knowledge sources. In: 2nd Int. Conference on Information Fusion, pages 572-581, Sunnyvale, USA (1999)
Modica et al., 2001	Modica, G., Gal, A., and Jamil, H.: The use of machine-generated ontologies in dynamic information seeking. In: 9th Int. Conf. on Cooperative Information Systems (CoopIS), Vol. 2172 of LNCS, pages 433-448, Trento, Italy (2001)
Montanelli and Castano, 2008	Montanelli, S. and Castano, S.: Semantically routing queries in peer-based systems: the H-Link approach. In: Knowledge Engineering Review 23(1): 51-72 (2008)
Moody and Filtman, 1999	Moody, D., Filtman, A.: A Methodology for Clustering Entity Relationship Models – A Human Information Processing Approach. In: 18th Int. Conference on Conceptual Modeling, pages 114-130, Paris, France (1999)
Nejdl et al., 2002	Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., and Risch, T.: Edutella: A P2P networking infrastructure based on RDF. In: 11th Int. World Wide Web Conference, pages 604-615, Honolulu, USA (2002)
Ng et al., 2003	Ng, W. S., Ooi, B. C., Tan, K.-L., and Zhou, A.: PeerDB: A P2P-based System for Distributed Data Sharing. In: 19th Int. Conf. on Data Engineering, pages 633-644, Istanbul, Turkey (2003)
Noy and Musen, 2000	Noy, N. F., Musen, M. A.: Prompt: Algorithm and Tool for Automated Ontology Merging and Alignment. In: 17th Nat. Conf. on Artificial Intelligence, Austin, Texas, USA (2000)
Noy and Musen, 2003	Noy, N., Musen, M.: The PROMPT suite: interactive tools for ontology merging and mapping. In: International Journal of Human-Computer Studies, 59(6):983-1024 (2003)
Noy, 2004a	Noy, N. F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec., 33(4):65-70 (2004)
Noy, 2004b	Noy, N. F.: Tools for mapping and merging ontologies. In: Steffen Staab and Rudi Studer, editors, Handbook on ontologies, chapter 18, pages 365-384. Springer Verlag, Berlin, Germany (2004)

Nyulas et al., 2007 — Nyulas, C., O'Connor, M., and Tu, S.: DataMaster - a Plug-in for Importing Schemas and Data from Relational Databases into Protégé. In: 10th Int. Protégé Conf., Budapest, Hungary (2007)

Oaks and Wong, 2004 — Oaks, S., Wong, H.: Java Threads. O'Reilly, 3rd Edition (2004)

Ooi et al., 2003 — Ooi, B. C., Tan, K.-L., Zhou, A., Goh, C. H., Li, Y., Liau, C. Y., Ling, B., Ng, W. S., Shu, Y., Wang, X., and Zhang, M.: PeerDB: Peering into Personal Databases. In: ACM SIGMOD International Conference on Management of Data, page 659, San Diego, California, USA (2003)

OWL API, 2009 — The OWL Application Programming Interface (API), http://owlapi.sourceforge.net/ (2009)

Palopoli et al., 2003 — Palopoli, L., Terracina, G., and Ursino, D.: DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. In: Software–Practice and Experience, 33(9):847-884 (2003)

Parent and Spaccapietra, 1998 — Parent, C., Spaccapietra, S.: Issues and approaches of database integration. In: Communications of the ACM, 41(5):166–178 (1998)

Pires et al., 2006 — Pires, C. E. S., Lóscio, B. F., and Salgado, A. C.: Data Management in P2P Systems. In: 21st Brazilian Symposium on Databases, page 310, Florianópolis, SC, Brazil (2006)

Pires, 2007a — Pires, C. E. S.: Semantic-based Connectivity in a Peer Data Management System. In: 6th Workshop of Thesis and Dissertation on Data Base, held in conjunction with the 22th Brazilian Symposium on Data Bases (SBBD'08), João Pessoa, PB, Brazil (2007)

Pires, 2007b — Pires, C. E. S.: Um Sistema P2P de Gerenciamento de Dados com Conectividade Baseada em Semântica. Thesis Proposal. Center for Informatics, Federal University of Pernambuco, Recife, PE, Brazil (2007)

Pires et al., 2008 — Pires, C. E. S., Souza, D., Lóscio, B. F., and Salgado, A. C.: An Ontology-based Approach for Data Management in a P2P System. SPEED Project, Technical Report, No. 2, Center for Informatics, Federal University of Pernambuco (2008)

Pires et al., 2009a — Pires, C. E. S., Alencar, V. B., Kedad, Z., and Salgado, A. C.: Building Ontology Summaries for PDMS. Submitted to the 28th Conf. on Conceptual Modeling, Gramado, Brazil (2009)

Pires et al., 2009b — Pires, C. E. S., Souza, D., Pachêco, T., and Salgado, A. C.: A Semantic-based Ontology Matching Process for PDMS. To appear in 2nd International Conference on Data Management in Grid and P2P Systems (Globe'09), Linz, Austria (2009)

Pires et al., 2009c — Pires, C. E. S., Souza, D., Kedad, Z., Bouzeghoub, M., and Salgado, A. C.: Using Semantics in Peer Data Management Systems. To appear in Colloquium of Computation: Brazil / INRIA, Cooperations, Advances and Challenges (Colibri'09), Bento Gonçalves, RS, Brazil (2009)

Prefuse, 2009 — The Prefuse Visualization Toolkit, http://www.prefuse.org/ (2009)

Rahm and Bernstein, 2001 Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. In: The VLDB Journal, 10(4):334–350 (2001)

Ramanathan et al., 2002 Ramanathan, M. K., Kalogeraki, V., and Pruyne, J.: Finding good peers in peer-to-peer networks. In: 16th International Parallel and Distributed Processing Symposium (IPDPS'02), page 24-31, Florida, USA (2002)

Ramaswamy et al., 2003 Ramaswamy, L., Gedik, B., and Liu, L.: Connectivity Based Node Clustering in Decentralized Peer-to-Peer Networks. In: 3rd International Conference on Peer-to-Peer Computing (P2P'03), pages 66-73, Linköping, Sweden.(2003)

Ratnasamy et al., 2001 Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S.: A Scalable Content-Addressable Network. In: ACM SIGCOMM, pages 161-172, San Diego, California, USA (2001)

Reynaud and Safar, 2007 Reynaud, C., Safar, B.: Exploiting WordNet as Background Knowledge. In: International ISWC'07 Ontology Matching (OM-07) Workshop, Busan, Korea (2007)

Rijsbergen, 1979 Rijsbergen, C. J.: Information Retrieval, 2nd Ed. Stoneham, MA: Butterworths, (1979) http://www.dcs.gla.ac.uk/Keith/Preface.html.

Rocha et al., 2004 Rocha, J., Domingues, M., Callado, A., Souto, E., Silvestre, G., Kamienski, C., Sadok, D.: Peer-to-Peer: Computação Colaborativa na Internet. In: Mini-curso 22º Simpósio Brasileiro de Redes de Computadores (SBRC'04), Gramado, RS, Brazil (2004)

Rousse and Berman, 2006 Rousse, C., Berman, S.: A Scalable P2P Database System with Semi-Automated Schema Matching. In: 26th IEEE Int. Conference Workshops on Distributed Computing Systems (ICDCSW'06), page 78, Lisboa, Portugal (2006)

Rousseeuw, 1987 Rousseeuw, P. J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. In: Journal of Computational and Applied Mathematics, Volume 20, pages 53-65 (1987)

Rousset et al., 2006 Rousset, M.-C., Adjiman, P., Chatalic, P., Goasdoué, F., and Simon, L.: Somewhere in the Semantic Web. In: 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SofSem), Volume 3831 of LNCS, pages 84-99, Merin, Czech Republic (2006)

Ruzzi, 2004 Ruzzi, M.: Data Integration: state of the art, new issues and research plan. Technical Report (2004)

Sabou et al., 2006 Sabou, M., d'Aquin, M., and Motta, E.: Using the Semantic Web as Background Knowledge for Ontology Mapping. In: 1st Int. Workshop on Ontology Matching (OM'06), collocated with the 5th International Semantic Web Conference (ISWC'06), Athens, Georgia, USA (2006)

Schlicht and Stuckenschmidt, 2008 Schlicht, A., Stuckenschmidt, H.: A Flexible Partitioning Tool for Large Ontologies. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Volume 1, pages 482-488, Sydney, Australia (2008)

Shvaiko and Euzenat, 2008	Shvaiko, P. and Euzenat, J.: Ten Challenges for Ontology Matching. In: 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE), pages 1164-1182, Monterrey, Mexico (2008)
Smith et al., 2004	Smith, M., Welty, C., and McGuinness, D.: OWL Web Ontology Language Guide. Recommendation, W3C, (2004)
Souza, 2007	Souza, D.: Reformulação de Consultas Baseada em Semântica para PDMS. Thesis Proposal. Federal University of Pernambuco, Recife, PE, Brazil (2007)
Souza , 2009	Souza, D.: Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments. Ph.D. Thesis. Center for Informatics, Federal University of Pernambuco, Recife, PE, Brazil (2009)
Souza et al., 2009	Souza D., Arruda, T., Salgado, A. C., Tedesco, P., and Kedad, Z.: Using Semantics to Enhance Query Reformulation in Dynamic Environments. To appear in the 13th East European Conference on Advances in Databases and Information Systems (ADBIS'09), Riga, Latvia (2009)
Staab and Studer, 2004	Staab, S., Studer, R.: Handbook on ontologies. International handbooks on information systems. Springer-Verlag, Berlin, Germany (2004)
Staab and Stuckenschmidt, 2006	Staab, S., Stuckenschmidt, H.: Semantic Web and Peer-to-Peer. Springer-Verlag, Berlin Heidelberg (2006)
Steiner and Biersack, 2005	Steiner, M., Biersack, E. W.: DDC: a dynamic and distributed clustering algorithm for networked virtual environments based on P2P networks. In: 1st ACM/e-NEXT International Conference on Future Networking Technologies, pages 288-289, Toulouse, France (2005)
Stoica et al., 2001	Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: ACM SIGCOMM, pages 149-160, San Diego, California, USA (2001)
Stuckenschmidt and Klein, 2004	Stuckenschmidt, H, Klein, M.: Structure-Based Partitioning of Large Concept Hierarchies. In: 3rd Int. Semantic Web Conf. (ISWC'04), pages 289-303, Hiroshima, Japan (2004)
Stumme and Mädche, 2001	Stumme, G., Mädche, A.: FCA-Merge: Bottom-up merging of ontologies. In: 17th International Joint Conference on Artificial Intelligence (IJCAI), pages 225-234, Seattle, USA (2001)
Sung et al., 2005	Sung, L. G. A., Ahmed, N., Blanco, R., Li, H, Soliman, M. A., and Hadaller, D.: A Survey of Data Management in Peer-to-Peer Systems. School of Computer Science, University of Waterloo (2005)
Tatarinov et al., 2003	Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X., Kadiyska, Y. Miklau, G., and Mork, P.: The Piazza Peer Data Management Project. In: ACM SIGMOD Record, Vol. 32, Issue 3, pages 47-52 (2003)
Tatarinov and Halevy, 2004	Tatarinov, I. and Halevy, A.: Efficient query reformulation in peer data management systems. In: ACM SIGMOD Int. Conf. on Management of Data, pages 539-550, Paris, France (2004)

Theodoridis and Koutroumbas, 2003	Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic Press, 2nd Edition (2003)
Toledo, 2005	Toledo, M. D. G.: A Comparison in Cluster Validation Techniques. In: Master Dissertation in Mathematics (Statistics), University of Puerto Rico (2005)
Uschold and Gruninger, 2004	Uschold, M., Gruninger, M.: Ontologies and Semantics for Seamless Connectivity. In: SIGMOD Record, Vol. 33, No. 4, pages 58-64 (2004)
Valduriez and Pacitti, 2004	Valduriez, P., and Pacitti, E.: Data Management in Large-Scale P2P Systems. In: Int. Conference on High Performance Computing for Computational Science (VecPar'04), pages 104-118, Valencia, Spain (2004)
Vdovjak et al., 2006	Vdovjak, R., Houben, G-J., Stuckenschmidt, H., and Aerts, A.: RDF and Traditional Query Architectures. In: Peer-to-Peer and Semantic Web: Decentralized Management and Exchange of Knowledge and Information, Springer-Verlag, pages 41-58 (2006)
Wache et al., 2001	Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S.: Ontology-based integration of information – a survey of existing approaches. In: IJCAI Workshop on Ontologies and Information Sharing, pages 108-117, Seattle, USA (2001)
Walkerdine et al., 2002	Walkerdine, J., Melville, L., and Sommerville, I.: Dependability properties of P2P architectures". In: 2nd International Conference on Peer to Peer Computing (P2P'02), pages 173-174, Linkoping, Sweden (2002)
Wang et al., 1997	Wang, W., Yang, J., and Muntz, R.: STING: A Statistical Information Grid Approach to Spatial Data Mining. In: 23rd VLDB Conference, pages 186-195, Athens, Greece (1997)
Wicaksana and Yétongnon, 2006	Wicaksana, W. S., Yétongnon, K.: A Peer-to-Peer Based Semantic Agreement Approach for Information System Interoperability. In: International Workshop on Ontology Matching collocated with the 5th International Semantic Web Conference, pages 216-220, Georgia, USA (2006)
Wiederhold, 1992	Wiederhold, G.: Mediators in the Architecture of Future Information Systems. In: IEEE Computer, 25(3):38-49 (1992)
Xiao, 2006	Xiao, H.: Query Processing for Heterogeneous Data Integration using Ontologies. Ph.D. Thesis. University of Illinois, Chicago, USA (2006)
Xu and Srimani, 2005	Xu, Z., Srimani, P. K.: Self-stabilizing Publish/Subscribe Protocol for P2P Networks. In: 7th Int. Workshop on Distributed Computing (IWDC'05), Volume 3741 of LNCS, pages 129-140, Kharagpur, India (2005)
Xu and Wunsch, 2005	Xu, R., Wunsch, D.: Survey of Clustering Algorithms. In: IEEE Transactions on Neural Networks, Volume 16, Number 3, pages 645-678 (2005)
Yang and Garcia-Molina, 2003	Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network. In: 19th International Conference on Data Engineering (ICDE'03), pages 49-60, Bangalore, India (2003)

Young, 2004	Young, W. A.: Evaluation of Peer-to-Peer Database Solutions, Course Project available at http://www.tonyyoung.ca/cs654paper.pdf (2004)
Yu and Jagadish, 2006	Yu, C., Jagadish, H. V.: Schema Summarization. In: 32nd Int. Conference on Very Large Data Bases, pages 319-330, Seoul, Korea. (2006)
Zhang et al., 1997	Zhang, T., Ramakrishnan, R., and Linvy, M.: BIRCH: An efficient data clustering method for very large data sets. In: Data Mining and Knowledge Discovery, 1(2): 141-182 (1997)
Zhang et al., 2007	Zhang, X., Cheng, G., and Qu, Y.: Ontology Summarization Based on RDF Sentence Graph. In: 16th International Conference on World Wide Web, pages 707-716, Alberta, Canada (2007)
Zhang et al., 2008	Zhang, R., Wang, Y., and Wang, J.: Research on Ontology Matching Approach in Semantic Web. In: International Conference on Internet Computing in Science and Engineering (ICICSE'08), pages 254-257, Harbin, China (2008)
Zhdanova and Shvaiko, 2006	Zhdanova, A. V., and Shvaiko, P.: Community-Driven Ontology Matching. In: 3rd European Semantic Web Conference (ESWC'06), pages 34-49, Budva, Montenegro (2006)
Zhuang et al., 2004	Zhuang, Z., Liu, Y., and Xiao, L.: Dynamic Layer Management in Super-Peer Architectures. In: International Conference on Parallel Processing (ICPP'04), Volume 00, pages 29-36, Montreal, Canada (2004)

APPENDIX A: EDUCATION ONTOLOGIES

A1. Ontology: *UnivBench.owl*

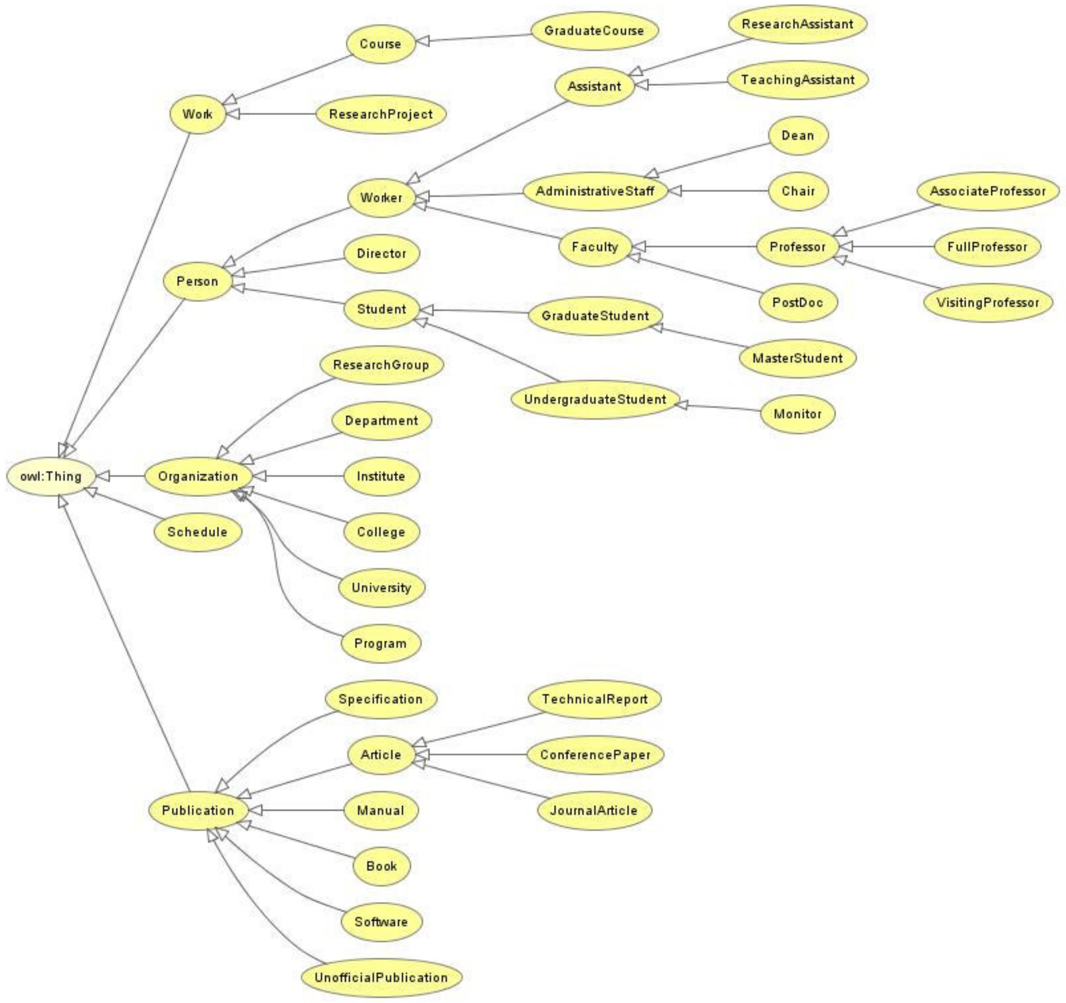


Figure A.1. The *UnivBench* ontology. Some statistics: #concepts=42 and #properties=28.

A2. Ontology: *Semiport.owl*

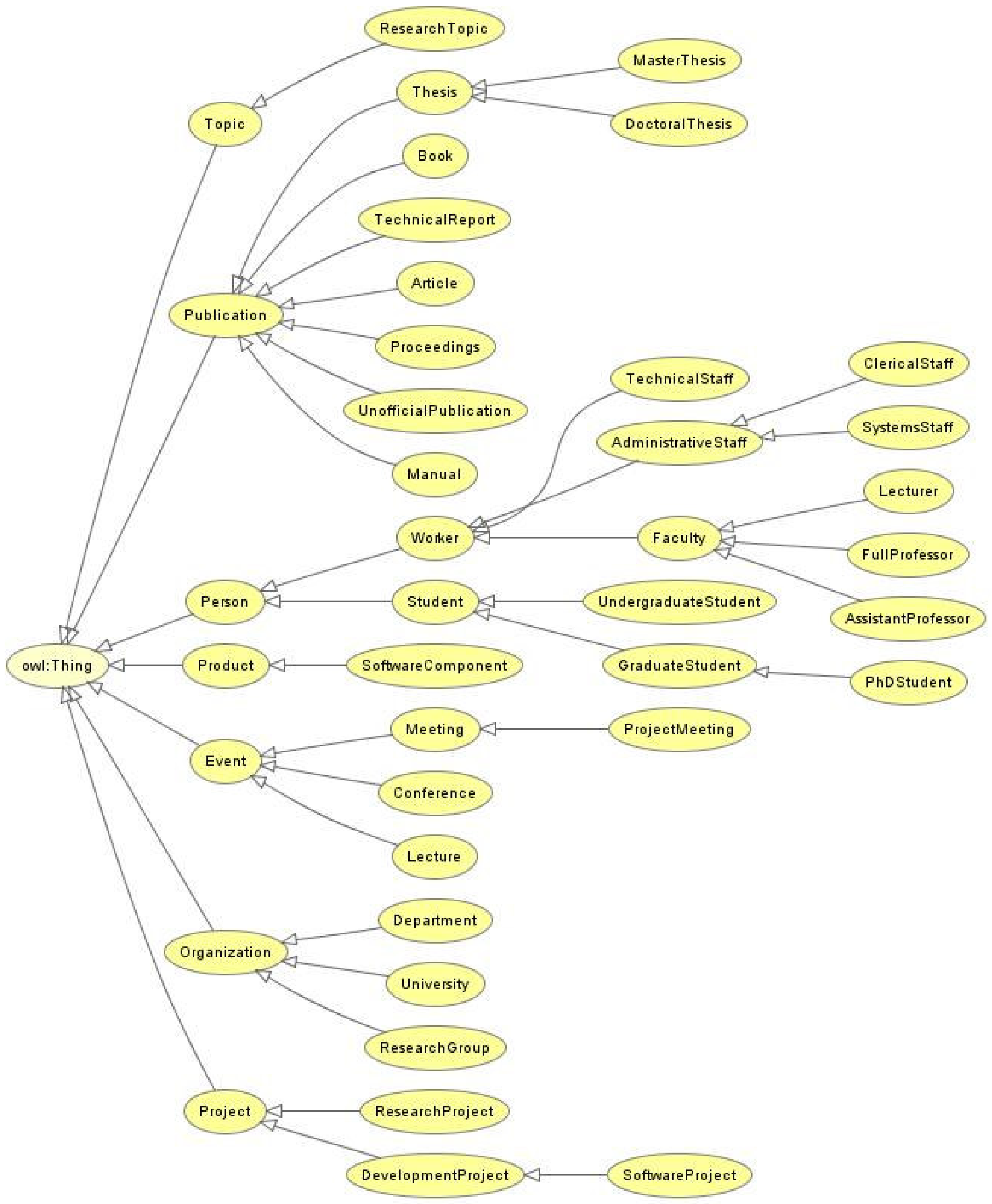


Figure A.2. The *Semiport* ontology. Some statistics: #concepts=41 and #properties=40.

A3. Ontology: *UnivCMOCS.owl*

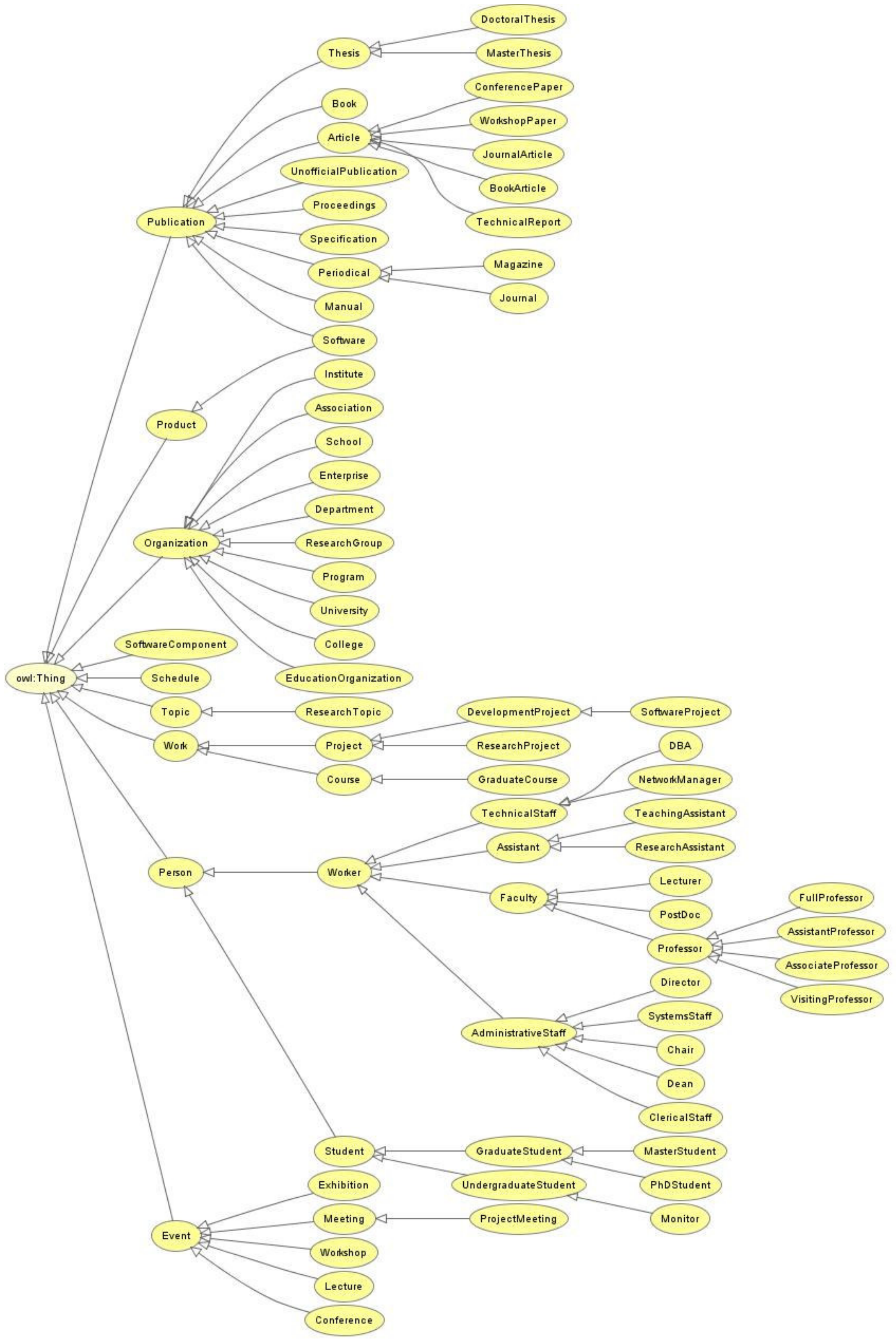


Figure A.3. The *UnivCSCMO* ontology. Some Statistics: #concepts=77 and #properties=78.

APPENDIX B: ONTOLOGY ALIGNMENTS

B1. Alignments (A_{ij}) between *Semiport.owl* and *UnivBench.owl*

Id	*Semiport* concept	*UnivBench* concept	Semantic relationship	Similarity value
1	Student	Student	isEquivalentTo	1.00
2	ResearchProject	ResearchProject	isEquivalentTo	1.00
3	Person	Person	isEquivalentTo	1.00
4	DevelopmentProject	ResearchProject	isCloseTo	0.75
5	Product	Software	isSuperConceptOf	0.68
6	ResearchGroup	ResearchGroup	isEquivalentTo	1.00
7	Project	ResearchProject	isSuperConceptOf	0.84
8	UnofficialPublication	UnofficialPublication	isEquivalentTo	1.00
9	TechnicalStaff	Worker	isSubConceptOf	0.68
10	ClericalStaff	ClericalStaff	isEquivalentTo	1.00
11	Publication	Publication	isEquivalentTo	1.00
12	Worker	Worker	isEquivalentTo	1.00
13	Organization	Organization	isEquivalentTo	1.00
14	SystemsStaff	SystemsStaff	isEquivalentTo	1.00
15	Department	Department	isEquivalentTo	1.00
16	Proceedings	Publication	isSubConceptOf	0.68
17	UndergraduateStudent	UndergraduateStudent	isEquivalentTo	1.00
18	AdministrativeStaff	AdministrativeStaff	isEquivalentTo	1.00
19	Thesis	Publication	isSubConceptOf	0.69
20	TechnicalReport	TechnicalReport	isEquivalentTo	1.00
21	University	University	isEquivalentTo	1.00
22	Article	Article	isEquivalentTo	1.00
23	AssistantProfessor	AssistantProfessor	isEquivalentTo	1.00
24	SoftwareComponent	Software	isPartOf	0.48
25	PhDStudent	GraduateStudent	isSubConceptOf	0.81
26	Lecturer	Lecturer	isEquivalentTo	1.00
27	Book	Book	isEquivalentTo	1.00
28	FullProfessor	FullProfessor	isEquivalentTo	1.00
29	Faculty	Faculty	isEquivalentTo	1.00
30	Manual	Manual	isEquivalentTo	1.00
31	SoftwareProject	ResearchProject	isCloseTo	0.75
32	GraduateStudent	GraduateStudent	isEquivalentTo	1.00
			Sum	**29.36**

B2. Alignments (A_{ji}) between *Semiport.owl* and *UnivBench.owl*

Id	*UnivBench* concept	*Semiport* concept	Semantic relationship	Similarity value
1	Institute	Organization	isSuperConceptOf	0.68
2	Book	Book	isEquivalentTo	1.00
3	AdministrativeStaff	AdministrativeStaff	isEquivalentTo	1.00
4	FullProfessor	FullProfessor	isEquivalentTo	1.00
5	Article	Article	isEquivalentTo	1.00
6	UndergraduateStudent	UndergraduateStudent	isEquivalentTo	1.00
7	Work	Project	isSuperConceptOf	0.68
8	Student	Student	isEquivalentTo	1.00
9	GraduateStudent	GraduateStudent	isEquivalentTo	1.00
10	Department	Department	isEquivalentTo	1.00
11	Organization	Organization	isEquivalentTo	1.00
12	ResearchProject	ResearchProject	isEquivalentTo	1.00
13	Lecturer	Lecturer	isEquivalentTo	1.00
14	GraduateCourse	GraduateStudent	isWholeOf	0.52
15	VisitingProfessor	AssistantProfessor	isCloseTo	0.75
16	Chair	AdministrativeStaff	isSubConceptOf	0.68
17	Worker	Worker	isEquivalentTo	1.00
18	University	University	isEquivalentTo	1.00
19	Course	GraduateStudent	isWholeOf	0.42
20	Director	AdministrativeStaff	isSubConceptOf	0.68
21	UnofficialPublication	UnofficialPublication	isEquivalentTo	1.00
22	TechnicalReport	TechnicalReport	isEquivalentTo	1.00
23	AssociateProfessor	ResearchProject	isPartOf	0.39
24	JournalArticle	Article	isSubConceptOf	0.84
25	Publication	Publication	isEquivalentTo	1.00
26	Professor	AssistantProfessor	isSuperConceptOf	0.84
27	AssistantProfessor	AssistantProfessor	isEquivalentTo	1.00
28	SystemsStaff	SystemsStaff	isEquivalentTo	1.00
29	Manual	Manual	isEquivalentTo	1.00
30	Dean	AdministrativeStaff	isSubConceptOf	0.68
31	Program	Organization	isSubConceptOf	0.71
32	PostDoc	Faculty	isSubConceptOf	0.68
33	ResearchGroup	ResearchGroup	isEquivalentTo	1.00
34	ConferencePaper	Article	isSubConceptOf	0.68
35	Faculty	Faculty	isEquivalentTo	1.00
36	Specification	Publication	isSubConceptOf	0.68
37	Assistant	Worker	isSubConceptOf	0.68
38	Software	Publication	isSubConceptOf	0.69
39	College	Organization	isSubConceptOf	0.71
40	Person	Person	isEquivalentTo	1.00

41	ClericalStaff	ClericalStaff	isEquivalentTo	1.00
			Sum	**34.99**

B3. Global Similarity Measure between *Semiport.owl* and *UnivBench.owl*

$$Weighted\ Average(Semiport, UnivBench) = \frac{29.36 + 34.99}{41 + 42} = 0.77$$

APPENDIX C: SPEED SIMULATOR

In order to execute a clustering test, first it is necessary to provide the order in which requesting peers will be connected. This can be done through an input file named *input.txt*. In fact, this file contains the name of the local ontologies to be associated with requesting peers. For instance, considering the ontologies depicted in Figure C.1 the first requesting peer is associated with the local ontology *LO05-Education.owl*, the following with *LO02-Education.owl*, and so on. The parameters *cluster threshold* and *neighbor threshold* are informed at the simulator interface.

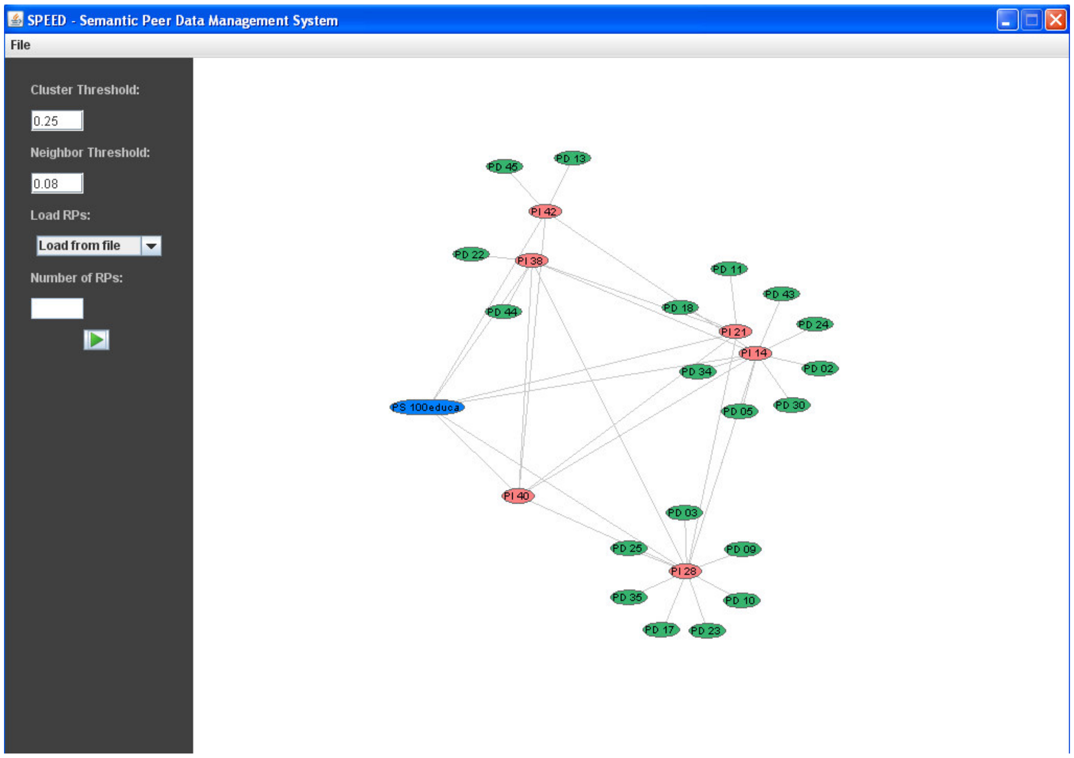
```
LO05-Education.owl
LO02-Education.owl
LO10-Education.owl
LO15-Education.owl
LO01-Education.owl
…
```

Figure C.1. Local ontologies to be associated with requesting peers.

After a successful execution of a simulation test, the network is exhibited in the screen (Figure C.2). For this part of the implementation we have used the graphical visualization tool Prefuse [Prefuse, 2009]. Due to the performance reasons, the interface can be omitted for tests with a high number of requesting peers in which the user is interested only in the clustering result.

The simulator also produces a log file named *output.txt*. The file provides the clustering result and contains information such as: (i) the number of clusters that were created; (ii) the number of peers in each cluster; (iii) the semantic neighbors of each cluster; and (iv) the cluster to which a requesting peer has been associated with. In fact, such information describes the network status after all requesting peers are connected. The file also contains the connection log of each requesting peer. Thus, it is possible to verify all the steps followed by a requesting peer before it joins a current cluster or creates a new one. Figure C.3 depicts the fragment of a log file created after a simulation with 45 requesting peers.

An option to calculate the statistical indices Rand Index, Jaccard Coefficient, Fowlkes-Mallows (FM) Index, Hubert's statistic, and Silhouette indexes is available. For the first four indices, the clustering result produced by the simulator must be compared with another clustering result that can be either provided manually or generated by another cluster algorithm.


SPEED - Semantic Peer Data Management System

Cluster Threshold: 0.25

Neighbor Threshold: 0.08

Load RPs: Load from file

Number of RPs:

Figure C.2. The SPEED's simulator interface.

```
Tue Mar 24 18:18:45 GMT-03:00 2009

RP45 is now connecting...
RP45 is now a Integration Peer with out semantic neighbors
Semantic Index:
<<Cluster: 45>>
  Exhibition(1) Event(1) Conference(1) Workshop(1)
Network:
Domain: education (represented by SP: 100)
   Cluster45(RP45)
…
Network:
Domain: education (represented by SP: 100)
   Cluster45(RP45, RP13, RP36, RP29, RP42)
   Cluster08(RP08, RP20, RP02, RP05, RP06, RP27, RP26, RP16, RP30)
   Cluster44(RP44, RP38, RP39, RP41, RP22, RP33)
   Cluster37(RP37, RP32, RP19, RP40)
   Cluster15(RP15, RP11, RP31, RP21, RP07, RP17, RP18, RP03)
   Cluster24(RP24, RP14, RP34, RP43)
   Cluster28(RP28, RP01, RP23, RP35, RP12, RP04, RP09, RP25, RP10)

Total number of messages: 561
#matchings between OS and LO: 251
#matchings between CLOs: 42
#matchings between CLO and LO: 42
Simulation time: 1161 seconds
External indices: RandIndex=0.942 JaccardCoefficiet=0.646 FMIndex=0.785 Hubbert=0.752
```

Figure C.3. The clustering result associated to a simulation test with 45 requesting peers.

Tese de Doutorado apresentada por **Carlos Eduardo Santos Pires** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Ontology-Based Clustering in a Peer Data Management System"**, orientada pela **Profa. Ana Carolina Brandão Salgado** e aprovada pela Banca Examinadora formada pelos professores:

[illegible]

Prof. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

Patricia Azevedo Tedesco

Profa. Patricia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

[illegible]

Profa. Marta Lima de Queiroz Mattoso
Programa de Engenharia de Sistemas / UFRJ

Ana Maria de C. Moura

Profa. Ana Maria de Carvalho Moura
Instituto Militar de Engenharia - RJ

Bernadette Farias Lóscio

Profa. Bernadette Farias Lóscio
Departamento de Computação / UFC

Visto e permitida a impressão.
Recife, 27 de abril de 2009.

[illegible]

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.