Peer-to-Peer and Multi-Agent Systems technologies for Knowledge Management Applications. An Agent-Oriented analysis.

D. Bertolini†, P. Busetta†, M. Nori†, A. Perini†
†ITC-IRST, Via Sommarive 18, 38055 Povo, Trento, Italy

Abstract— **This paper focuses on a framework for developing knowledge management (KM) applications that integrates peer-to-peer (P2P) and multi-agent systems (MAS) technologies. The objective of this framework is to support a particular KM paradigm that emphasizes aspects such as autonomy and distributedness of knowledge sources. In particular, we present a characterization of peer-to-peer in terms of a general architectural pattern and a set of guidelines for designing peer-to-peer applications according to the proposed framework. We adopt an agent-oriented approach that extends Tropos, a software engineering methodology introduced in earlier papers.**

I. Introduction

In the last few years, we assisted to an explosive growth of application areas such as electronic commerce and services at support of business processes (including knowledge management systems), which are raising challenging issues for the research in distributed systems technology and in analysis and design methodologies. Several technological solutions are currently proposed, ranging from technologies for implementing multi-tiers client-server architectures, to multi-agent systems (MAS), and peer-to-peer architectures (P2P).

When designing these complex systems, we face the problem of evaluating different technologies at the light of the application's requirements, and we analyze previous experiences in which they where applied in order to find useful design patterns that solved recurrent problems or even reusable software code.

Focusing on knowledge management (KM) applications, some critical aspects such as autonomy and heterogeneity of knowledge sources, as well as the intrinsic distributedness of many processes (such as discovery and deployment of content) have to be taken into account. In this context, technologies such as MAS [8] and P2P [16] have been proposed as promising solutions.

In this paper, we propose to use an agent-oriented approach for analyzing P2P from an organizational point of view, as well as an application and technological solution. Our aim is twofold: to address a particular KM problem (distributed information retrieval), and to propose a possible integration between P2P and MAS. First, we characterize P2P in terms of an architectural pattern, obtained by applying the *Tropos* methodology, presented in earlier papers [9]. Then, we present an architectural framework that integrates P2P and multi-agent systems, basically proposing P2P as a way of deploying an open MAS and providing some supporting services, which are regarded as basic elements of distributed applications (such as a discovery mechanism). We sketch also a set of guidelines for a system designer on when and how to apply a P2P approach; these guidelines are targeted at a specific class of applications (knowledge management systems) but can be easily generalized to others.

This paper is structured as follows. Section II briefly presents the distributed knowledge management approach that motivated this research, and other work relevant to this paper. Section III, introduces some basic Tropos concepts and notations, which are used, in Section IV, to discuss some typical peer-to-peer architectures and to identify their common patterns. In Section V, we focus on JXTA, the P2P framework we are currently using. Section VI presents the guidelines for designing the architecture of a DKM system applying both P2P and MAS concepts. Section VII introduces our current experimental system. Finally, Section VIII presents conclusions and future works.

II. Background

Common KM systems support the collection and categorization of knowledge with respect to a single, shared perspective, and its redistribution to its users by a variety of means. In many instances, this leads to the construction of one or a few repositories of documents, organized around a single ontology or other meta-structures. Users are given tools to search, browse, or receive documents as soon as they become available, varying from simple Web interfaces to sophisticated agents running on the users' desktops.

However, common wisdom is that building the shared perspective at the core of a KM systems is an expensive task, sometimes impossible to achieve when users have substantially different goals and background. To tackle this issue, our research group is investigating a novel approach called *distributed knowledge management* (DKM) [5]. The idea at the basis of DKM is supporting the integration of autonomously managed KM systems, without forcing the creation of centralized repositories, indexes, nor shared ontologies. This integration involves the deployment of a set of complex protocols and algorithms for information retrieval, natural language processing, machine learning, deductive and inductive reasoning, and so on, sometimes requiring the direct participation of human users. Initial publications include [12], [6][1].

[1]The work presented in the paper is realized as part of the *EDAMOK* project,

To analyse both organizational and technical aspects of a DKM system, we adopt an extensions to Tropos methodology, that for the support of DKM are part of the development of a larger framework, whose aim is to provide both a methodology and technological support to analysts, designers, and developers. As part of this framework, we are investigating both multi-agent and peer-to-peer technologies.

Different lines of research are relevant to the work presented here, ranging from peer-to-peer and agents applications. For instance, two recent works have attempted to blend peer-to-peer and agents into a single framework: Anthill [1] and InfoQuilt [15]. Both see peer-to-peer as a paradigm for networks of autonomous systems that may join or leave at any time.

Of particular interests is the proposal of using MAS as a core technology for KM (see [8]). For instance, Frodo (FRamework fOr Distributed Organizational memories, a 3-year project developed at DFKI in Germany from January 2000[2]) aims at the design of a scalable, agent-based middle-ware for distributed Organizational Memories managing enterprise knowledge. This agent framework, implemented by a FIPA-complaint platform, provides knowledge representation, communication and inference services especially tailored for the realization of distributed OM applications.

In our approach, a P2P characterization is used to identify organizational and non-functional requirements of a class of distributed systems. The proposed guidelines at support of application design consider aspects of decentralization and autonomy of system components that can be realized adopting some basic services commonly available in P2P systems (i.e. discovery, advertisement of peer and services, group management and so on), which can be used by a multi-agent platform to provide specific knowledge functionalities.

III. An Overview of Tropos

The *Tropos* methodology [9] adopts an agent oriented approach to software development starting from the very early stage of requirement specifications, when the environment and the system-to-be are analyzed, down to system design and implementation. The methodology identifies a number of phases (*early requirements*, *late requirements*, *architectural design*, *detailed design*, and *implementation*) and has been applied to several case studies [10]. The core process of the methodology consists in performing conceptual modeling using a visual language that provides intentional and social concepts such as actor, goal, belief, plan and dependency. An *actor* models an entity that has strategic goals and intentionality, such as a physical *agent*, a *role* or a *position*. A *role* is an abstract characterization of the behavior of an actor within some specialized context, while a *position* represents a set of roles, typically covered by one agent. The notion of actor in Tropos is a generalization of the classical AI notion of software agent. *Goals* represent the strategic interests of actors. A *dependency* between two actors indicates that an actor depends on another in order to achieve a goal, execute a plan, or exploit a resource. Tropos distinguishes between hard and soft goals, the latter having no clear-cut definition and/or criteria as to whether they are satisfied. Softgoals are useful for modeling software qualities [7], such as security, performance and maintainability. A Tropos model is represented as a set of diagrams. In particular, actor and dependency models are graphically represented as *actor diagrams* in which actors are depicted as circles, their goals as ovals. The network of dependency relationships among actors are depicted as two arrowed lines connected by a graphical symbol varying according to the dependum: a goal, a plan or a resource. Actor goals and plans can be analyzed from the point of view of the individual actor using three basic reasoning techniques: *means-end analysis*, *contribution analysis*, and *AND/OR decomposition*. During this analysis, new actor dependencies can be identified.

Tropos is currently being extended with concepts suitable to model some specific notions – such as distributed knowledge, autonomy and coordination – that are peculiar to DKM, in order to support an early requirement model of a DKM domain [13].

In this paper, we focus on a later phase in software development: given a requirement analysis of a DKM problem, we provide guidelines for defining one or more suitable system architectures. According to *Tropos*, these types of issues are typically faced during the *architectural design* phase.

IV. The Peer-to-Peer Virtual Community Pattern

In this section, we use Tropos to analyze which functional and non-functional requirements are successfully satisfied by peer-to-peer systems [14]. In particular, we consider two well known systems, Gnutella and Napster, both supporting file sharing (in particular MP3 sound files).

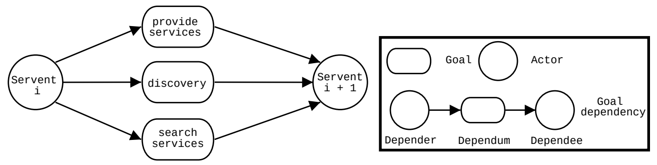

Fig. 1. Gnutella: architectural design, according to the Tropos methodology

The logical architecture of *Gnutella*[3] is described in the *Tropos* actor diagram depicted in Figure 1. Both actors model a basic role, the Servent (Gnutella's terminology for peer), that has the goals of discovering other servents, looking for MP3 files, and providing its own MP3s, and depends on all other servents for achieving them. In other words, the goals of a servent are achieved only by means of a virtual community of peers, each one playing the role of servent. In the diagram, this is represented as goal dependencies between two generic servents. The dependencies are symmetric for every servent; for simplicity,

funded by the Provincia Autonoma di Trento (P.A.T.)
`http://sra.itc.it/projects/edamok/index.html`

[2]FRODO Project HomePage `http://www.dfki.uni-kl.de/frodo/`

[3]`http://www.gnutella.com`

we have shown them going only in one direction. The discovery goal enables the community to be dynamic, since peers can join and leave the community at any time without having to register their presence anywhere. There is no centralized service local to any one peer; conversely, all the peers provide (and take advantage of) the same sets of services.

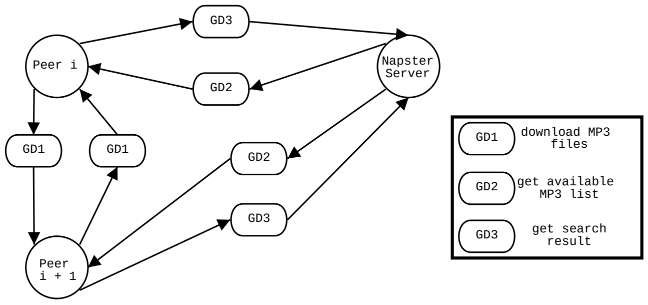

Fig. 2. Napster: architectural design

In *Napster*[4], each client, represented by Peer actors in Figure 2, has similar goals to those of Servents in Gnutella (search and download files). However, in order to obtain a search result, the client must contact a central server (Napster Server). This server maintains a list of all the active clients and a list of all the shared MP3 files. The server depends on the clients for building these lists. This situation is represented by a set of goal dependencies in Figure 2: the generic Peer depends on Napster Server for getting a search result (goal dependency GD3) and Napster Server depends on Peer to get the lists of available MP3 files (goal dependency GD2). Analogously to Gnutella, two generic peers depend on each other for downloading MP3 files (goal dependency GD1). So, the individual goal is obtained by means of a community of peers (Napster clients) that coordinate with an actor playing a distinct role in the community, i.e. the server.

We abstract these two architectures, and others not discussed here, in the basic model depicted in Figure 3. The Individual actor models somebody who has at least one of two goals: accessing a resource (or, equivalently, using a service); and, letting others access her own resources (or use her own services). An individual has a set of constraints and preferences (shown as softgoals in the diagram) which drive her behavior when achieving these goals. In particular, an important requirement is being autonomous, i.e. being able to control what to access or to make available to the external world, and when.

The actor Virtual P2P community has the main goal of letting its members cooperate for sharing a specific type of resource or service. Three additional requirements (shown as softgoals) have been identified: being highly dynamic (available resources and services provided by members can be added or removed at any time); being decentralized, i.e. the community is able to achieve its main goal independently of any specific member or component; and finally, being composed of peers on an equal

[4]http://opennap.sourceforge.net

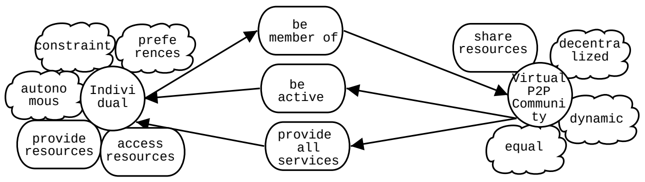

Fig. 3. The peer-to-peer virtual community pattern

basis, that is, every member is compelled to provide (at least potentially) resources or services and has the right to access the services of others.

The dependency be member of captures the fact that the individual's goals can be satisfied by joining the community. The act of joining implies the acceptance of the community's main goal and rules, highlighted above. Conversely, a community can achieve its goal only if it is active, and this can be accomplished only by having individuals as members. Finally, the dependency provide all services models the rule that in a peer-to-peer community all members are equal, i.e. provide the same set of services.

Going back to the systems discussed previously, Gnutella is a "perfect" P2P system, since it satisfies all non-functional requirements of the community highlighted in Figure 3. Similarly, it may be argued that Napster is not a real P2P system, since the community depends on a centralized service.

The P2P virtual community represents an organizational analysis pattern, similar to the patterns described in [11], and could be applied to identify specific requirements that characterize a part of a distributed system.

V. JXTA P2P: Logical Architecture Elements

In order to design a distributed application, we need a technological infrastructure that support the goals and the social dependencies discovered in the requirements analysis phase.

We focus now on *JXTA*[5], a set of open, generalized peer-to-peer protocols that allow devices to communicate and collaborate through a connecting network.

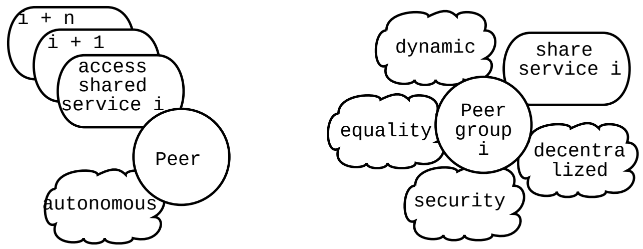

Fig. 4. JXTA: architectural design overview

From our perspective, the fundamental concepts of JXTA are three: *peer*, *peer group* and *service*.

[5]A P2P open source effort started in April 2001. http://www.jxta.org/ and
http://spec.jxta.org/v1.0/docbook/JXTAProtocol.html

A *peer* is "any device that runs some or all the Project JXTA protocols". The complex layering of the JXTA protocols and the ability for a peer to simultaneously participate to more than one peer group (described below) imply that a peer is – at least conceptually – a multi-threaded program.

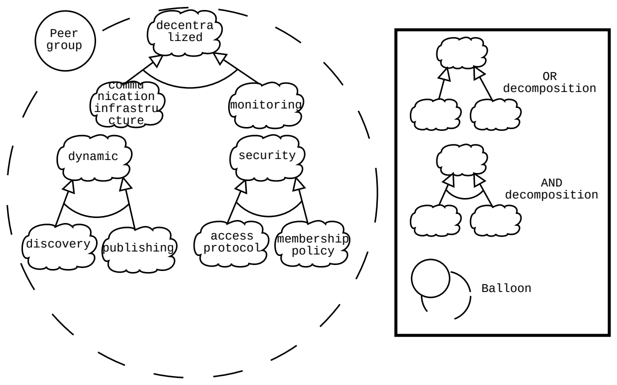

Fig. 5. JXTA: PeerGroup architectural design

A *peer group* is a collection of peers that have agreed upon a common set of rules to publish, share and access "codats" (shorthand for code, data, applications), and communicate among themselves. Each peer group can establish its own membership policy, from open (anybody can join) to highly secure and protected (join only if you have sufficient credential). Thus, a peer group is used to support structuring (based on social organizations or other criteria) on top of the basic peer-to-peer network. As mentioned above, a peer may be part of many groups simultaneously.

A peer provides to each of its group a set of *services*, which are advertised to the other members of the group. In other words, groups in JXTA support a model for service publishing which is alternative to the traditional, centralized directory services model. Each group specifies a set of services that have to be provided by its members; they may include supporting basic JXTA protocols (e.g. discovery) as well as user-written applications, such as file sharing.

From a high-level perspective, these concepts can be modeled in terms of roles in the architectural design diagram depicted in Figure 4, which extends the pattern depicted in Figure 3. A Peer has *n* goals access shared service *i*, one for each type of service it needs, and the requirement (specified as a softgoal) of being autonomous. A generic actor PeerGroup has the goal of sharing a service of a specific type *i* and a set of requirements: to support the decentralized distribution of services, to allow for dynamic membership of the group, to support security management, and to require that all members provide equal services. JXTA tackles these requirements using a number of mechanisms, represented as goal decompositions in Figure 5. The publishing and discovery mechanisms, together with a message-based communication infrastructure and peer monitoring services, support decentralization and dynamism. security is supported by a membership policy (which authenticates any peer applying to a peer group) and an access protocol (for authorization control).

In summary, it can be said that the creation of a PeerGroup follows from the need to define a set of peers that are able to communicate and interact with a common set of capabilities and protocols, and that are aggregated for security reasons (i.e., no extraneous peer can intervene amongst them).

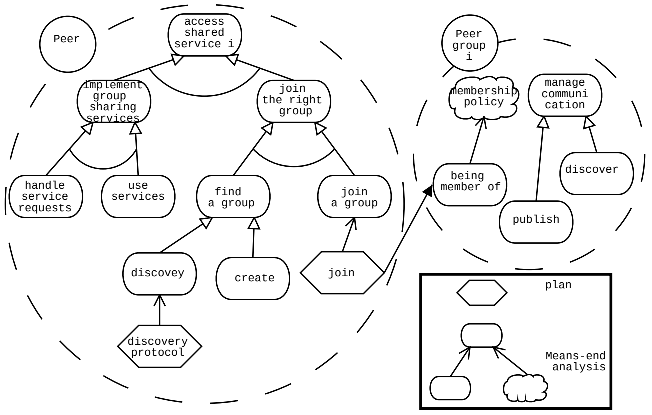

Fig. 6. JXTA: Peer architectural design

Figure 6 shows that a Peer's main goal (access shared service *i*) can be decomposed into two subgoals: join the right group and implement group sharing services. The first means that the peer must find a group that it is authorized to join and that provides the services it needs. This goal can be further decomposed into join a group and find a group. The latter can be satisfied by using the JXTA discovery service; if no adequate group is found, a new one can be created. Joining a group that has been discovered or created depends on its membership policy. Once a group has been joined, a Peer must implement all the services required by that group, implementing both the client side (use services) and the server side (handle service requests). It is important to stress that a Peer is autonomous in deciding how to provide a service (only protocols are predefined), and that a Peer can join different PeerGroups in order to achieve different goals.

VI. Architectural Design of Communities in a DKM System

In this section, we sketch the guidelines for designing the architecture of a DKM system, focusing on the support for virtual communities. Input to the process described here is, in Tropos terms, the output of a late requirements phase (Section III); that is, a domain analysis which included the system-to-be. The late requirements phase led to the identification of the stakeholders, including knowledge and service sources and of their users; how this analysis is performed is outside the scope of this paper. The guidelines consist of the following steps:

1) apply the P2P virtual community pattern;
2) design the JXTA peer group implementing the virtual community;

3) design the agents that implement a peer service.

Each step is briefly described below, while in the next section we present their practical application.

a) **Applying the P2P Virtual Community Pattern:** At this step, the designer has to answer to a basic question: can a Tropos late requirement model of a DKM problem be refined as a virtual P2P community model?

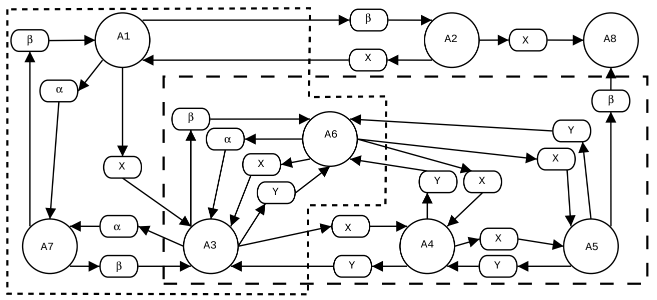

Fig. 7. Identifying P2P virtual communities. The *first community* (dashed line) is composed by actors involved in pair dependencies for *goals X and Y*, the *second* (dotted lines) by actors involved in pair dependencies for *goals* α *and* β

By definition, actors in a DKM scenario, as well as individuals in the community pattern, want to keep their autonomy. A designer, then, should look for the following conditions to be realized (they are illustrated in Figure 7 where two P2P virtual communities can be identified):

- there is a subset C of the actors that have pair-wise dependencies (e.g., actor A1 depends on A7 for goal α and A7 depends on A1 for β) in such a way that any actor is dependent on many, if not most, other actors in C. In other words, each actor is at the center of a sort of "star" network of reciprocal dependencies with many others;
- these dependencies can be generalized to one or a few types only. In particular, they are of type "access" and "provide" a service or a resource.

If these conditions are satisfied, then the designer can apply the P2P virtual community pattern to generate a model for a community supporting C. The main goals of this virtual community are suitable abstractions of the generalized dependencies linking the actors in C. The goals of an individual are also generated from each generalized dependency, and are two: achieve the objective of the dependency, and, conversely, satisfy it. A validation step is necessary, and consists of two main activities:

- verifying that the goals of the actors in C can be effectively satisfied by adopting the goals of the individual in the community model; that is, verify that the P2P virtual community helps in achieving the actors' goals. This may be performed as a means-end analysis, which decomposes those hard goals into plans that, eventually, are satisfied by adopting the individual's goals;
- verifying that the general soft goals of a P2P virtual community (dynamism, equality, and decentralization) are at least partially achieved by the community being designed.

b) **Designing a JXTA Peer Group:** Once that one or more P2P virtual communities have been identified, their supporting infrastructure can be designed. Assuming that JXTA is adopted as the basic communication technology, our major – and natural – choice is then to associate a JXTA peer to each individual, and a JXTA peer group to each P2P virtual community. Thus, a social actor (no matter if it is a single person or a group) will have, as its supporting system, a peer participating to as many peer groups as communities that have been identified during the requirement analysis of its DKM problems.

c) **Agents as JXTA Services:** The final step is to design how individuals implement their goals with respect to each of the communities they form part of. From a JXTA perspective, it is necessary to specify which application services have to be provided by a peer, and how they are published on the network. JXTA, however, leaves the designer total freedom concerning their communication protocols and internal implementation. JXTA provides its own communication mechanisms (unreliable message queues called "pipes" and an XML based encoding format), but the application may decide to use something different, since JXTA supports publishing and discovery of communication end-points of any type (i.e., the address to which to send messages in order to obtain a service from a certain peer) as long as they can be represented in XML. Moreover, JXTA allows the publishing of additional service-specific information – also encoded as XML documents – along with their communication end-points; this gives a nice opportunity for targeted discoveries and filtering of potential peers.

The services to be provided by the peer associated to an individual are easily identified from the individual's goals with respect to the community. Once the peer protocols and publishing information have been defined, the design of the actors participating to the community can be performed in parallel. The Tropos methodology naturally leads to the design of multi-agent systems; thus, it is most likely that the implementation of services are agents themselves, interacting both with agents internal to their own actor and with other agents implementing other actors of the same community.

VII. A Technological Framework

In the previous sections, we focused on the analysis and characterization of a class of distributed applications, i.e. peer-to-peer systems, and proposed an initial set of guidelines for the design of distributed KM applications. In parallel to this methodological work, a P2P architecture coherent with the vision of DKM has been developed (see [4]). This knowledge exchange system (called KEx) embodies the two principles at the base of DKM, i.e. *semantic autonomy* and *semantic coordination*, in a quite straightforward way:

- each community of knowing is represented by a peer that provides all the services needed to create and organize its own local knowledge (autonomy);

- social structures and protocols of meaning negotiation have been defined in order to achieve semantic coordination.

More specifically, KEx is a P2P system that allows a collection of so-called Knowledge Nodes (individuals or groups) to categorize, provide and search documents on a semantic basis without imposing a global classification (ontology or schema). Every node manages its knowledge (documents and data) using a local schema (e.g. a file-system structure or a database schema); then, a context is defined as a partial representation of the world from an individual's perspective, organizing and classifying its knowledge. That context is used by a peer when it must look for specific information, in order to "negotiate" a semantic meaning with other peers and provide a more accurate query resolution service.

The first application built with KEx is under testing within an Italian national bank (for more detail see [3]). Future KEx applications will be developed adopting the methodology proposed in the sections above, in order to validate it and to define the many elements currently missing.

VIII. Conclusions and Future Works

This paper sketches initial results of a study aimed at defining a framework for building DKM applications, adopting P2P concepts as organizational and technological components and an agent-oriented metodology. The process is based on *Tropos*, an agent oriented software engineering methodology. We focused on the architectural design phase, when the application requirements are analyzed in the light of known architectural patterns, and defined some guidelines driving the development of DKM systems as a combination of a specific peer-to-peer infrastructure, JXTA, and multi-agent technologies. More details on the proposed framework, including a case-study to illustrate, can be found in [2].

Our long term objective is to complete the framework, working both on the conceptual aspects required to model a DKM application and on the technology, currently being tested in a real-world application.

References

[1] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A framework for the development of agent-based peer-to-peer systems. Technical Report UBLCS-2001-09, University of Bologna, Italy, 2001. http://www.cs.unibo.it/projects/anthill/.

[2] D. Bertolini, P. Busetta, A. Molani, M. Nori, and A. Perini. Designing peer-to-peer applications: an agent-oriented approach. In *Proc. of Workshop "Agent Technology and Software Engineering" at NODe'2002 Conference*, Montreal CA, 2002.

[3] M. Bonifacio, P. Bouquet, and R. Cuel. Knowledge nodes: the building block of a distributed approach to km. *Journal of Universal Computer Science*, 8(6), 2002.

[4] M. Bonifacio, P. Bouquet, G. Mameli, and M. Nori. A peer-to-peer architecture for distributed knowledge management. In *Proc. of Fourth International Conference on Practical Aspects of Knowledge Management (PAKM02)*, Vienna, Austria, 2002.

[5] M. Bonifacio, P. Bouquet, and P. Traverso. Enabling Distributed Knowledge Management: Managerial and technological implication. *Novatica and Informatik/Informatique*, 3(1), 2002.

[6] P. Bouquet, A. Donà, L. Serafini, and S. Zanobini. Contextualized local ontologies specification via ctxml. In *MeaN-02 – AAAI workshop on Meaning Negotiation*, Edmonton, Alberta, Canada, 2002.

[7] L. K. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Publishing, 2000.

[8] Virginia Dignum. An overview of agents in knowledge management.

[9] F. Giunchiglia, A. Perini, and J. Mylopoulus. The Tropos Software Development Methodology: Processes, Models and Diagrams. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS02)*, pages 63–74, Bologna, Italy, July 2002. ACM press.

[10] F. Giunchiglia, A. Perini, and F. Sannicolò. Knowledge level software engineering. In J.-J.C. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, LNCS 2333, pages 6–20. Springer-Verlag, Seattle, WA, USA, Proceedings of the eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL) edition, August 2001. Also IRST Technical Report 0112-22, Istituto Trentino di Cultura, Trento, Italy.

[11] M. Kolp, P. Giorgini, and J. Mylopoulos. A goal-based organizational perspective on multi-agents architectures. In *Proceedings of the eighth International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages, ATAL'01*, Seattle, USA, August 2001.

[12] B. Magnini, L. Serafini, and M. Speranza. Linguistic based matching of local ontologies. In *MeaN-02 – AAAI workshop on Meaning Negotiation*, Edmonton, Alberta, Canada, 2002.

[13] A. Molani, P. Bresciani, A. Perini, and E. Yu. Intentional analysis for knowledge management. Technical report, ITC-IRST, 2002.

[14] Andy Oram, editor. *Peer-to-Peer Harnessing the Power of Disruptive Technologies*. O'Reilly Associates, 2001.

[15] S. Patel and A. Sheth. Planning and optimizing semantic information requests using domain modeling and resource characteristics. In *Proc. of the International Conference on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, September 2001. http://lsdis.cs.uga.edu/proj/iq/iq_pub.html.

[16] Eric Tsui. Technologies for personal and peer-to-peer (p2p) knowledge management.