KEx: a Peer-to-Peer solution for Distributed Knowledge Management

M. Bonifacio[1,2], P. Bouquet[1,2], G. Mameli[2], and M. Nori[2]

[1] Dept. of Information and Communication Tech. – University of Trento (Italy)
[2] Istituto per la Ricerca Scientifica e Tecnologica, Trento (Italy)

Abstract. Distributed Knowledge Management is an approach to Knowledge Management based on the principle that the multiplicity (and heterogeneity) of perspectives within complex organizations should not be viewed as an obstacle to knowledge exploitation, but rather as an opportunity that can foster innovation and creativity. Despite a wide agreement on this principle, most current KM systems are based on the idea that all perspectival aspects of knowledge should be eliminated in favor of an objective and general representation of knowledge. In this paper we propose a peer-to-peer architecture (called KEx), which embodies the principle above in a quite straightforward way: (i) each peer (called a K-peer) provides all the services needed to create and organize "local" knowledge from an individual's or a group's perspective, and (ii) social structures and protocols of meaning negotiation are defined to achieve semantic coordination among autonomous peers (e.g., when searching documents from other K-peers).

1 Introduction

Distributed Knowledge Management (DKM), as described in [6], is an approach to KM based on the principle that the multiplicity (and heterogeneity) of perspectives within complex organizations should not be viewed as an obstacle to knowledge exploitation, but rather as an opportunity that can foster innovation and creativity.

The fact that different individuals and communities may have very different perspectives, and that these perspectives affect their representation of the world (and therefore of their work) is widely discussed – and generally accepted – in theoretical research on the nature of knowledge. Knowledge representation in artificial intelligence and cognitive science have produced many theoretical and experimental evidences of the fact that what people know is not a mere collection of facts; indeed, knowledge always presupposes some (typically implicit) interpretation schema, which provide an essential component in sense-making (see, for example, the notions of context [18, 7, 13], mental space [12], partitioned representation [10]); studies on the social nature of knowledge stress the social nature of interpretation schemas, viewed as the outcome of a special kind of "agreement" within a community of knowing (see, for example, the notions of scientific paradigm [16], frame [15]), thought world [11], perspective [3]).

Despite this large convergence, it can be observed that the high level architecture of most current KM systems in fact does not reflect this vision of knowledge (see [5, 6, 4] for a detailed discussion of this claim). The fact is that most KM systems embody the assumption that, to share and exploit knowledge, it is necessary to implement a process of knowledge-extraction-and-refinement, whose aim is to eliminate all subjective and contextual aspects of knowledge, and create an objective and general representation that can then be reused by other people in a variety of situations. Very often, this process is finalized to build a central knowledge base, where knowledge can be accessed via a knowledge portal. This centralized approach – and its underlying objectivist epistemology – is one of the reasons why so many KM systems are deserted by users, who perceive such systems either as irrelevant or oppressive [9].

In this paper we propose a peer-to-peer (P2P) architecture, called KEx, which is coherent with the vision of DKM. Indeed, P2P systems seem particularly suitable to implement the two core principles of DKM, namely the principle of autonomy (communities of knowing should be granted the highest possible degree of semantic autonomy to manage their local knowledge), and the principle of coordination (the collaboration between autonomous communities must be achieved through a process of semantic coordination, rather than through a process of semantic homogenization) [6]. In KEx, each community of knowing (or Knowledge Nodes (KN), as they are called in [4]) is represented by a peer, and the two principles above are implemented in a quite straightforward way: (i) each peer provides all the services needed by a knowledge node to create and organize its own local knowledge (autonomy), and (ii) by defining social structures and protocols of meaning negotiation in order to achieve semantic coordination (e.g., when searching documents from other peers).

The paper goes as follows. In section 2, we describe the main features of KEx, and argue why they provide a useful support to DKM; in 3, we describe its implementation in a peer-to-peer platform called JXTA; finally, we draw some conclusions and future work.

2 KEx: a P2P architecture for DKM

KEx is a P2P system which allows a collection of KNs to search and provide documents on a semantic basis without presupposing a beforehand agreement on how documents should be categorized, or on a common language for representing semantic information within the system. In the following sections, we describe the high-level architecture of KEx, and explain what role each element plays in a DKM vision.

2.1 K-peers

KEx is defined as a collection of peers, called knowledge peers (K-Peers), each of which represents a KN, namely an individual's or a group's perspective on a given body of knowledge. Each K-peer can play two main roles: *provider* and *seeker*. A K-peer acts as a provider when it "publishes" in the system a body of knowledge, together with an explicit perspective on it (called a *context*, e.g. a topic hierarchy used to categorized local documents [8]); a K-peer acts as a seeker when it searches for information by making explicit part of its own perspective, and negotiates it with other K-peers.

Each K-peer has the structure shown in Figure 1. Below we illustrate the main modules and functionalities.

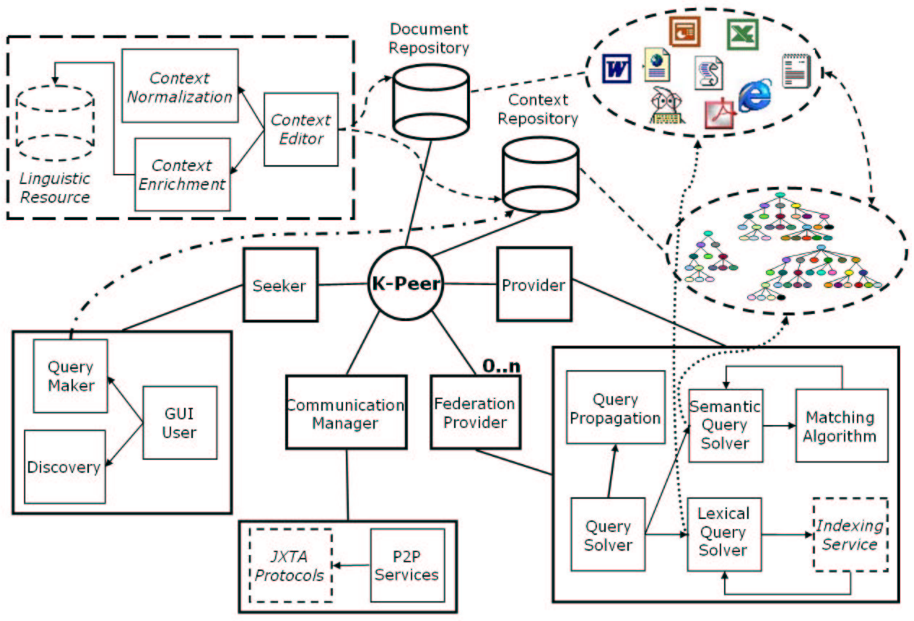

Fig. 1. The KEx's main components

Document Repository. A *Document Repository* is where each KN stores its own local knowledge. We can imagine a private space in which the KN maintains its document and data, possibly using a local semantic schema (e.g., a file-system structure, or a database schema), or a document management system in order to organize and access them.

Context Repository. Following [2], we define a context as a partial and approximate representation of the world from an individual's or a group's perspective. The reason why we adopt this notion of context is that it provides a robust formal framework (called Local Models Semantics [13]) for modeling both contexts and their relationships.

In order to use contexts in KEx, we adopted a web-oriented syntax for contexts, called CTXML. It provides an XML-Schema specification of context for document organization and classification[3].

In KEx, each context plays the role of a category system for organizing and classifying documents, or any other kind of digital information identifiable by a URI, stored in a document repository. Each peer can use more than one context to classify local knowledge; a K-peer's contexts are stored in a *context repository*.

From the standpoint of DKM, contexts are relevant in two distinct senses:

[3] Currently, contexts are trees, whose nodes are labelled with words defined in some name space. Arcs are Is-A, Part-Of or generic relations between nodes. Details can be found in [8].

– on the one hand, they have an important role within each KN, as they provide a dynamic and incremental explicitation of its semantic perspective. Once contexts are reified, they become cognitive artifacts that contribute to the process of perspective making [3], namely the consolidation of a shared view in a KN, continuously subject to revision and internal negotiation among its members;
– on the other hand, contexts offer a simple and direct way for a KN to make public its perspective on the information that that KN can provide. Therefore, as we will see, contexts are an essential tool for semantic coordination among different KN.

It is important to observe that contexts provide only a common syntax for classification structures. Indeed, we could see them as a language for wrapping any classification structure (e.g., like directory systems, databases schemas, web directories). This means that in principle people can continue to work with their preferred document management system, provided it can be wrapped using CTXML.

Context management module. The context management module allows users to create, manipulate, and use contexts in KEx. The module has two main components:

– **Context editor**: provides users with a simple interface to create and edit contexts, and to classify information with respect to a context. This happens by allowing users to create links from a resource (identified by a URI) to a node in a context. Examples of resources are: documents in local directories, the address of a database access services, addresses of other K-peers that provide information that a KN wants to explicitly classify in its own context.
– **Context browser**: is part of Seeker component (GUI User in Figure 1) and allows users to navigate contexts in the context repository. The main reasons for navigating a context in KEx are two. The first is obviously to find document in the local knowledge repository by navigating the semantic structure. The second, and more important reason, is to build queries. The intuitive idea is that users can make context dependent queries (namely, from their perspective) by selecting a category in one of the available contexts. Once a category is selected, the context browser builds a *focus*[4] – namely a contextual interpretation of the user's query – by automatically extracting the relevant portion of the context to which the category belongs. The focus is then used as a basis for meaning coordination and negotiation with other K-peers during the search.

2.2 Roles of K-peers in KEx

Each K-peer can play two main roles: seeker and provider. Their interactions are represented in Figure 2, and described in detail in the following two sections.

Seeker As a seeker, a K-peer allows users to search for documents (and other information) from other K-peers and federations (see Section 2.3). The seeker supports the

[4] See [17] for a formal definition of focus.

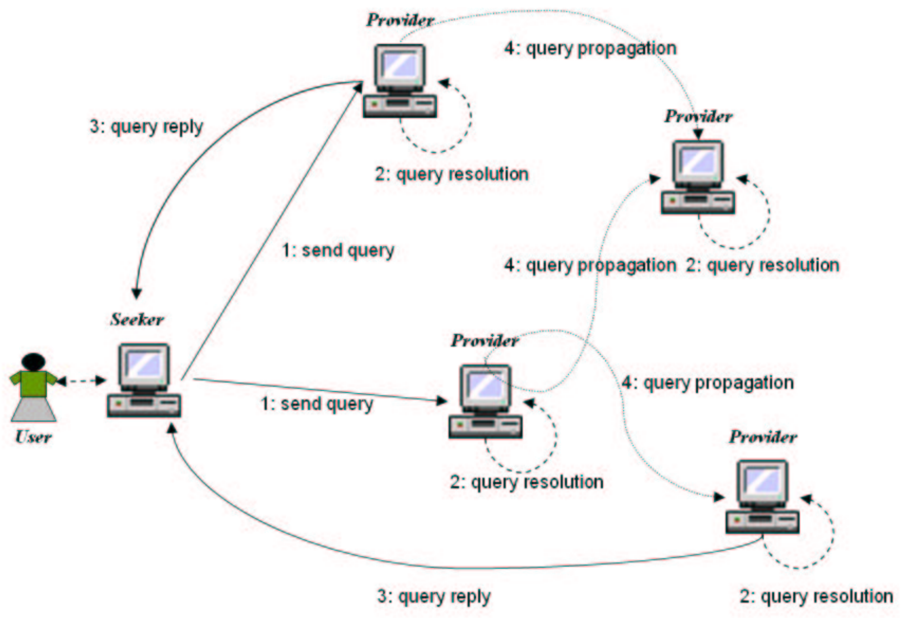

Fig. 2. The KEx system: interaction between Seeker and Provider roles

user in the definition of context-dependent queries through the context browser. A query is composed by a query expression and a focus. A query expression is a list (possibly empty) of one or more keywords provided by a user; a focus is a portion of a context determined by the category that the user has selected. Moreover, the seeker provides the discovery mechanism, used to find resources to which the query is to be sent. The user decides to send the query to some of the available K-peers and federations. When the user submits the query, the seeker activates a session associated to that query (there can be only one active session for each seeker). In a session, a seeker can receive several asynchronous replies from the providers which resolved the query (through the meaning negotiation protocol, see below) and called back the seeker. The results returned to the user are composed by the aggregation of all the results received from the providers; each result is made up of a list of document descriptors (i.e., name of the document, short description, and so on). Each result is presented together with the part of context that the provider has matched against the current query. This relationship between contexts can be used as an opportunity for learning relationships across contexts of different KNs that the seeker can store and reuse for future queries (see section 2.3). Finally, if one or more interesting documents are found, the seeker can contact the K-peers that have the documents and, if possible, download them.

Provider. The provider is the second main role in the KEx system. It contains the functionalities required to take and resolve a query, and to identify the results that must to be returned to the seeker. When a K-peer receives a context-dependent query (keywords + focus), it instantiates a provider (which is configured to use a set of contexts and to provide documents in a given portion of the knowledge repository), and tries to resolve the query in two ways:

– **Semantic resolution**: using a context matching algorithm [17], the provider searches for relations between the locally available contexts and the query's focus. More specifically, the matching algorithm searches categories whose associated contextual information in the providers contexts matches (in a sense defined in [17]) with the querys focus. If a match is found, the URIs of the resources associated to the provider's context are returned to the seeker, together with a short information on the reason why a semantic match was found. If the matched category contains also links to resources in other K-peers, the provider propagates the query to those K-peers.
– **Lexical resolution**: using a keyword-based indexer, the provider searches for the occurrence of specific keywords into the set of documents of the local repository.

If the query contains only keywords, the provider will use only the lexical search; if it contains only a focus, the provider will use only the semantic search; if both are available, the outcome will be the result of intersecting the semantic and lexical results.

2.3 K-Services

KEx provides a collection of services which have an important role in supporting knowledge exchange (that's why they are called K-services). The main K-services are described in the following sections.

Context normalization and enrichment. This service allows to perform a linguistic normalization (e.g., deleting stop words, tokenizing, part-of-speech tagging, etc.) on user defined contexts, and to use knowledge from an external linguistic resource (e.g., WordNet) to add semantic information to the categories in a context.

Normalization uses pretty standard NLP techniques, so we do not discuss it here. As to enrichment, it is applied offline to a context defined by a user (see [17] for details). It takes a user-defined context (e.g., a context built with the context editor) as input and returns a semantically enriched context as output. In our current implementation, the result is that linguistic knowledge (e.g., senses, synonyms, hierarchical relations with other categories, and so on) is extracted from WordNet and is "attached" to each context node label.

It is important to say why enrichment is not equivalent to introduce a shared ("objective") semantics in KEx. Indeed, the intuition is that the meaning of a label in each context node has two components:

– the first is the linguistic component, which means that the words used as labels have a meaning (or, better, a set of meanings) in a "dictionary". This is used to record that many words have different meaning (e.g., "apple" as a fruit, "apple" as a tree, and "apple" as a computer brand), even if only one of them is likely to be the relevant one in a given context;

– the second is a sort of pragmatic component, which is given by its position in a context (e.g., its position in the path from the context root). This helps in understanding what the user means on a particular occasion with a word (e.g., "apple" in a path like "computer/software/apple" is different from "apple" in a path like "computer/hardware/printers/apple", even though "apple" has the same dictionary meaning').

The first component of meaning is dealt with in the normalization and enrichment phase (for example, given a context, some potential senses are deleted because of the position of a label in that context); the second is dealt with in the meaning negotiation protocol, and cannot be computed beforehand, as it expresses a user's perspective in making a query (so this is the more "perspectival" aspect of meaning).

It is extremely important to stress that different linguistic resources can be used to enrich a context. So far, we've been using only WordNet, but there's no reason why other resources (like CYC or any other domain-specific ontology) can't be used to replace WordNet. Of course, this introduces a further problem, as the enriched contexts cannot be compared as directly as in the case of a shared resource. In this case, what happens is that each provider, after receiving a query from a context which is normalized with a different linguistic resource, applies a runtime normalization-and-enrichment of the query's focus, this way interpreting the query from its perspective. Then the query can be matched against local contexts. Of course, this introduces a further degree of semantic heterogeneity, and the returned results could be unsatisfactory for the seeker even if the semantic match good from the provider's perspective.

K-federation. A *K-federation* is a group of K-Peers that agree to appear as a unique entity to K-peers that perform a search. Each K-federation can be though as a "social" aggregation of K-peers that display some synergy in terms of content (e.g., as they provide topic-related content, or decided to use the same linguistic resource to create a common "vocabulary", thus providing more homogeneous and specific answers), quality (certify content) or access policies (certify members).

Seekers can send queries directly to K-federations, and the query is managed internally at the federation. In our current implementation, the query is simply distributed to all the members of the federation (and therefore the result is the same as if the query was sent directly to each member of the federation, the only difference being that K-peers explicitly answer as members of the federation), but the idea is that in general each K-federation can implement different methods to process queries.

To become a member of a K-federation, a K-Peer must provide a K-federation Service (the Federation Provider in Figure 1). It implements the required federation protocol (reply to queries sent to the K-federation) and observes the federation membership policy.

Discovery. Discovery is a mechanism that allows the user to discover resources in the P2P network. The user needs to discover K-peers and K-federations available in the network to contact and query them. A peer advertises the existence of resources by publishing an XML document (called *Advertisement*). In the KEx system, two types of resources are advertised:

- **K-peers** that have a provider service to solve queries. The main elements of the advertisement are a description of the peers contexts, and the peer address to contact it, to send it queries, and to retrieve documents;
- **K-federations**, namely sets of peers that have a federation service to solve queries. The federation assures that a query sent to a federation is propagated to all active peers that are member of the federation. In this case the main elements of the advertisement are the federation topic, its address and information for joining the federation.

To discover resources, a peer sends a discovery request to another known peer, or sends a multi-cast request over the network, and receives responses (a list of advertisements) that describe the available services and resources. It is possible to specify search criteria (currently only keywords or textual expression) that are matched against the contents provided by the advertisement related to each peer or federation description.

Query Propagation. This functionality allows the KEx system to distributed queries in a highly dynamic environment. When a provider receives a query, it can forward it to other providers. To decide to which peers the query is to be forwarded, a peer has two possibilities:

- *physical proximity*: the query is sent to peers known through the discovery functionality. This way, peers or providers that are non directly reachable from a seeker, or have just joined the system, can advertise their presence and contribute to the resolution of queries;
- *semantic proximity*: this functionality exploits the fact that contexts can be used not only to classify resources like documents or database records, but also other peers. Thus, if the provider computes some matching between a query and a concept in its own context, and other peers are classified under that concept, the provider forwards the query to these other peer, this way increasing the chances that other relevant peers are reached.

The propagation algorithm is based upon a cost function which allows choosing peers that are regarded as providing more relevant information (assigning a higher value to peers discovered through the semantic method than to peers reached through physical proximity one).

Obviously, several parameters and mechanisms controlling the scope of the search can be implemented to prevent a “message flood”. For example, the time-to-live (TTL), the number of hops, the list of peers already reached, and so on.

Learning. When the matching algorithm finds a semantic correspondence between concepts of different contexts, the provider can store this information for future reuse. This information is represented as a semantic “mapping” between concepts (see [8]), and can be used in three ways:

- when the K-Peer receives a query from a seeker, it can reuse stored mappings to facilitate (and possibly to avoid executing) the matching algorithm;

- a provider can use the existing mapping to forward a query to other peers that have a semantic relation with the query's focus;
- the seeker can search into the available mappings to suggest the user a set of providers with which it already had previous interactions and are considered qualified with respect to the semantic meaning of the concept selected in a query.

Using this mechanism, the K-peer network defines and increases the number and quality of the semantic relations among its members, and becomes a dynamic web of knowledge links.

3 Development Framework

In this section we briefly show how the non-functional requirements of a DKM system drive the choice of a particular architectural pattern design (a *peer-to-peer* system) and an underlying technology framework (the *Jxta* Project). In particular, KEx is under development within the business setting of an Italian national bank, and of an international insurance company[5].

From an implementation point of view, we started from *JXTA*[6], a set of open, generalized peer-to-peer protocols that allow devices to communicate and collaborate through a connecting network. This P2P framework provides also a set of protocols and functionality such as: a decentralized discovery system, an asynchronous point-to-point messaging system, and a group membership protocol. A *peer* is a software component that runs some or all the JXTA protocols; every peer needs to agree upon a common set of rules to publish, share and access resources (like services, data or applications), and communicate among each others. Thus, a JXTA peer is used to support higher level processes (based, for example, on organizational considerations) that are built on top of the basic peer-to-peer network infrastructure; they may include the enhancement of basic JXTA protocols (e.g. discovery) as well as user-written applications. JXTA tackles these requirements with a number of mechanisms and protocols: for instance the publishing and discovery mechanisms, together with a message-based communication infrastructure (called "pipe") and peer monitoring services, supports decentralization and dynamism. Security is supported by a membership service (which authenticates any peer applying to a peer group) and an access protocol (for authorization control). The flexibility of this framework allows to design distributed systems that cover all the requirements of a DKM application, using the JXTA P2P capabilities, completed and enhanced through the implementation of user-defined services. As shows in the previous sections, in the KEx system we combine the P2P paradigm (characterizing a network of knowledge nodes as a network of distributed peers) and JXTA as an implementation infrastructure.

[5] This architecture is under development as part of EDAMOK (*Enabling Distributed and Autonomous Management of Knowledge*), a joint project of the Institute for Scientific and Technological Research (IRST, Trento) and of the University of Trento.

[6] JXTA is a P2P open source project started in 2001 and supported by Sun Microsystems. See `http://www.jxta.org/` for details.

4 Conclusions and Research Issues

In this paper, we argued that technological architectures, when dealing with processes in which human communication is strongly involved, must be consistent with the social architecture of the process itself. In particular, in the domain of KM, technology must embody a principle of distribution that is intrinsic to the nature of organizational cognition. Here, we suggest that P2P infrastructures are especially suitable for KM applications, as they naturally implement meaning distribution and autonomy. It is perhaps worth noting at this point that other research areas are moving toward P2P architectures. In particular, we can mention the work on P2P approaches to the semantic web [1], to databases [14], to web services [19]. We believe this is a general trend, and that in the near future P2P infrastructure will become more and more interesting for all areas where we can't assume a centralized control.

A number of research issues need to be addressed to map aspects of distributed cognition into technological requirements. Here we propose two of them:

– **social discovery and propagation**: in order to find knowledge, people need to discover who is reachable and available to answer a request. On the one hand, broadcasting messages generates communication overflow, on the other hand talking just to physically available neighbors reduces the potential of a distributed network. A third option could be for a seeker to ask his neighbors who they trust on a topic and, among them, who is currently available. Here the question is about social mechanisms through which people find – based on trust and recommendation – other people to involve in a conversation. A similar approach could be used in order to support the propagation of information requests;
– **building communities**: if we consider communities as networks of people that, to some extent, tend to share a common perspective [3], mechanisms are needed to support the bottom-up emergence of semantic similarities across interacting KNs. Through this process, which are based on meaning negotiation protocols, people can discover and form virtual communities, and within organizations, managers might monitor the evolving trajectories of informal cognitive networks. Then, such networks, can be viewed as potential neighborhoods to support social discovery and propagation.

References

1. M. Arumugam, A. Sheth, and I. Budak Arpinar. The peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In *WWW2002 Workshop on Real World RDF and Semantic Web Applications. Honolulu, Hawaii (USA)*, 2002.
2. M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual Reasoning Distilled. *Journal of Theoretical and Experimental Artificial Intelligence*, 12(3):279–305, July 2000.
3. J.R. Boland and R.V.Tenkasi. Perspective making and perspective taking in communities of knowing. *Organization Science*, 6(4):350–372, 1995.
4. M. Bonifacio, P. Bouquet, and R. Cuel. Knowledge Nodes: the Building Blocks of a Distributed Approach to Knowledge Management. *Journal of Universal Computer Science*, 8(6):652–661, 2002. Springer Pub. & Co.

5. M. Bonifacio, P. Bouquet, and A. Manzardo. A distributed intelligence paradigm for knowledge management. In *Working Notes of the AAAI Spring Symposium Series 2000 on Bringing Knowledge to Business Processes*. AAAI, March 18-20 2000.
6. M. Bonifacio, P. Bouquet, and P. Traverso. Enabling distributed knowledge management. managerial and technological implications. *Novatica and Informatik/Informatique*, III(1), 2002.
7. P. Bouquet. *Contesti e ragionamento contestuale. Il ruolo del contesto in una teoria della rappresentazione della conoscenza*. Pantograph, Genova (Italy), 1998.
8. P. Bouquet, A. Donà, and L. Serafini. ConTeXtualizedlocal ontology specification via ctxml. In *MeaN-02 – AAAI workshop on Meaning Negotiation*, Edmonton, Alberta, Canada, 2002.
9. G. C. Bowker and S. L. Star. *Sorting things out: classification and its consequences*. MIT Press., 1999.
10. J. Dinsmore. *Partitioned Representations*. Kluwer Academic Publishers, 1991.
11. D. Dougherty. Interpretative barriers to successful product innovation in large firms. *Organization Science*, 3(2), 1992.
12. G. Fauconnier. *Mental Spaces: aspects of meaning construction in natural language*. MIT Press, 1985.
13. C. Ghidini and F. Giunchiglia. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.
14. F. Giunchiglia and I. Zaihrayeu. Making peer databases interact – a vision for an architecture supporting data coordination. In *6th International Workshop on Cooperative Information Agents (CIA-2002), Universidad Rey Juan Carlos, Madrid, Spain, September 18 - 20, 2002*, 2002. Invited talk.
15. I. Goffaman. *Frame Analysis*. Harper & Row, New York, 1974.
16. T. Kuhn. *The structure of Scientific Revolutions*. University of Chicago Press, 1979.
17. B. Magnini, L. Serafini, and M. Speranza. Linguistic based matching of local ontologies. In P. Bouquet, editor, *Working Notes of the AAAI-02 workshop on Meaning Negotiation. Edmonton (Canada)*. AAAI, AAAI Press, 2002.
18. J. McCarthy. Notes on Formalizing Context. In *Proc. of the 13th International Joint Conference on Artificial Intelligence*, pages 555–560, Chambery, France, 1993.
19. M.P. Papazoglou, J. Jang, and B.J.Kraemer. Leveraging web-services and peer-to-peer networks. October 2002.