Agent-oriented Constructivist Knowledge Management
Renata S. S. Guizzardi

Research in Knowledge Management (KM) has evolved substantially in the past 30 years, coming from a centralized view of KM processes to a distributed view, grounded in organizational and cognitive sciences studies that point out the social, distributed, and subjective nature of knowledge. However, KM systems still face considerable resistance, mainly because they generally impose a specific process instead of fitting in the current practices of the organization.

This thesis defends a human-centric view on KM, proposing *Constructivism* as the theoretical framework to guide the development of KM systems and practices. In general, a constructivist perspective on KM focuses on how knowledge emerges, giving great importance to the knowledge holders and their natural practices.

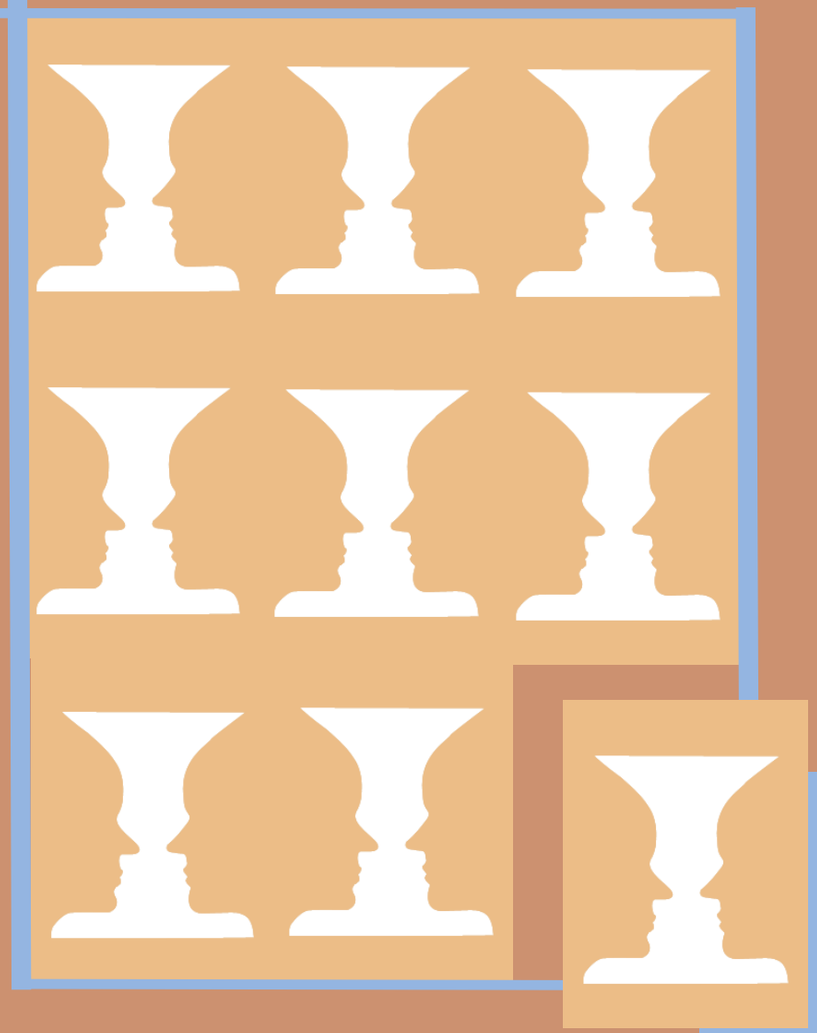
Aiming at observing the compliance of the organizational environment to the principles that characterize Constructivist KM, we propose *ARKnowD*, an agent-oriented methodology to develop KM systems. ARKnowD places strong emphasis in the earlier phases of software development, supporting the analyst on understanding the organizational environment before actually developing a system. Furthermore, ARKnowD consistently conducts to the design of the proposed solution, modeling the system entities, interaction and internal behavior.

In this thesis, we demonstrate the applicability of ARKnowD on the analysis of an organizational scenario. This analysis leads to the proposal and development of a socially-aware recommender system named *KARe*. The core of the system regards a recommendation mechanism, based on an innovative information retrieval technique presented in this thesis. Our work comprises the description, implementation and evaluation of such mechanism.

Renata S.S. Guizzardi

AGENT-ORIENTED CONSTRUCTIVIST KNOWLEDGE MANAGEMENT

AGENT-ORIENTED CONSTRUCTIVIST KNOWLEDGE MANAGEMENT



RENATA S.S. GUIZZARDI

Agent-oriented Constructivist Knowledge Management

Renata S. S. Guizzardi

Enschede, The Netherlands, 2006
CTIT PhD Thesis Series, No. 06-78

Samenstelling promotiecommissie:

Vorzitter, secretaris:	prof.dr.ir. A. J. Mouthaan (Universiteit Twente)
Promotor:	prof.dr. D. Konstantas (University of Geneva)
Assistent Promotor:	dr.ir. M.J. van Sinderen (Universiteit Twente)
Leden:	prof.dr. G. Wagner (Brandenburg University of Technology at Cottbus)
	prof.dr. J. Mylopoulos (University of Toronto)
	prof.dr.ir. A. Nijholt (Universiteit Twente)
	prof.dr. B. Collis (Universiteit Twente)
	dr. L. Aroyo (Eindhoven University of Technology)

CTIT PhD.-thesis series, No. 06-78
ISSN 1381-3617; No. 06-78
ISBN 90-365-2313-3
Centre For Telematics and Information Technology, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

AGENT-ORIENTED CONSTRUCTIVIST KNOWLEDGE MANAGEMENT

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus
prof.dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 9 februari 2006 om 15.00 uur

door
Renata Silva Souza Guizzardi
geboren op 19 november 1974
te Volta Redonda, Brazilië

Dit proefschrift is goedgekeurd door:
prof.dr. D. Konstantas (promotor) en dr.ir. M.J. van Sinderen
(assistent-promotor).

To each member of my family,
for contributing to this endeavor
in countless ways.

To my husband, Giancarlo Guizzardi,
because life without you would be comparable
to Brazil without football, Holland without bicycles,
and Italy without pasta... simply unconceivable!

Preface

In Ancient Times, when written language was introduced, books and manuscripts were often considered sacred. During these times, only a few persons were able to read and interpret them, while most people were limited in accepting these interpretations. Then, along with the industrial revolution of the XVIII and XIX centuries and especially boosted by the development of the press, knowledge slowly became available to all people. Simultaneously, people were starting to apply machines in the development of their work, usually characterized by repetitive processes, and especially focused in the production of consuming goods, such as furniture, clocks, clothes and so on. Following the needs of this new society, it was finally through science that new processes emerged to enable the transmission of knowledge from books and instructors to learners. Still today, people gain knowledge based on these processes, created to fulfill the needs of a society in its early stages of industrialization, thus not being compatible with the needs of the information society.

In the information society, people must deal with an overloading amount of information, by the means of the media, books, besides different telecommunication and information systems technology. Furthermore, people's relation to work has been influenced by profound changes, for instance, knowledge itself is now regarded as a valuable work product and, thus, the workplace has become an environment of *knowledge creation* and *learning*. Modifications in the world economical, political and social scenarios led to the conclusion that knowledge is *the* differential that can lead to innovation and, consequently, save organizations, societies, and even countries from failing

in achieving their main goals.

Focusing on these matters is the Knowledge Management (KM) research area, which deals with the *creation, integration* and *use* of knowledge, aiming at improving the performance of individuals and organizations. Advances in this field are mainly motivated by the assumption that organizations should focus on *knowledge assets* (generally maintained by the members of an organization) to remain competitive in the information society's market. This thesis argues that KM initiatives should be targeted based on a *constructivist* perspective. In general, a constructivist view on KM focuses on how knowledge emerges, giving great importance to the knowledge holders and their natural practices.

With the paragraph above, the reader may already have an intuition of how this work faces and targets Knowledge Management, however, let us be more precise. Research in Knowledge Management has evolved substantially in the past 30 years, coming from a centralized view of KM processes to a distributed view, grounded in organizational and cognitive sciences studies that point out the social, distributed, and subjective nature of knowledge. The first Knowledge Management Systems (KMSs) were centrally based and followed a top-down design approach. The organization managers, supported by knowledge engineers, collected and structured the contents of an organizational memory as a finished product at design time (before the organizational memory was deployed) and then disseminated the product, expecting employees to use it and update it. However, employees often claimed that the knowledge stored in the repository was detached from their real working practices. This led to the development of evolutionary methods, which prescribe that the basic KM system is initially developed and evolves proactively in an on-going fashion. However, most of the initiatives are still based on building central repositories and portals, which assume standardized vocabularies, languages, and classification schemes. Consequently, employees' lack of *trust* and *motivation* often lead to dissatisfaction. In other words, workers resist on sharing knowledge, since they do not know who is going to access it and what is going to be done with it. Moreover, the importance attributed to knowledge may give an impression that these central systems

take away a valuable asset from his or her owner, without giving appreciable benefits in return.

The problems highlighted in the previous paragraph may be attenuated or even solved if a *top-down/bottom-up strategy* is applied when proposing a KM solution. This means that the solution should be sought with aim at organizational goals (top-down) but at the same time, more attention should be given to the *knowledge holders* and on the natural processes they already use to share knowledge (bottom-up). Being active agency such an important principle of Constructivism, this work recognizes that the **Agent Paradigm** (first defined by Artificial Intelligence and more recently adopted by Software Engineering) is the best approach to target Knowledge Management, taking a technological and social perspective. Capable of modeling and supporting social environments, agents is here recognized as a suitable solution for Knowledge Management especially by providing a suitable *metaphor* used for modeling KM domains (i.e. representing humans and organizations) and systems. Applying agents as metaphors on KM is mainly motivated by the definition of agents as cognitive beings having characteristics that resemble human cognition, such as *autonomy, reactivity, goals, beliefs, desires,* and *social-ability*. Using agents as human abstractions is motivated by the fact that, for specific problems, such as software engineering and knowledge management process modeling, agents may aid the analyst to abstract away from some of the problems related to human complexity, and focus on the important issues that impact the specific goals, beliefs and tasks of agents of the domain. This often leads to a clear understanding of the current situation, which is *essential* for the proposal of an appropriate solution. The current situation may be understood by modeling at the same time the overall goals of the organization, and the needs and wants of knowledge holders.

Towards facilitating the analysis of KM scenarios and the development of adequate solutions, this work proposes *ARKnowD* (Agent-oriented Recipe for Knowledge Management Systems Development). Systems here have a broad definition, comprehending both technology-based systems (e.g. information system, groupware, repositories) and/or human systems, i.e. human processes supporting KM using non-computational artifacts (e.g. brain-

stormings, creativity workshops). The basic philosophical assumptions behind ARKnowD are: a) the interactions between human and system should be understood according to the constructivist principle of *self-construction*, claiming that humans and communities are self-organizing entities that constantly construct their identities and evolve throughout endless interaction cycles. As a result of such interactions, humans shape systems and, at the same time, systems constrain the ways humans act and change; b) KM enabling systems should be built in a bottom-up approach, aiming at the organizational goals, but understanding that in order to fulfill these goals, some personal needs and wants of the knowledge holders (i.e. the organizational members) need to be targeted; and c) there is no "silver bullet" when pursuing a KM tailoring methodology and the best approach is combining existing agent-oriented approaches according to the given domain or situation.

This work shows how the principles above may be achieved by the integration of two existing work on agent-oriented software engineering, which are combined to guide KM analysts and system developers when conceiving KM solutions. Innovation in our work is achieved by supporting top-down/bottom-up approaches to KM as mentioned above. The proposed methodology does that by strongly emphasizing the earlier phases of software development, the so-called *requirement analysis activity*. In this way, we consider all stakeholders (organizations and humans) as agents in our analysis model, and start by understanding their relations before actually thinking of developing a system. Perhaps the problem may be more effectively solved by proposing changes in the business processes, rather than by making use of new technology. And besides, in addition to humans and organizations, existing systems are also included in the model from start, helping the analyst and designer to understand which functionalities are delegated to these so-called artificial agents. In addition to that, benefits as a result of the application of ARKnowD may be also attributed to our choice of using the proper agent cognitive characteristics in the different phases of the development cycle.

With the main purpose of exemplifying the use of the proposed method-

ology, this work presents a socially-aware recommender agent named *KARe* (Knowledgeable Agent for Recommendations). **Recommender Systems** may be defined by those that support users in selecting items of their need from a big set of items, helping users to overcome the overwhelming feeling when facing a vast information source, such as the web, an organizational repository or the like. Besides serving as a case for our methodology, this work also aims at exploring the suitability of the KARe system to support KM processes. Our choice for supporting knowledge sharing through *questioning and answering processes* is again supported by Constructivism proponents, who understand that social interaction is vital for active knowledge building. This assumption is also defended by some KM theories, claiming that knowledge is created through *cycles of transformation* between two types of knowledge: tacit and explicit knowledge. Up to now, research on KM has paid much attention to the formalization and exchange of *explicit knowledge*, in the form of documents or other physical artifacts, often annotated with metadata, and classified by taxonomies or ontologies. Investigations surrounding *tacit knowledge* have been so far scarce, perhaps by the complexity of the tasks of capturing and integrating such kind of knowledge, defined as knowledge about personal experience and values, usually confined on people's mind. Taking a flexible approach on supporting this kind of knowledge conversion, KARe relies on the potential of social interaction underlying organizational practices to support knowledge creation and sharing.

The global objective of this work is to support knowledge creation and sharing within an organization, according to its own natural processes and social behaviors. In other words, this work is based on the assumption that KM is better supported if knowledge is looked at from a constructivist perspective. To sum up, this thesis aims at:

1) Providing an agent-oriented approach to guide the creation and evolvement of KM initiatives, by analyzing the organizational potentials, behaviors and processes concerning knowledge sharing;

2) Developing the KARe recommender system, based on a semantically

enriched Information Retrieval technique for recommending knowledge artifacts, supporting users to ask and answer to each others' questions.

These objectives are achieved as follows:

- Defining the principles that characterize a Constructivist KM supporting environment and understanding how they may be used to support the creation of more effective KM solutions;
- Providing an agent-oriented approach to develop KM systems. This approach is based on the integration of two different agent-oriented software engineering works, profiting from their strengths in providing a comprehensive methodology that targets both analysis and design activities;
- Proposing and designing a socially aware agent-oriented recommender system both to exemplify the application of the proposed approach and to explore its potential on supporting knowledge creation and sharing.
- Implementing an Information Retrieval algorithm to support the previously mentioned system in generating recommendations. Besides describing the algorithm, this thesis brings experimental results to prove its effectiveness.

Acknowledgements

Culture is said to play an essential part in either motivating or inhibiting Knowledge Management (KM) practices, meaning that the particular norms and behaviors underlying the organizational environment have a direct impact on how people create and share knowledge. The *culture* theme has been part of my main interests for a long time, since my teenage years, when I worked as a volunteer for the AFS Intercultural Programs. And it was also the will to get to know and experience a different cultural background that drove me to take this PhD in the Netherlands. This also explains my deep satisfaction in realizing that this thesis is the result of my interaction with an absolutely multi-cultural group of researchers composed of Brazilians, Greeks, Germans, Italians, Portuguese, Americans, Dutch and Bulgarians. I feel fortunate and privileged to have worked with such bold, brilliant and kind people, and I can do no other than expressing my gratitude to them all.

I start by thanking my promoter Dimitri Konstantas, who believed in me since the earliest stages of my PhD, playing an important part on the definition of the scope of my work, and providing me with interesting insights about what a PhD thesis is all about. I also thank Lora Aroyo for supervising me for three quarters of my PhD. I particularly acknowledge her participation in helping me refine my initial ideas, making them more precise . Additionally, I would like to show my gratitude to all my colleagues of the Architecture and Services of Network Applications (ASNA) group for their friendship and support. I especially thank Val Jones for helping me find this PhD position. I also express my most sincere appreciation to Aart

van Halteren and Marten van Sinderen, who supported me in the last year of my work, particularly guiding me in the complex task of writing this book. And finally, I am immensely grateful to Annelies Klos, who assisted me in a variety of tasks throughout these past four years, making my work easier and more enjoyable.

I feel greatly honored to have Prof. Anton Nijholt, Prof. Betty Collis, Prof. Gerd Wagner, Prof. John Mylopoulos, and Dr. Lora Aroyo in my defense committee. I am thankful to them for their dedication in reading my thesis and participating in my defense.

The starting point of my PhD was the work on the Agent Academy European Project that was for me a great laboratory in which I was able to experiment ideas and learn. I thank all the members of the project, and especially Adamantios Koumpis for his incentives and good humor, and Dionisis Kechagias, Robert Magnus, and Felix Schmid, for the great peer work.

Three outstanding researchers from outside the University of Twente helped me to shape and evolve the contents of this thesis: Gerd Wagner, Anna Perini and Virginia Dignum. Gerd was the one to suggest that I focused on the application of Agents to KM, which then led me to meet both Anna and Virginia. I thank him not only for that, but also for being so available in a crucial moment of my PhD, and for triggering a number of relevant discussions without which this work would not have been possible. Anna and Virginia have been more than collaborators, my true friends and partners, supporting me throughout most of my PhD research. I greatly admire their dedication and effort on doing high quality research, and I feel fortunate to have shared lots of pleasant moments with them.

I thank Anna Perini, Gianluca Mameli and Paolo Traverso for giving me the opportunity to visit the Automated Reasoning Systems (SRA) division at the Technological and Scientific Research Institute (IRST), in Trento, Italy, for four months in the Spring/Summer 2003. The rich and collaborative environment I found there was central for making tangible the main topics targeted in my work. In this regard, I am very grateful to Diego Sona, Amy Soller, Alessandra Molani, Matteo Bonifacio and Roberto Tiella

for their teamwork and great insights. I would also like to show my appreciation to Paolo Giorgini and John Mylopoulos, who gave me the chance to come back to Trento in April 2005 to finish writing my thesis, meanwhile participating in the Tropos Project. Furthermore, I thank Nicola Fazzi, Angelo Susi, the participants of the Tropos PhD lunch meetings, and my friends at the Laboratory of Applied Ontology (LOA) for their contribution in the final stages of my work.

For the work on the KARE system, I am immensely indebted to Diego Sona and Pablo Gomes Ludermir. I am grateful for the unpretentious way Diego shared with me his valuable experience on machine learning and information retrieval. Thanks for the encouragement and the expert assistance. Pablo appeared in a decisive moment of my research, and his enthusiasm and diligence were vital to accomplish the system implementation and experimentation. Very special thanks to him!

I would also like to thank Crediné Silva de Menezes (my eternal tutor), José Gonçalves Pereira Filho, Sandra Bortolon, Davidson Cury, Rosane Caruso and Ricardo Falbo. They are friends from my former University in Brazil and have provided me with great guidance and incentives in a special difficult moment of my work.

I have no words to thank all my family and friends, for contributing in so diverse and peculiar ways for this accomplishment. Some of these friends have greatly contributed to our smooth adjustment to the Netherlands, including Alex Slingerland, Marike Hasperhoven, Maarten Wegdam, Susan ten Buuren, Helen Kool, Gloria Tuquerres, David Bueno, Clever and Kelen Farias, Ciro Barbosa, Marcos Salvador, Remco and Martina Dijkman, Lia and Robin van Steijn, Valeria Carazzai, Sávio Castro, José Laurindo dos Santos, Valerie Gay, Nikolay Diakov, Galina, and Christian Tsolov. Some others came later into my life, making it even more enjoyable, like Katarzyna Wac, Roberta Cuel, Jaqueline Nicolau, Ricardo Neisse, Tiago Fioreze, Stanislav and Vania Pokraev. And finally, some compose our Brazilian (or almost!) family abroad: Junior, Denise, Letícia and Clara Brandão, Leonardo Dardengo, Gustavo Moraes, Fabiano Gonçalves, Luiz Olavo e Luciana Bonino, Mariana Costa, Anton and Thomas Bovens, Peter de Jong, Maura Merson,

João Paulo Almeida, Patricia Dockhorn, Pablo Ludermir, Flavia Machado, Diana Ros, Diego Ríos and Sonia Taborcia. Special thanks to Diego and Sonia for the great work on the design of this thesis cover. Furthermore, I would like to thank all my friends from the Dutch Spiritist Council for their ever being so welcoming and kind, with special mention to Maria Moraes, Dalva Marçal, and Elias and Virgínia Nascimento. I appreciate the work we have done together. And finally, I thank all the friends and relatives who are in Brazil, especially those who came to visit us in our home abroad. Your presence (real or virtual) has made our experience much more pleasant and rewarding.

The love and admiration I feel for my direct family is indescribable. Having Eloi Corveto de Souza and Dalva Silva Souza as my parents is a great blessing. All they taught me about life has been mostly fundamental to facilitate the journey I undertook during these past four years. And I cannot imagine my life without my three younger brothers: Luiz Gustavo, Vítor Estêvão and Marcelo Henrique Silva Souza. I thank them for all the light and joy they bring into my life, and for all I learn from them everyday. I am also very grateful to my grandmas Lydia Mendes Silva (in memorian) and Penha Corvetto de Souza, for their affection and unconditional support.

At last, I must thank the one person without whom none of this would have happened. My husband Giancarlo Guizzardi is my everlasting source of strength, inspiration and happiness. I thank him for being my perfect partner in all life adventures! In fact, I am truly positive this one was just the beginning.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

"Perplexity is the beginning of knowledge."
Kahlil Gibran

This chapter discusses the motivations behind this work, also presenting some background information. With this, it brings our work in line with the developments in Knowledge Management and provides the first flavor of our own views regarding this field. The chapter also describes the objectives of the thesis and the approach used to achieve these objectives.

This chapter is organized as follows: section 1.1 presents the main motivations behind this work; section 1.2 focuses on some background information, thus giving the context in which this thesis has been developed; section 1.3 describes the research questions that guided our research and clearly sets up the objectives of this work; section 1.4 discusses the research approach used to achieve the proposed objectives; and finally, section 1.5 presents the structure of the remaining of this thesis.

1.1 Motivation

An ever changing and competitive market forced people to find new ways to improve their performance. They must maintain themselves up to date, seeking for new knowledge and improving their competences and skills. Concurrently, these changes had similar impacts within organizational settings.

Mainly aiming at staying in business or seeking for higher profits, organizations need support for fostering innovation and boosting production. Knowledge Management (KM) is appointed as a solution for both organizations and individuals to achieve excellence in performance (Alavi and Leidner, 1999) (Nonaka and Takeuchi, 1995) (Wiig, 1994).

KM may be broadly defined as *tools, techniques* and *processes* for the most effective and efficient management of intellectual assets (e.g. products and process documentation, historical records, and information related to organizational member's personal experience and intuition). In other words, KM deals with providing the right information to the right people, at the right time (Fischer and Ostwald, 2001). This task is at the same time supported and challenged by the technological advances brought by the new era of the information society. On one hand, these advances have granted people with more ready access to information. But as a side effect, people must cope with great amounts of information that is constantly being broadcasted over the Internet and through organizational Intranets. In this context, one might ask the following questions: how can one separate valuable from useless information? Assuming that a lot of what is broadcasted may be of use, how to process so much information? How to determine if an information source is trustworthy or not? These are common problems faced both by individuals and by organizations. In addition to that, organizations also face other difficulties, such as: a) guaranteeing an effective knowledge flow among its employees, in order to foster innovation and new knowledge creation; and b) making sure that, in case one employee leaves, his/her knowledge is kept. This thesis tries to prove that Computer Science can play an important role in providing methodologies and information systems, offering a solution to many of these problems.

Although being currently popular to provide assistance to the problems mentioned above, KM systems have often failed to meet their goals, especially due to a general lack of acceptance by the system's users (Pumareja et al., 2003) (Orlikowski, 1992a). This work identifies some of the problems leading to this lack of acceptance, and presents innovative ways to solve them, proposing a *methodology* to guide the development of KM systems,

and developing a *recommender system* that meets the philosophical assumptions we consider essential for enabling effective KM both personally, and within modern organizations.

1.1.1 Challenges of the Information Society

Applewhite (2004) in his report entitled *"The view from the top"* provides a quick assessment of the importance of the Internet in modern society. The report presents the result of a survey made with 40 technology experts (among business top executive, academic institution's managers, researchers, and others) regarding technological advances of the past, present and future. When questioned about the most important technology of the last 40 years, 9 interviewees directly mentioned the Internet, and 4 others made indirect reference to it, providing answers such as "information technology" and "global communication networks". Moreover, these experts predict that in the near future, society will feel even more impact coming from telecommunication and information technology, since these are considered by 16 of the interviewees as the most important technologies for the coming decade.

Examining closer the way information technology has changed people's life in the past few years makes it difficult to ignore the 'gains vs. losses' dichotomy that has emerged from the latest advances. On one hand, information technology has presented people with new and more efficient ways to address important problems in their daily lives. But on the other hand, it has also brought about some serious challenges. Most of these challenges have to do with coping with an incredible amount of information that one is expected to process in one's work and life in general. For example, one may say that electronic mail technology changed the way people communicate for the best. Email is often considered as a very effective way to communicate, since it is fast (i.e. messages arrive at their destination very soon after being sent), asynchronous (i.e. the contacted person does not need to be there at the time the message comes, as in the case with phone calls) and relatively reliable (i.e. messages rarely get lost, and quick replies are generally issued if there is a problem with the receiver's address). However, email has

also caused some time and effort overhead on people's routine, since one is expected to read one's email box at least once everyday, and replies are generally expected to be sent soon (at most one or two days after the message arrived). If one of the above does not happen, the message receiver possibly becomes socially known as inconsiderate, lazy and/or unreliable. So the big question is: has the emergence of email resulted in a *solution* or in a *new problem*? As in general, it has most probably resulted in a little bit of both.

In human history, while trying to solve a problem, human kind has often created others as collateral effects. This is the case with industrialization leading to deforestation and pollution, long use of antibiotics for an acute skin infection causing stomach pain, and why not, with technology making peoples' lives more comfortable in one side and more complex on the other. It seems that this is inevitable simply because humans are not able to predict all problems a solution might bring, and usually learn more by making mistakes than by doing the right thing at the first try. This general case certainly applies to organizational KM, since promising practices and technologies often lead to unforeseen issues, frustrating both direct users and managers of the organization, who usually head such initiatives.

If all collateral effects created with the adoption of a new practice or technology may not be completely eradicated, some of them may be at least predicted. Consequently, such effects may be compensated or even avoided, while benefits resulting from the adoption of specific solutions may be reinforced and sought. For that, KM practitioners must be empowered with an engineering methodology capable of first, *analyzing* the organizational setting in need of support and then, *designing* the desired solution in terms of the adoption of new information systems, or changes in organizational structures and processes. We claim that providing effective KM solutions depends on paying more attention to the actual users of KM system (or participants of KM practices). In this way, we defend a human-centric view of KM instead of a techno-centric one, having in mind that knowledge is primarily the product of human minds, and knowledge owners should thus be the focus of KM enabling processes and information systems.

1.1.2 Towards a Human-centric Knowledge Management Perspective

Business giants such as IBM (Gongla and Rizzuto, 2001) (Garvin, 1993), Xerox (Brown, 1991), SIEMENS and Hewlett Packard (Kankanhalli et al., 2003) are included among the biggest investors in KM. All these organizations share a single objective: enhancing the performance of their workers, while also profiting from their personal knowledge to make organizational processes more efficient. Although aiming at the same targets, these organizations often follow different strategies to attain them. Hansen et al. (1999) classify KM projects into the following two categories:

1. *Codification approach*: based on systematically storing worker's knowledge in repositories and databases, hoping that such knowledge may be reused in the future.

2. *Personalization strategy*: founded on supporting knowledge holders on their natural processes of knowledge exchange, motivating direct person-to-person contacts, and facilitating worker's communication.

Hansen et al. (1999) also point out that while the codification approach is geared towards *reuse* of old knowledge, the personalization strategy is more prone to result in *knowledge creation* and *innovation*. The view defended in this thesis is closer to the position maintained by the adopters of personalization strategies. We are particularly interested in initiatives that provide workers with high degree of *autonomy* on knowledge sharing and on their *active participation* in generating new knowledge. The current working processes supported at companies such as 3M Co. may be given as an example of a success story in this regard. In 3M Co., workers are given the chance to spend 15 to 20 percent of their working hours for personal projects (Karlin, 2004), i.e. new ideas that they have developed by themselves but which can become new company products. This policy has been adopted because the managers of 3M Co. have realized that giving their employees autonomy to explore their passions and exchange knowledge as they wish fosters creativity, thus leading to innovation.

Some research initiatives have realized that often, the knowledge generated in one particular occasion is difficultly transferable to a different context or situation in the future (Bonifacio and Bouquet, 2002). This claim supports the development of practices and information systems used to support the daily processes of knowledge sharing, instead of simply capturing and codifying knowledge. *Groupware, recommender systems, decision support system, content management systems* and *knowledge portals* are examples of system that may be adopted for such purposes. However, we emphasize that the system needs to fit within real activities of the knowledge workers, being adjusted according to the organization's environment

1.2 Background

The results of this work have been accomplished with basis on previous developments in several research topics under the realm of organizational, educational and computer science. Naming the thesis *Agent-Oriented Constructivist Knowledge Management* suggests that *agent-orientation* and *constructivism* are two cornerstones of this work. The subsequent subsections provide a brief discussion on these two topics, thus providing the main context of this thesis.

1.2.1 Constructivism and Knowledge Management

Organizational KM is mainly about learning, but not formally, by the means of a course. Rather, it is about some kind of *unintentional* learning, meaning that it is a natural consequence of people's daily activities and collaboration with peers within organizations. According to Lave et al. (1991), this kind of learning process can be defined as situated learning or legitimate peripheral participation, which they explain as follows:

"The individual learner is not gaining a discrete body of abstract knowledge which he will then transport and reapply in later contexts. Instead, he acquires the skill to perform by actually engaging in the process, under the attenuated conditions of legitimate peripheral participation. This central

concept denotes the particular mode of engagement of a learner who participates in the actual practice of an expert, but only to a limited degree and with limited responsibility for the ultimate product as a whole."

In other words, organizations can be seen as knowledge sharing communities in which people collaborate and exchange knowledge to perform their work activities, consequently learning from each other. By interacting and collaborating, people get involved in a rich interchange of experiences that gradually reduces the doubts each participant might have regarding organizational processes and products, continuously broadening workers' mental models as they 'learn by doing'.

In educational science, different theories have been created with the aim of supporting active and collaborative learning. The most prominent ones are based on the philosophical principles of Constructivism, which can be summarized as follows (Mahoney, 2004):

- *Active agency*: learning involves active participation in the process instead of passive behavior. In other words, individuals do not learn by being instructed but by engaging themselves in an active dialogue with the learning content, with instructors and with peers.
- *Finding Order or Structure*: many human activities are devoted to finding some kind of order or structure for things and processes, i.e. people engage themselves in patterning of experiences by the means of tacit, emotional meaning-making processes.
- *Self construction*: persons live and grow in a living web of relationships, and they are endlessly constructing and reconstructing their personal identity, when interacting with others.
- *Social-symbolic relatedness*: humans cannot be understood apart from their organic embeddedness in social and symbolic systems, and such situatedness affect the way their cognitive models develop and evolve.
- *Lifespan development*: in human life, order and disorder co-exist in lifelong quests for a dynamic balance that is never quite achieved.

Piaget (Piaget and Inhelder, 1969), Vygotsky (1978), Freire (1970) and Papert (1993) are important precursors of Constructivism. According to Piaget, people learn by balancing what they already know and what is novel for them. He has noted the fact that humans are in constantly self-organization (the processes of identity construction and reconstruction mentioned above), and proposed a theory that explains how knowledge is created (genetic epistemology). Studying his work can provide knowledge managers with great insight on how to foster knowledge creation and growth.

Vygotsky has also made important contributions, creating the socio-historical constructivism. The most relevant results of his studies to our work are his discoveries that learning is profoundly shaped by the historical and socio cultural context in which it is carried out; and his claim that collaboration among individuals from different performance levels is essential for learning. These two postulates show us that: a) the culture involving the organizational setting has great impact over how people progress in their working activities; and b) expert employees and newcomers must collaborate in order to improve their work performances.

Finally, Freire brings a humanistic view into the constructivist research, proposing that only dialogue can intermediate the solution for conflicts, and that all voices should be heard, despite people's position and power. His work has been broadly recognized in the educational field, and should now also be acknowledge in the KM area, proposing collaboration (instead of competition) among all organization's members as the best attitude to guarantee success, both for the organization and to each of its employees.

Building over Piaget's constructivism, Papert's construcionism emphasizes the importance of sharing knowledge by the means of concrete artifacts. Papert claims that learning effectively occurs when the learner is engaged in the construction of a shareable artifact. Building something meaningful and sharable leads to a cyclic process of externalizing the knowledge that is in the mind of the learner and internalizing new structures, as a result of the social interaction around this external artifact. This externalization and internalization cycle seems to coincide with Nonaka and Takeushi's Knowledge Management (KM) theory (Nonaka and Takeuchi, 1995). According

to them, there are two types of knowledge: explicit and tacit. The former refers to codifiable components, which can be disembodied and transmitted, while the latter refers to knowledge that is "confined in people's mind", being difficult to articulate and disseminate. Through social interaction and collaboration, tacit knowledge is turned into explicit, and individual knowledge is turned into organizational. Organizational knowledge creation is a result of a continuous and dynamic process of conversion between these two knowledge types.

As indicated in the previous section, this work supports the creation of KM projects highly based on autonomy and peer collaboration. In this thesis, we call *Constructivist KM* the human-centric view on KM, which prescribes that constructivist principles should be taken into account when designing a KM system and/or process. These principles are further defined and illustrated throughout this thesis.

1.2.2 Agent-oriented Knowledge Management

Agents have frequently been proposed as appropriate entities to enable the analysis and design of complex systems, made up of several components that behave autonomously and interact with each other in order to achieve a common objective (i.e. the system's overall functionality) (Jennings et al., 1998) (Wooldridge, 1999) (Wooldridge and Ciancarini, 2001). The social and cognitive (or mentalistic) characteristics of agents are their main strength, turning them into promising constructs to emulate human interaction and rational behavior. The analysis of the current social structures embedded in the organization may lead to more appropriate system proposals. Then, the developed systems enable such structures to evolve in terms of efficiency and performance.

Currently, organizational tasks and processes are often distributed in different divisions and branches of the organization. In addition to that, these processes follow dynamic kinds of control structure, such as those of market or collaborative network societies (Dignum, 2004a). Such characteristics require present organizational structures and processes to be well-understood

and often redesigned. So, designing KM solutions presents both challenges of process re-engineering and of information system design, as they must be shaped to respond to the specific needs of the organizational environment. In fact, many KM systems are abandoned or fall into disuse because of inadequate understanding of the organizational context (Dignum and van Eeden, 2003) (Pumareja et al., 2003). Hence, analysis and design activities claim for adequate modeling constructs, such as those proposed in the agent's paradigm.

Agents in Artificial Intelligence (AI) have been defined as cognitive beings having characteristics such as *goals*, *beliefs*, *commitments* and *claims*, being influenced by studies from different research communities, including economics, sociology, and cognitive science. An agent can be defined as an autonomic entity inhabiting an environment from which it perceives certain events (perceptors), and on which it acts causing changes (effectors) (Wooldridge and Ciancarini, 2001). The behavior of perceiving the environment and acting as a result of such perception defines agent reactiveness. But besides reacting, agents are able to adopt goal-driven behavior, deciding to act on their own (proactively), motivated by their given beliefs about the world and their desires with respect to how they would like the world to be. Moreover, agents may "live" in a community of other agents, interacting with them in several ways, meanwhile pursuing its goals and/or reacting to events (which here include communicative events triggered by incoming messages from other agents).

Recently, research in this area has moved its focus from the individual characteristics of an agent, to the consequences resulting from agents' interactions. This has given life to a new research area known as *Agent Organizations* (Sichman et al., 2005). Work in this area has focused, for example, on: a) the complexness of self-organizing communities (Di Marzo Serugendo et al., 2004); b) on how the organizational structure may affect the behavior of human organizations, and how this understanding might help organizations adapt to changes (Dignum et al., 2004) and c) on modeling organizations (Dignum, 2004a) (Guizzardi and Perini, 2005).

Concurrent to the evolution in organizational models, more appropriate

agent-based abstractions have been developed, allowing the understanding of the organization's social, economic and technological dimensions. Advances in agent societies are often focused on coordination frameworks that enable agents' interaction, in such a way that they autonomously but cooperatively achieve their goals. Some authors classify agent organizations as having more structure than agent societies, having in common the fact that the agents in the system work towards a common overall purpose. In this sense, the main differences between organizations and societies may be given by the emphasis on the decision processes that underlie organizations, making more explicit the division of labor among agents (usually through *roles*) (Dignum, 2004a) (Ferber et al., 2004) (Hubner et al., 2002) (Esteva et al., 2002). However, organizations and societies could also be considered as synonyms, as work on both fields should be targeted at empowering agents with social structures, providing them with more complex abstractions to model and support organizations.

The features highlighted above show that agents are adequate constructs in representing humans in domain models and organizational abstractions. We can profit from the organizational view, defined by the notions of *purpose*, *structure*, *rules* and *norms* (Dignum, 2004a) when modeling systems to be adjusted to organizational processes and practices. Applying agents as human abstractions allows the system developer to abstract away from some of the problems related to human complexity, focusing on the important issues that interfere with specific *goals*, *beliefs* and *commitments* of the domain agents in each modeling activity. This allows the developer to clearly understand the current situation, and this is an essential factor for the proposal of the appropriate solution. Moreover, such kinds of models make communication with the stakeholders much more effective, since the developer uses concepts that are more familiar to the common user (e.g. goal, task and belief) than technology-oriented terminology (like tables, SQL query, middleware and threads).

Applying agents as a metaphor on system development is not new and has been observed in (Jennings et al., 1998) (Wooldridge and Ciancarini, 2001). However, especially in KM domains, agent organizations seem to be

an interesting approach as agents may represent not only artificial beings, but also the *human users* and the *organizations* involved in a given scenario (Guizzardi et al., 2004b) (Dignum, 2004a) (Perini et al., 2004). This allows, for example, the requirement engineer to understand, before modeling a KM system itself, how knowledge flows within the organization. As a result, besides introducing new technology, the business processes used in the organization may be changed in order to enhance these knowledge flows. Moreover, if a technological solution is needed, agents enable legacy systems to be considered in the analysis, allowing the new solution to be based on approaches of integration of old and new components. This may lead to more satisfaction to end users, who are already familiar with the interface and methods applied in the systems in use.

1.3 Thesis Scope and Objectives

As aforementioned, this thesis considers KM to be more appropriately supported if knowledge is looked at from a constructivist perspective. In other words, people (both individually or as organizational members) are the driving force behind knowledge creation. Hence, when developing KM systems, personal requirements should be taken into account. On the other hand, the research community should provide methodologies and systems to mediate negotiation between people and organizations, considering the general organizational intentions behind the KM activities to enable the accomplishment of these goals.

1.3.1 Research Questions

The motivations behind this work led to the elaboration of the research questions presented in Table 1.1. These research questions are described in detail in the context of the objectives of this thesis, presented in the subsequent section.

1.3.2 Objectives

Taking the previously stated research questions, the global objective of this work may be summarized as:

Developing innovative ways to support knowledge creation and sharing within an organization, according to its inherent culture and processes, according to what we call Constructivist KM.

Research Questions
RQ 1: Can agents be suitably used as metaphors to model human organizations, supporting the creation of Constructivist KM?
RQ 2: How can a comprehensive methodology be tailored in order to guide the analysis and design of appropriate KM information systems and/or practices?
RQ 2.1: Which of the agent cognitive concepts should be considered in each development activity?
RQ 2.2: How can agent's cognitive concepts be materialized in concrete elements of a system?
RQ 3: What are the requirements for a recommender system supporting Constructivist KM?
RQ 3.1.: How can social and cognitive aspects involving organizational members be used in the creation of knowledge recommendations?
RQ 3.2: Which agent-oriented architecture should be proposed in order to provide users with recommendations in a non-intrusive way?
RQ 3.3: Which technique must be used in the creation of recommendations?

Table 1.1: Thesis Research Questions

This main objective is further detailed as follows:

1. Propose an agent-oriented approach named *Agent-oriented Recipe for Knowledge Management Systems Development (ARKnowD)* to guide the creation and evolvement of KM solutions.

2. Develop a recommender agent named Knowledgeable Agent for Recommendations (KARe) as a case study of the ARKnowD methodology, but also aiming at exploring this system's ability to effectively support Constructivist KM.

Let us focus on the first objective listed above. To the best of our knowledge, currently there is no engineering methodology specifically tailored for KM settings, comprehending all development activities. One of the main contributions in this area, proposed by (Dignum, 2004a) is focused on the analysis activity but lacks support for the detailed design of KM systems. Furthermore, our methodology provides greater strength to the initial stages of requirements analysis. This is motivated by our claim that an effective solution should be based on a deep understanding of the organizational potentials, behaviors and processes concerning knowledge sharing.

As suggested by research question RQ 1, this work aims at assessing the appropriateness of the agent-oriented paradigm for the analysis and design of Constructivist KM practices and enabling technologies. Our appropriateness criteria regards understanding if agents can explicitly capture the entities, relations and behaviors characterizing human organizations, allowing a deep analysis of the organizational processes and culture. In particular, such criteria concerns the determination of what principles characterize a Constructivist KM environment, and the investigation of how well the concepts underlying agents enable modeling and reasoning about such principles.

However, having the right abstraction is not enough for guaranteeing the development of adequate solutions for the organization. For that, a consistent KM engineering methodology is needed (RQ 2), granting developers with a set of modeling constructs besides methodological guidelines regarding which modeling activities to perform and which techniques to apply. In particular, we seek at grasping which cognitive (or mentalistic) notions characterizing agents should be applied in each development activity, and how they can be defined (RQ 2.1). Moreover, in order to be implemented in a system, such concepts must become concrete, giving the developer a clear sense of how the previous theoretical definitions may be used in practice

(RQ 2.2).

Our second objective regards the development of an agent-oriented recommender system named KARe. This takes us to RQ 3, where we ask which are the requirements for such a system, specifically aimed at supporting Constructivist KM. Related to this question, we are particularly interested in three distinct focus points. The first one regards grasping the social and cognitive aspects that generally characterize the members of an organization, and how these aspects can improve recommendations (RQ 3.1). The second concerns the development of KARe's agent-oriented architecture (RQ 3.2). Answering this question requires the definition of which agents compose the system and how they interact to provide recommendations. Finally, the third focus point targets the implementation of the recommendation mechanism, which demands the development of an effective technique to assist the system agents in finding from the information maintained by the community of users, the one that responds to a particular knowledge request (RQ 3.3).

1.4 Research Approach

Figure 1.1 presents an overview of the research approach undertaken in this work.

Our first step towards accomplishing the objectives setup in the previous section regards the definition of the Constructivist KM building blocks. Intuitively, we understand that constructivism is in line with the KM view we aim at supporting. However, beyond intuitions, this work presents a detailed description of which principles characterize a Constructivist KM supporting environment. We define such principles based on comparing and contrasting the theories of the constructivists early mentioned in section 1.2.1 and some prominent KM theories, coming from the organizational sciences. Here, we refer to these principles as *building blocks* as they can be seen as the raw material for the development of KM systems and practices complying with Constructivist KM.

Following, we develop the ARKnowD methodology that when applied to

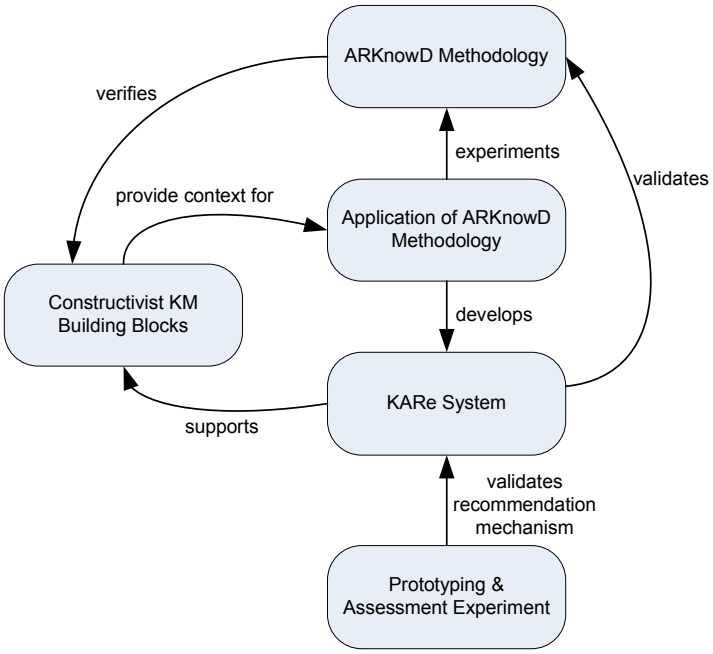


Figure 1.1: Research approach

model an organizational environment, is able to verify to which extent this environment supports the Constructivist building blocks. This is why Fig. 1.1 states that the building blocks provide the context for the application of ARKnowD's methodology, i.e. they work as a checklist to be consulted in the methodology enabled analysis. Included in the development of the methodology is the understanding of which constructs ARKnowD should comprehend, which scenarios of use it should target, and what activities and life cycles it should follow. Particularly, the methodological constructs reflect the concepts comprehended in an ontology we present for the agent-oriented domain. This ontology enables us to evaluate, adjust and combine the notations adopted in ARKnowD.

Aiming at validating the proposed methodology, we apply ARKnowD to analyze a fictitious scenario specifically tailored to illustrate some of the main KM challenges. The scenario is built to reflect a realistic problem domain, according to existing KM literature. Following ARKnowD, the scenario is analyzed, and changes in the environment are proposed, both regarding KM practices and the adoption of enabling technology, more par-

ticularly the KARe recommender system. Still using ARKnowD, we design KARe, prompting it for implementation. This way, KARe consolidates the validation of the ARKnowD methodology, showing that it is able to take the developer from a detailed domain analysis to a consistent design activity.

Besides exemplifying the use of the methodology, the KARe recommender system is a contribution in itself. KARe is proposed and designed taking the Constructivist KM building blocks into account, thus providing support to these earlier identified principles. Finally, a prototype of the system is implemented with the main purpose of assessing its core recommendation algorithm. The prototype comprehends the core functionalities that lead to the creation of knowledge recommendations. These functionalities are obtained by developing an Information Retrieval technique to recommend knowledge artifacts that satisfy incoming knowledge requests. An experiment has been developed to assess our technique in comparison with a standard approach. Through this experiment, we are able to assess the effectiveness of our approach. The remaining features of the system, along with scalability and usability issues are left as future work.

1.5 Thesis Structure

The findings of this thesis are reported within the six remaining chapters organized as follows:

- **Chapter 2** provides state of the art on existing theories about Knowledge Management, Constructivism and Agents, stating the Constructivist KM building blocks and motivating the need for a KM development methodology.
- **Chapter 3** introduces the ARKnowD methodology, describing it in its full extension, including its underlying concepts, scenarios of applicability, activities and life cycle.
- **Chapter 4** presents the requirements analysis of the organizational domain described by the chosen scenario, coming finally to the pro-

posal of the KARe system to support knowledge sharing in the given domain.

- **Chapter 5** presents the main description of the proposed system, also focusing on its architecture and design models.
- **Chapter 6** describes KARe's recommendation mechanism, presents an assessment experiment performed to evaluate such mechanism, and provides details about two implemented prototypes.
- **Chapter 7** concludes this thesis, by presenting a discussion of the main results of this work and describing our research agenda for the future.

Chapter 2

Theoretical Framework

"One of the most tragic illnesses of our society
is the bureaucratization of the mind."
Paulo Freire

The main objective of this chapter is to set the conceptual basis of this work. Having this in mind, this chapter describes state-of-the-art on Knowledge Management (KM) and on the Agent-oriented paradigm. In addition to that, it discusses how Constructivism has influenced our views and choices regarding: a) how to target knowledge management; and b) what kind of methodological and technological support should be provided to enhance KM.

This chapter is organized as follows: section 2.1 introduces the chapter; section 2.2 presents KM background work, including definitions, a discussion on KM systems, a description of the KM main challenges, and a presentation of some KM theories from an organizational science point of view; section 2.3 describes some constructivist theories that greatly impacted the views and choices underlying this work; section 2.4 present the state of the art on agent technology; and finally, some discussions are presented in sections 2.5 and 2.6, respectively focusing on developing a constructivist view on KM and on the development of agent-oriented support to *Constructivist KM*.

2.1 Introduction

Knowledge is today recognized as one of the most important assets of competitive businesses (Alavi and Leidner, 1999) (Nonaka and Takeuchi, 1995) (Wiig, 1994). In other words, organizations realized that the quality of their products and services depend on the effective use of the knowledge that is created and shared by its members. For this reason, the organization should be able to establish an environment that favors the creation of knowledge and innovation. This usually involves the development of a knowledge sharing culture that enables a good flow of knowledge among all members of the organization.

According to Nonaka and Takeuchi (1995), organizations should be tuned to knowledge creation that is defined as "a process that organizationally amplifies the knowledge created by individuals and crystallizes it as a part of the knowledge network of the organization". This is especially important in today's market when work force is highly dynamic and mobile, to guarantee that knowledge is kept within organizational boundaries when employees leave it. Moreover, such processes are aimed at making sure that people in all organizational points-of-action have the information they need for their particular activities, despite of being geographically distributed or working in different organizational units (Wiig, 1994).

Alavi and Leidner (1999) remind us that the concept of capturing and communicating knowledge in organizations is not in itself a novelty and has long been accomplished through training, employee development programs, and access to organizational documentation, such as reports and manuals. However, KM adds the dimension of potentially using enabling information technologies (such as the Internet, Intranets, data warehouses, data filters and software agents) to support the systematic creation, integration, and dissemination of knowledge. It is also important to note that contrarily to training, KM aims at conveying knowledge to people in an informal way, rather than doing it in formal classes or tutorials. That is why KM is said to lead to unintentional learning (Lave et al., 1991), embedded in organizational practices, policies and routines.

The informality of knowledge transmission does not imply, however, that this happens as a pass of magic. On the contrary, it requires the organization to actively invest time and resources on creating a conducive environment for knowledge creation and sharing. Organizational reengineering, the adoption of specific methods such as "total quality management" and "organizational learning techniques" can partially deal with separate parts of this challenge (Wiig, 1994). However, new methodologies specifically tailored to understand the current organizational conditions, subsequently proposing changes in practices besides technology adoption are also essential for effective KM to take place.

This work claims that agents have a great potential to support Knowledge Management (KM), serving as rich metaphors to enable the analysis of organization's scenarios, and moreover functioning as building blocks for KM systems' development. In fact, the suitability of agents for KM support has been recognized elsewhere, both in theory (Dignum, 2004b) (van Elst et al., 2004), and in the development of agent-oriented systems (Abecker et al., 2003) (Gandon et al., 2002) (Preece et al., 2001). However, this debate is hardly finished. Our work takes advantage of the mentalistic notions supported by agents, such as *autonomy, reactiveness, proactiveness* and *social ability* (Wooldridge, 1999). In general, we propose that agents can represent besides systems, human beings and organizations, allowing the analysis of their interrelationships, behavior and motivations, before an actual KM solution can be proposed. In this way, we hope to be able to adjust KM practices and systems to the organization's particular requirements, aspect of concern of many KM researchers (Wiig, 1994) (Davenport and Prusak, 1998). We argue that it is important to understand where and how agent's mentalistic notions can be used to help system developers to abstract away from unimportant matters, while focusing on the right concepts in each development activity. Although various agent-oriented methodologies target these notions in different ways, a deep discussion about their conceptualization and practical use concerning information systems development is currently an open issue.

KM practices can take different forms and follow diverse points-of-view.

Here, we take *Constructivism* as a theoretical background to be closely followed when proposing a new KM enabling process and/or system. This comes from the realization that KM is ultimately about learning. And moreover, that the constructivist paradigm is largely compliant with the kind of learning KM wants to accomplish: unintentional, situated, and based on active engagement by the one who learns. Constructivism also alerts that regardless of position within the organization, all members should be seen as potential sources of knowledge. In other words, according to this paradigm, learning and knowledge sharing are more effective when non-hierarchical, idea that has been previously defended by knowledge managers and theorists in the area (Orlikowski and Gash, 1994). In this chapter, we discuss how constructivist theories can aid us in shaping KM practices and systems, allowing us to understand some of their general requirements beforehand.

2.2 Knowledge Management

We start by providing a general overview about the KM field. We focus on the basic definitions underlying this field (section 2.2.1) , and slowly progress towards the exploration of some themes related to how it can be supported. In this realm, we present a discussion about KM systems, with a few examples (section 2.2.2). Then, we discuss the main challenges on providing KM solutions (section 2.2.3). And, finally, we present some theories from the viewpoint of the organization sciences that comply with our vision regarding KM (section 2.2.4).

2.2.1 Definitions

Before we understand how KM may be facilitated, it is important to have a clear picture of what knowledge actually is. Fields related to Information Processing usually define knowledge in contrast to the related definitions of *information* and *data* (Alavi and Leidner, 1999) (Davenport and Prusak, 1998). Data is raw by nature and thus, does not inform one of anything, except if it is analyzed and interpreted. A sentence like “Beatrixstraat” and

a number like "121" are examples of data. When data is analyzed and interpreted, it becomes information. For instance, knowing that "Beatrixstraat" refers to a street in the city of Enschede (The Netherlands) and that "121" refers to a house number, one suddenly has information about an address. When information is authenticated and contextualized, including indications of how it should be applied in action, it finally becomes knowledge. Figure 2.1 illustrates the transformation of data into information, and information into knowledge.

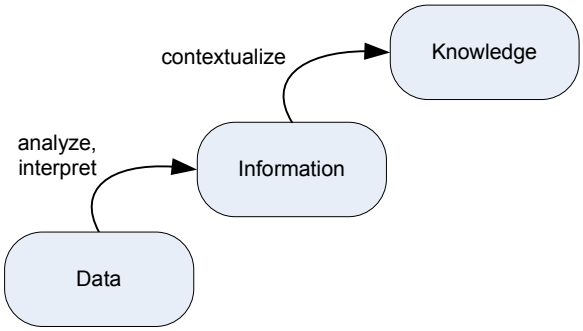


Figure 2.1: The relationship between data, information and knowledge

According to (Alavi and Leidner, 1999), such definition is limited, as it presumes a hierarchy from data to information, and from information to knowledge, with each varying along some dimension, such as context, usefulness, or interpretability. Instead, they argue that *"knowledge is information possessed in the mind of an individual: it is personalized or subjective information related to facts, procedures, concepts, interpretations, ideas, observations and judgments (which may or may not be unique, useful, accurate, or structurable). (...) information becomes knowledge once it is processed in the mind of an individual (...). This knowledge then becomes information again (...) once it is articulated or communicated to others in the form of text, computer output, spoken, or written words or other means."* (pg. 6) Consequently, Alavi and Leidner differentiate between information and knowledge analogously to Nonaka and Takeuchi's definitions of *explicit* and *tacit knowledge* (Nonaka and Takeuchi, 1995). These two knowledge types are defined as follows:

- *tacit knowledge* has to do with personal values, intuitions, and experience, usually embodied in the person's actions and choices. Thus, this kind of knowledge is hard to formalize and communicate;

- *explicit knowledge*, as the name suggests, refers to knowledge that is codifiable and transmittable to others.

More philosophical views on knowledge define it as "justified true belief" (Audi, 1998), meaning that knowledge is a belief that can actually be verified to be true in the objective reality, and for which we can provide a reason (or justification). This definition is founded on the empiricist view of epistemology[1], which claims that the only source of knowledge is sensory experience. Such view is challenged by rationalist traditions that argue that true knowledge can be attained by reasoning, hence not being necessarily justified by sensory experience. Epistemology studies how knowledge emerges through its relations with human abilities, such as perception, memory, reflection, introspection and testimony (Audi, 1998).

Whatever view seems more appealing, rather than debating knowledge definitions, our main goal here is understanding what it represents to organizational contexts and how it can improve organizational products and processes. In this sense, knowledge can be seen at the same time as a **basic ingredient** and a **product** of everyday work within an organization. As an ingredient, it can be regarded as **personal belief** and **information** that enhances the ability of an individual to **make decisions** and **take effective action**. And as a product, it can be seen as the direct result of organizational members' actions, usually embedded in their own practices and/or in their work outcomes.

Note that, although considered as a concrete organizational asset, knowledge behaves in a very different way when compared to physical and natural resources. Instead of being bound to finish like natural resources, knowledge can be replicated and expanded endlessly, enabling wide reuse besides new

[1]Epistemology is the philosophy of knowledge, focusing on what knowledge is, how it is created and how it relates to other concepts such as memory, belief, perception and reasoning (Audi, 1998).

knowledge creation. Moreover, when an individual possessing it shares it with others, he or she does not give it away, as in the case of physical resources. Instead, knowledge is retained by both parties: the one who gives it away, and the one who gets it (Allee, 1999).

Having all this in mind, KM can be defined as a systematic process for acquiring, organizing and communicating both tacit and explicit knowledge to all members, enabling them to be more effective and productive in their work (Alavi and Leidner, 1999). This process is based on practices and technologies that motivate knowledge exchange, so that knowledge can be replicated and amplified to be used in all points-of-action within the organization.

Fischer and Ostwald (2001) defines KM as a cyclic process composed of three main activities: creation, integration and dissemination of knowledge. *Knowledge creation*, as the name suggests, is the activity that leads organizational members to generate new knowledge; *knowledge integration* refers to converting it into a sharable technological format, while also connecting it to existing knowledge within the organization; and finally, *knowledge dissemination* enables access of specific knowledge to all employees and units that need to apply it in practice.

According to Nonaka and Takeuchi (1995), teams are a cornerstone of effective KM, working as a shared context for social interaction, which is essential to knowledge creation and dissemination. New ideas and points of view are created through dialogue and discussion. This dialogue can involve considerable conflict and disagreement, but it is exactly such conflict that motivates organizational members to question their assumptions, making sense of their experience in a new way. Consequently, such interactions often lead to the transformation of personal knowledge into organizational knowledge. Lave et al. (1991) and Wenger (1998) have also emphasized the role of teams and communities in the knowledge sharing process. A wider discussion on this topic is found in section 2.2.4.

2.2.2 Knowledge Management Systems

Knowledge Management Systems (KMSs) have evolved in the past 30 years, coming from systems based on central repositories of knowledge built by knowledge engineers to distributed systems, which grant the users with full autonomy over knowledge exchange. Following this evolution, three phases can be distinguished.

The first phase is characterized by central-based systems. The organization managers, supported by knowledge engineers, collected and structured the contents of an organizational memory as a finished product at design time (before the organizational memory was deployed) and then disseminated the product, expecting employees to use it and update it. Such approaches were top-down in that they assumed that management creates the knowledge and that workers receive it (Fischer and Ostwald, 2001). The organization employees often disliked such approaches and deserted the systems, because:

- workers claimed that the knowledge stored in the repository was detached from their real working practices;
- the work of constantly updating the knowledge base was seen as extra work and as a burden.

These claims led to the development of evolutionary methods to build KM systems, starting the second phase of the KMSs evolution. According to these approaches, the basic KM platform was initially developed and evolved proactively in an on-going fashion (Hahn and Subramani, 2000). The system users were viewed as stakeholders in the platform development, participating in the elicitation of requirements that would more closely relate to their daily activities. An example of this is the "Seeding, Evolutionary Growth, Reseeding (SER)" process model, developed to understand the balance between centralized and decentralized evolution in sustained development of large systems (Fischer and Ostwald, 2001). However, as most of the initiatives were still based on building central repositories and portals,

issues of trust and motivation often led to the abandonment of the systems (Dignum, 2004a) (Pumareja et al., 2003). In other words, workers resist on sharing knowledge, since they do not know who is going to access it and what is going to be done with it. Moreover, the importance attributed to knowledge may give an impression that these central systems take away a valuable asset from his or her owner, without giving appreciable benefits in return.

Currently, a new trend has been inaugurated: Distributed Knowledge Management (DKM), initiating a new phase in the development of KMSs. This modality of KM recognizes the users as owners of their knowledge, prescribing that they should decide the means and conditions for knowledge exchanged (Dignum, 2004a). The idea at the basis of DKM (Bonifacio and Bouquet, 2002) is supporting the integration of autonomously managed nodes, without forcing the creation of centralized repositories, indexes or shared ontologies. Proposals for distributed knowledge management systems can be found in (Bonifacio et al., 2004) (Guizzardi et al., 2004a) (Yu and Singh, 2002). This new phase is also largely characterized by the recognition that knowledge cannot be separated from the communities that create it, use it, and transform it. This realization has motivated organizations on supporting and even fostering Communities of Practice (CoPs), i.e. groups of workers who share similar interests, personal affinity and trust.

Common technologies used to implement central organizational memories are large relational databases or data warehouses, the latter providing more complex reasoning capability over knowledge (O'Leary, 1998). With the growing interest for Internet-based applications, much of these repositories gained web-based interfaces, commonly known as enterprise knowledge portals (van Elst et al., 2004). Groupware technology, such as the Lotus Notes platform, has also been largely applied in the development of KMSs (Pumareja et al., 2003) (Orlikowski, 1992a). Slowly, the system's underlying technology has shifted to more flexible structures, culminating in the use of software agents (Abecker et al., 2003) (Gandon et al., 2002) (Preece et al., 2001) (Yu and Singh, 2002). This paradigm continues to be frequently researched and applied in the third phase of KMSs evolution, which also in-

troduced the use of peer-to-peer architectures to support DKM (Bonifacio et al., 2004) (Guizzardi et al., 2004a). In the following subsections, we describe some types of KMSs. We do not intent here to provide a complete list, but solely to discuss some of the most prominent examples of KM enabling technology.

Organizational Memory Systems & Content Management Systems

Organizational Memory (OM) can be defined as the means by which past knowledge is made available for current activities, enabling the organization to act more effectively. It includes organizational goals, plans, handbooks, manuals, and standard operating procedures (Chen et al., 2003). Systems that support the creation and maintenance of an OM are known as OM systems (OMSs). They are generally built over advanced database technologies (such as data warehousing, data mining, and knowledge extraction) and network technologies (especially Intranet and Internet-based technologies) (Lehner et al., 1998).

Advanced database and network technologies have created the possibility to store, retrieve and share large amounts of data. But the storage and retrieval of information is hardly the biggest problem in this domain. Rather, eliciting and contextualizing useful knowledge is the greatest challenge (Chen et al., 2003). According to (Conklin, 1997), an OM composed only of "formal" knowledge (such as manuals, client information and procedures) is essentially an immense heap of disconnected things, a giant organizational attic. More valuable OMs are the ones that document more tacit kinds of knowledge, such as why decisions have been taken, the content of informal communication, and the result of particular actions and choices. In this realm, it is typical to include in the OMSs, records on *lessons learned* and *best practices* regarding specific projects and/or procedures. The former refers to negative experiences, including choices made and reasons for failure, in order to avoid similar mistakes in the future; conversely, the latter documents success stories that are encouraged and expected to be repeated

(O'Leary, 1998).

OM systems often combine advanced network and database techniques with different types of technology, for example, case-based reasoning to support *experience management*. Delaitre and Moisan (2000) propose case-based reasoning applied to the OM to support risk management in hazardous situations, such as fire-fighting; while Henninger (2001) uses this reasoning technique to allow software developer companies to standardize development methodologies. Both cases rely on previous experience stored in the OM and adapt them as a potential solution to a problem at hand.

More recently, a new trend has emerged to provide richer knowledge contextualization methods: content management systems. These systems often use taxonomies and ontologies as a means to provide a conceptualization of the OM, besides classifying the knowledge items it contains (Davies et al., 2003a) (Bonifacio et al., 2004). Taxonomies refer to hierarchy of concepts, usually visualized in a tree-structure. An ontology is a shared conceptualization of a domain, composed of a set of concepts and relations. This model provides semantics about a specific domain, i.e. by looking at the model, one is able to understand how the model's developer (a person or a community) interpret that specific domain. Ontologies have recently been in the spotlight, especially with developments related to the SemanticWeb (Davies et al., 2003b)

Examples of content management systems are Ontoshare (Davies et al., 2003a) and KEEx (Bonifacio et al., 2004). Ontoshare adopts a shared ontology between a community of practice, and encourages the community members to annotate documents using RDF, linking them to the shared ontology. Instead of relying on a centralized conceptualization KEEx, knowledge assets are assigned to concepts in taxonomies named "contexts". KEEx support building and mapping of several contexts, which can be either collective or individual.

Groupware

Groupware systems are computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment (Ellis et al., 1991). Examples of groupware are workflow management systems, email systems, chat applications, shared whiteboards, co-authoring systems, group calendering and scheduling systems, collaborative virtual environments and conferencing systems (Farias, 2002).

Different classifications have been created in order to facilitate understanding of the wide variety of existing groupware. Table 2.1 presents a taxonomy of groupware systems based on the distribution of collaborating team members in time and space (Ellis et al., 1991). It classifies groupware applications according to whether they support work at the same place and time, at the same place but at different times, at different places but at the same time, or at different places and different times. Thus, following these considerations of place and time, presented in Table 2.1, electronic meeting room technology, for example, would fit within the upper left cell. On the other hand, a physical bulletin board could be placed within the upper right cell. Video conferencing belongs in the lower left cell and an email system in the lower right cell.

	Same Time	**Different Time**
Same Place	face-to-face interaction	asynchronous interaction
Different Places	synchronous distributed interaction	asynchronous distributed interaction

Table 2.1: Groupware taxonomy based on time and space

The importance of the use of groupware to enable KM can be understood by at least two factors:

1. Groupware supports social interaction and collaboration, considered essential for knowledge sharing (Nonaka and Takeuchi, 1995).

2. The knowledge exchanged through groupware applications (such as emails or chat applications) is maintained in messages and/or log files,

remaining stored for future reference and use.

This recognition has lead to the adoption of groupware in practice in several organizations. Besides general email systems, the most prominent example of groupware is IBM Lotus Notes, largely adopted in organizational intranets (Orlikowski, 1992a) (Robertson et al., 2000) (Pumareja et al., 2003) (Sumner, 1999).

Decision Support Systems & Expert Systems

Decision Support Systems are computer systems based on reasoning techniques to support the decision making process. They are usually rule-based and are suitable for semi-structured or unstructured and unanticipated types of decisions. Expert Systems are special kinds of Decision Support Systems, which capture the knowledge of an expert in a narrow domain of knowledge (Luger, 2005). The main aim of Expert Systems is to simulate the problem-solving behavior of an expert in his/her domain of expertise. These systems have been applied in a number of different areas such as medicine, for supporting diagnosis Pedersen (2004), education, to support learning by simulating human tutors (Doyle et al., 1996), and agriculture, for agricultural management and irrigation control (Hassan et al., 2004).

Decision Support Systems and Expert Systems are very similar in nature. They are usually composed of a) a *knowledge base* containing factual knowledge; and b) an *inference engine*, which applies the knowledge contained in the knowledge base to solve a specific problem.

Prior to its storage on the knowledge base, knowledge needs to be represented in a computational form. The most common knowledge representation technique is the use of rules, i.e. condition-action pairs that indicate that if the condition is satisfied, the indicated action should be taken. Besides rules, frames are also largely applied. Typically, a frame consists of a list of properties of the entity and associated values for those properties.

After knowledge has been captured and stored, the inference engine can now execute its problem-solving strategy, reasoning over the available knowledge. Different strategies may be applied, such as forward or backward

chaining. Forward chaining starts by processing a set of conditions and goes towards the conclusion, while in backward chaining, a conclusion is stated (for instance, the desired outcome) and the path to that conclusion is then inferred from the rules.

When compared to the other types of systems here presented, this class of systems is highly based on AI techniques. Thus, Decision Support Systems and Expert Systems are much more complex to develop than Groupware and Content Management Systems. When building Expert Systems, for instance, a knowledge engineer needs to capture the knowledge of an expert and integrate it into the system before it is deployed. Except in cases in which machine learning techniques are used to derive new knowledge from the one previously integrated, the knowledge base keeps static throughout time. Contrarily, Groupware and Content Management Systems are usually fed by the users while the system is already running. Furthermore, the nature of of expert knowledge is another source of complexity for this kind of system. "Expert knowledge is a combination of a theoretical understanding of the problem and a collection of heuristic problem-solving rules that experience has shown to be effective in the domain." (Luger, 2005, pg. 21). Usually, in order to produce effective results, Expert Systems require knowledge from a well-studied domain, which has clearly defined problem-solving strategies. Despite these complexity issues, the applications of this class of systems for Knowledge Management is clear, as it can support managers and workers of an organization to take flexible decisions, based on knowledge priorly acquired by other experts in their area of action.

Recommender Systems

Recommender systems support users in selecting items of their interest or need from a big set of items, helping users to overcome the overwhelming feeling when facing a vast information source, such as the web, an organizational repository or the like. This kind of systems has become very popular in the late 1990s, especially due to the popularization of the Internet and the consequent danger of information overload. Today, as the number of users

of the information society grows rapidly, recommender systems are not less needed.

Having in mind that KM refers to "providing the right people with the right piece of knowledge, at the right time", it becomes evident that recommender systems can be of much value in organizational settings. Workers can rely on recommender systems to find out specific information, or to look for people who would know what they need.

Recommendations may be based on similar items to those a given user has liked in the past (content-based recommendation); or on items owned by users whose taste is similar to those of the given user (collaborative recommendation) (Balabanovic and Shohan, 1997). Both types of recommendation may be quite beneficial in Knowledge Management communities, where knowledge is distributed and, thus, the knowledge one needs may be hard to find and sort out.

Besides the difference given by the aforementioned recommendation approaches, recommender systems are also differentiated by (Montaner et al., 2003): the items they recommend (systems have been developed to recommend web pages (Balabanovic and Shohan, 1997), movies (Good et al., 1999), etc.); the nature of the user models they use to guide the recommendations (e.g. history of items accessed by the user, topics indicating user interest, etc.); the recommendation techniques (mainly, how the user model is represented, what kinds of relevance mechanisms are used to update the user model, and which algorithm is used to generate recommendations); and the recommendation trigger, i.e. whether the recommendation is started by the user or by the proactive behavior of the system.

Enterprise Knowledge Portals

Rather than a different kind of KM system, Enterprise Knowledge Portals (van Elst et al., 2004) can be seen as a friendly web-based interface to serve as a unique access point for all KM tools used in an organization. As organizations adhere more and more to the KM economy, adopting new tools and applications, such as diverse groupware, different content management

systems, and various decision supporting systems, it may be difficult for organization's members to keep up with so many knowledge sources. Thus, Knowledge Portals come to support the integration of these diverse sources, facilitating knowledge access to all members.

Knowledge portals are also popular as a virtual "home" for communities of practice. They collect the community's memory, guide newcomers on accessing content and tools, and often advertise communities activities and accomplishments, both for members and outsiders. Examples of knowledge portals are ECOT (Brazelton and Gorry, 2003) and Knowledge Board [2].

2.2.3 Main Challenges

As reported in the previous section, technology presents a variety of means to support knowledge capture, structuring and sharing. However, problems with the acceptance and use of KM systems continue to be reported. These systems are often abandoned or, in some cases, not explored in their full potential to improve the work performance of organizational members and keep knowledge from leaving the organization. Most of the challenges related to these problems seem to have sociological nature. These problems profoundly influence how people see and use the adopted KM systems.

The KM literature has mentioned several times that the efficacy of KM processes and systems are very much impacted by organizational culture (Allee, 2000) (Orlikowski and Gash, 1994) (Nonaka and Takeuchi, 1995) (Alavi and Leidner, 1999). As in culture in general, organizational culture is given by the common sense knowledge, accepted behavior, and cultivated values within the organization. There are often reports on the fact that the adopted KM systems are based on architectures and methods that reinforce old pernicious habits and power structures, instead of creating new and beneficial dynamics (Newell et al., 1999) (Orlikowski and Gash, 1994). This shows the need for analyzing the current organizational setting, including cultural habits and values, in order to propose a solution that reflects the changes that the organization require, in contrast to reassuring the existing

[2] http://www.knowledgeboard.com

vices.

One of the most common problems related to knowledge sharing refers to the fact that organizational environments lean toward competition rather than collaboration. Consequently, while KM requires true collaborative behavior, work recognition and promotion are usually based on competition among colleagues (Orlikowski, 1992a). This means that having a specific piece of knowledge may be the differential that one needs to climb the next step in his/her career. This problem alone can prevent organizational members to volunteer their knowledge. Reviewing evaluation methods to favor collaboration, and creating an incentive program for knowledge sharing are often cited as good practices aimed at enhancing the appeal of knowledge sharing within organizations.

The fear of making mistakes is another cited obstacle for knowledge exchange (Orlikowski, 1992a). Generally, people feel that they can explain things better if they talk directly to the person who needs their knowledge. They also fear that a static description of their expertise can be misleading, as knowledge is highly dynamic and difficult to transfer to different situations. This is a challenge for system developers, for allowing a piece of knowledge to be contextualized and connected with past knowledge and experiences. It is also a reminder that the organization should create an environment which stimulates learning, and is tolerant to mistakes. In fact, mistakes can be valuable triggers for knowledge creation. This is recognized by Garvin (1993), who describes situations involving corporate giants such as IBM and Boeing, in which great successes have been achieved by learning from past failures.

Related to the fear of making mistakes is the suspicion that something unethical or inappropriate can be made using one's knowledge. This may be characterized as lack of trust, a common reason for keeping one's knowledge to oneself. Conducive environments for social interaction and network building, such as communities of practice (refer to section 2.2.4) may be able to overcome this problem. As for systems, those that provide privacy and accessibility options controlled by the user are favorable in these situations.

Lack of time also scores high in the reasons for not using KM systems, as does the fact that searching for knowledge requires great effort while useful knowledge is seldom available (Pumareja et al., 2003) (Orlikowski, 1992a). These problems stem from the fact that the use of KM Systems are often imposed by top management without an actual revision in the working processes underlying the organization. In order to guarantee that the KM system is going to fulfill its promises, it is thus necessary that a comprehensive analysis of the organizational processes is made, leading to changes in these processes in order to better accommodate the use of the adopted technological solution.

Besides these common cited challenges, there can be domain dependent or organization specific problems. An example is cited by Desouza (2003), who describes a case involving a software engineering company. In this company, software engineers regretted and often avoided being considered an expert in specific languages or methods. Known experts in this organization would be continuously allocated to participate in projects related to that specific language or method, which would soon become boring and repetitive. Thus, something that is usually viewed as a symbol of status and recognition, in this particular organization was seen as a burden. This demonstrates that an effective KM solution, both in terms of processes and systems, can only be attained case by case, after the current organizational culture and processes are clearly understood.

2.2.4 Theoretical Background

A few KM theories from the point-of-view of the organizational sciences comply with our views on KM and thus assisted in shaping our work. Here, we summarize these theories, aiming at describing their main claims and ideas, so as to provide a flavor of how this thesis approaches KM.

The Knowledge Management Spiral

Nonaka and Takeuchi (1995) describe the creation and evolvement of knowledge throughout the organization using a spiral metaphor. According to this metaphor, innovation is generated by cyclic conversions between tacit and explicit knowledge, as defined in section 2.2.1. Rather than a phenomenon that is confined in one's mind, this conversion happens through social interactions between people engaged in actions in a common environment.

There are four different modes of knowledge conversion:

1. *Socialization* (from tacit knowledge to tacit knowledge): a process of sharing experiences and thereby creating tacit knowledge such as shared mental models and technical skills. This process generates what Nonaka and Takeuchi named sympathized knowledge.

2. *Externalization* (from tacit knowledge to explicit knowledge): a process of articulating tacit knowledge into explicit concepts, metaphors, analogies, hypotheses or models. The type of knowledge resulting from this process is called conceptual knowledge.

3. *Combination* (from explicit to explicit knowledge): a process of systemizing concepts into a knowledge system, involving the combination of different bodies of explicit knowledge. New knowledge is then obtained through sorting, combining and categorizing existing information, resulting in systemic knowledge.

4. *Internalization* (from explicit knowledge to tacit knowledge): a process closely related to "learning by doing". When experiences through socialization, externalization, and combination are internalized into individuals' tacit knowledge bases in the form of shared mental models or technical know-how, they become valuable assets. The new knowledge obtained through this process is called operational knowledge.

The knowledge spiral with its four knowledge conversion modes is illustrated on figure 2.2.

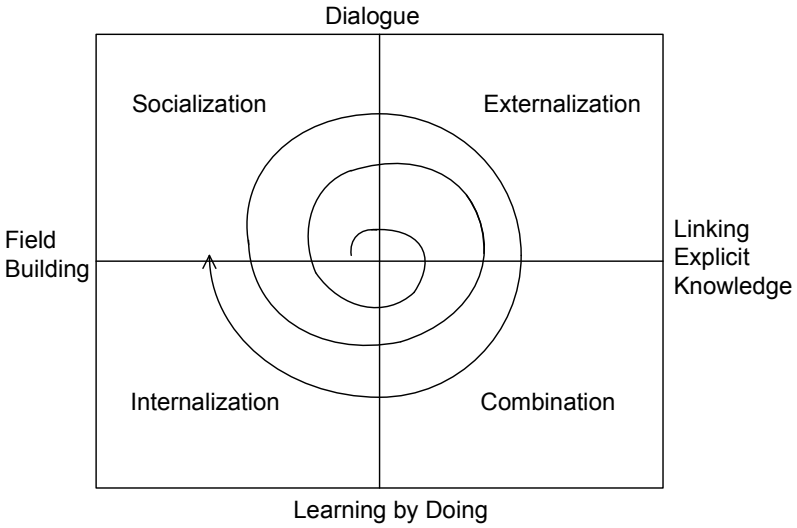


Figure 2.2: The Knowledge Spiral (Nonaka and Takeuchi, 1995)

When tacit and explicit knowledge interact, an innovation emerges. For instance, let us consider an externalization/internalization situation, happening in a conversation between two people. When a person explains something to another, he/she is obliged to externalize some of his/her knowledge, while the other person processes internalization. However, the knowledge the second person internalizes is not the same tacit knowledge of his/her interlocutor. The internalized knowledge gains a new interpretation based on the listener's own personal values and experiences. In the conversation process, the two individuals negotiate meanings, compare their understandings, and eventually generate new ideas based on their combined views.

While knowledge can only be created by individuals, the organization has the role of motivating and supporting creative processes, maintaining an environment that is appropriate for innovation. In other words, the organization has to mobilize tacit knowledge created and accumulated at the individual level, amplifying this knowledge through the four modes of knowledge conversion, and crystallizing it at the collective level. When performing this role, there are five conditions an organization should guarantee to promote the knowledge spiral:

- *Intention*: defined as an organization's aspiration to its goals, it is

the driving force behind the knowledge spiral. To create knowledge, business organizations should foster their employees' commitments by formulating an organizational intention and proposing it to them. This can be achieved by stating a mission or creating strategies that can motivate employees to get involved in knowledge conversion activities.

- *Autonomy*: all members of an organization should be allowed to act according to his/her own wishes, as far as the circumstances permit. Autonomy leads to unexpected opportunities and motivates individuals to create and share knowledge.

- *Fluctuation* and *Creative Chaos*: fluctuation refers to an order whose pattern is hard to predict at the beginning, although different from complete disorder. If organizations adopt an open attitude toward environmental signals, they can exploit those signals' ambiguity, redundancy, or noise in order to improve their own knowledge systems. Chaos is generated naturally when the organization faces a real crisis, such as rapid decline of performance due to changes in market needs or significant growth of competitors. But it can also be generated intentionally when the organization's leaders try to evoke a sense of crisis among organizational members by proposing challenging goals.

- *Redundancy*: refers to intentional overlapping of information concerning business activities, management responsibilities, and the company as a whole. Sharing redundant information promotes the sharing of tacit knowledge, because individuals can sense what others are trying to articulate. Redundancy is especially important in a stage in which a new concept is being development, when it is critical to articulate images rooted in tacit knowledge. Redundancy also facilitates the interchange between hierarchy and non-hierarchy. Finally, it provides the organization with a self-control mechanism to keep it moving in a certain direction. There are several ways to build redundancy into the organization. One is to adopt an overlapping approach in which different functional departments work together in a "fuzzy" division of labor ("rugby style"). Another way is through a "strategic rotation"

of personnel, especially between vastly different areas of technology or functions such as R&D and marketing. Besides, redundancy can be achieved by the use of appropriate technology which maintains the same piece of knowledge available in different points of action.

- *Variety*: an organization's internal diversity must match the variety and complexity of the environment in order to deal with challenges posed by the environment. To maximize variety, everyone in the organization should be assured of the fastest access to the broadest variety of necessary information, going through the fewest steps.

Communities of Practice and Situated Learning

Recently, organizations have recognized communities of practice as a potential strategy to enable effective knowledge creation and sharing (Allee, 2000) (Wenger, 1998). This understanding is motivated by the observation that knowledge cannot be separated from the communities that create it, use it, and transform it. Communities of practice (CoPs) can be defined as informal groups of workers generally gathered based on similar interests, common work, personal affinity and trust. What holds these workers together is a common sense of purpose and a real need to know what each other knows (Allee, 2000).

Being in the same department or organization can be beneficial but is not a strong requirement for the formation of such communities. On one hand, not all people grouped together in the same space or unit form a CoP. On the other hand, a CoP can extrapolate the boarders of departments and organizations, connecting people that are in several units and geographically dispersed. According to Wenger (1998), what defines a CoP is the combination of three distinct dimensions: *mutual engagement, joint enterprise*, and *shared repertoire*. Mutual engagement concerns the shared practices among the participants of a community, i.e. the actions in which they are engaged, in a constant negotiation of meanings and behaviors. Membership to a community thus depends on being involved in these same actions and negotiation processes. By joint enterprise, the author means the common

objectives and mutual accountability shared by community members, as a result of the collective processes in which they are involved. Joint enterprise controls the community's rhythm, and functions as a tool for coordination and sense-making. And finally, shared repertoire refers to the resources created or used in the course of the community's existence. These resources include routines, words, tools, procedures, stories, symbols, actions, and concepts.

In the perspective of organizational work, what makes these environments appealing is the fact that the dynamics of the community lead people to naturally share knowledge and learn from each other. Lave et al. (1991) argue that learning is situated, i.e. it happens as a product of activity, context and culture. Rather than asking what kinds of cognitive processes and conceptual structures are involved, *situated learning* focuses on what kinds of social engagements provide the proper context for learning to take place. CoPs can provide this context, being an ideal environments for the enhancement of performance of old-timers, while at the same time supporting newcomers to engage into organizational practices.

According to the situated learning theory, people are part of CoPs that embody a set of beliefs, norms and behaviors (i.e. culture), where they acquire and exchange knowledge. Lave et al. (1991) claim that this process leads to legitimate peripheral participation, which concerns the process by which newcomers become part of the community. In the beginning, a newcomer is in the periphery of the community. As the newcomer moves from the periphery to the community's center, he/she becomes more active and engaged within the culture, hence assuming the role of expert or old-timer. At the same time, legitimate peripheral participation also takes into accounts the transformation of communities of practice, as a result of their members' interaction and practices.

Situated learning is very much related to the idea of apprenticeship, where a learner evolves his knowledge and abilities by assisting an expert in doing his work. In contrast with formal education, in apprenticeship settings, the learner does not gain a discrete body of abstract knowledge, which can be transported and reapplied in later contexts. Instead, he/she acquires the

skill to perform by actually engaging in the process.

It is important to note that CoPs cannot be forcibly created, but they may be fostered, by acquiring from the organization the means to grow and mature within working settings (Gongla and Rizzuto, 2001) (Dignum and van Eeden, 2003). Dignum and van Eeden (2003) emphasize the importance of setting up real targets to communities of practice, guaranteeing their value for the organization to be concretely perceived and measured. In addition to that, fostering also includes creating the conditions for a community to emerge, both giving social and technological support for it. In the social dimension, community members can, for instance, be rewarded and remembered. As for the technological support, an appropriate infrastructure needs to be provided to facilitate knowledge sharing.

Distributed Knowledge Management

Distributed Knowledge Management (DKM) (Bonifacio and Bouquet, 2002) has recently appeared as an attempt to provide an alternative to centralized systems, which often lead to abandonment or misuse. According to DKM proponents, KM is generally faced and pursued according to an objectivistic epistemology, in which knowledge can be expressed following an objective and general codification (based on a supposedly shared conceptualization and understanding between organizational members). Besides, it is also assumed that knowledge can be shared and reused disconnected from the individual or community that has created it. In addition to that, traditional organizational models and paradigms of control are usually privileged, leading to centrally managed knowledge bases, containing knowledge that once made explicit, is property of the organization.

Contradicting this view, DKM focuses on the social and subjective nature of knowledge. According to this theory, an organization is formed by multiple units named *Knowledge Nodes*, defined as individuals, teams or communities that have different terminology and working practices. These Knowledge Nodes should be allowed to locally manage their own knowledge, thus having knowledge property and autonomy with respect to sharing it

with others. KM becomes a problem of coordinating these multiple sources of knowledge in a distributed way. So, while sharing knowledge, the source's terminology (or interpretative schema) should be communicated, and perhaps translated to the destiny's terminology, thus providing the context in which that knowledge was created. In summary, these assumptions can be summarized in two principles (Bonifacio and Bouquet, 2002):

- Principle of Autonomy: each unit (person or group) should be granted a high degree of autonomy to manage its local knowledge;
- Principle of Coordination: each unit must be enabled to exchange knowledge with other units not by imposing the adoption of a single, common interpretative schema but through a mechanism of mapping other units' context onto its context from its own perspective.

In order to enable knowledge sharing, technology should be shaped reflecting these principles, resulting in a distributed architecture.

2.3 Constructivism

Knowledge Management is indirectly aimed at learning, as one of its main objectives is to allow members of an organization to evolve in terms of gaining knowledge and developing skills. This suggests that a special attention should be given to understanding and analyzing how knowledge emerges in human thinking, acting and interacting with the world.

Note that, here, we refer to learning processes undertaken without particular educational intervention, but rather unintentionally, embedded in people's daily routine. Complying with this view, Constructivism is based on the active participation of individuals in the construction of their own knowledge, instead of being instructed. In this sense, the person's cognitive system cannot be seen as an empty vessel to be filled in. Instead, each new element assimilated by the cognitive system will be contrasted and combined with the already existing elements, according to the understandings of that particular individual.

Although agreeing on a few general principle (e.g. active and autonomous participation of learners in the learning process, learner's self-construction, etc), different theorists have presented their own views on Constructivism, focusing on different elements to explain how knowledge emerges. Here, we summarize the main contributions of four thinkers that have profoundly influenced how we view knowledge creation in this work. Later in this chapter, we present a general discussion, comparing and contrasting the described contributions, and relating them to the KM theories discussed in section 2.2.4.

2.3.1 Jean Piaget: Genetic Epistemology

Piaget's theory (Piaget and Inhelder, 1969) suggests that knowledge is originated by a continuous construction and elaboration of new cognitive structures through a central cognitive equilibration process. In general, a modification in the environment generates a perturbation in the individual's cognitive system, taking it out of a state of virtual equilibrium. As this cognitive system is conditioned to seek equilibrium, it tends to stabilize although real equilibrium can never be completely achieved. New stabilization, given by another temporary cognitive equilibrium state, depends on assimilation and accommodation processes. First, assimilation allows that an external element is incorporated in the individual's conceptual schemas. Subsequently, this new element is accommodated in the existing mental model, considering the characteristics of this element and contrasting it with what the individual already knows. This process is illustrated in Figure 2.3.

Figure 2.3 depicts the cognitive balancing process as a spiral. The line in the center of the spiral represents the real equilibrium that although sought, is never actually achieved by the individual. In several moments (indicated in the figure by black dots), the individual experiences a perturbation triggered by modifications in the environment with which he/she interacts. This is followed by phases of assimilation and accommodation that finally leads to another virtual equilibrium state. In a following moment, there is a new perturbation, triggering another cycle. At each stage, the individual

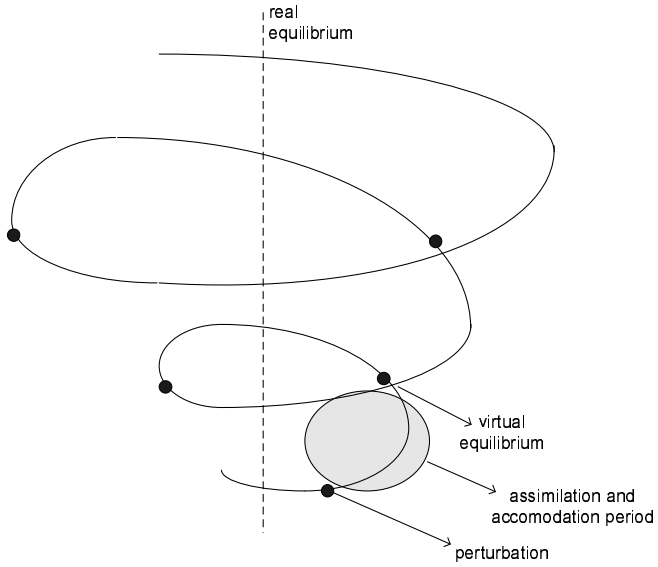


Figure 2.3: Cognitive balancing process

broadens his/her mental model, which is represented in the figure by the growing spiral arcs.

Although not extensively focusing on social interaction, Piaget has shown its importance for learning (de La Taille, 1992). According to him, the stages of development of the logical operations correspond to the correlative stages of social development. In a child's sensorimotor stage, there is no socialization of intelligence, i.e. knowledge sharing, which can only begin with language acquisition. However, even after learning how to speak, in the preoperational stage, some characteristics prevent children to establish effective knowledge sharing. On one hand, the child lacks the ability to commit to a common system of meanings with other individuals. On the other hand, he/she does not necessarily maintain his/her points of view throughout the dialogue. And finally, the child is not able to place himself/herself in the point of view of the other, and thus reciprocity relations are not established. These three characteristics form what Piaget called egocentric thinking. Starting from the operational stage, with the emergence of logical thinking, the child overcomes egocentric thinking and is finally able to establish a real dialogue, by expressing his/her individual points of view while contrasting it with the perspectives of others.

2.3.2 Lev Vygotsky: Social-historic Constructivism

Vygotsky (1978) has emphasized the role of culture and social interactions in learning processes. For him, culture has a great impact in the development of one's intelligence. It is through the interaction with family members and other persons of his/her environment that a child creates his/her conceptualizations and develops his/her mental abilities. This is clear in Vygotsky's following statement:

"Every function in the child's cultural development appears twice: first, on the social level, and later, on the individual level; first, between people (interpsychological) and then inside the child (intrapsychological). This applies equally to voluntary attention, to logical memory, and to the formation of concepts. All the higher functions originate as actual relationships between individuals." (Vygotsky, 1978, pg. 57)

According to this Russian psychologist, humans are capable of individually constructing knowledge up to a certain level, called level of real development. But a person can go beyond this level, to the level of potential development, if helped by a more capable peer, usually a parent, a teacher, or a more experienced peer. The difference between these two levels is what Vygotsky called the zone of proximal development. Vygotsky's theory is illustrated in figure 2.4.

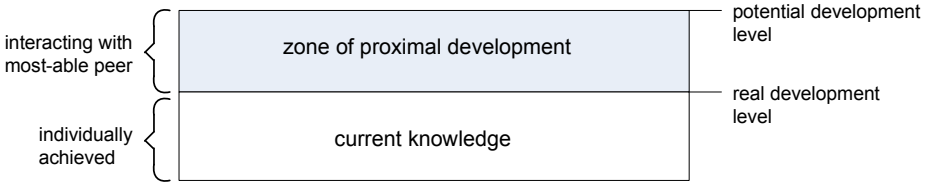


Figure 2.4: Illustration of Vygotskys theory

2.3.3 Seymour Papert: Constructionism

Papert has been one of the first thinkers to realize the potential of computer technology in the development of intelligence (Papert, 1993). In the sixties, he proposed the use of computers as an educational strategy, and created

the programming language named LOGO to enable children to build and explore graphics and simulations as instruments of learning and enhancing creativity.

The core idea in the Constructionist theory proposed by Papert is illustrated in figure 2.5. He claims that individuals learn more effectively when engaged in the construction of something external and sharable, such as a sand castle, a robot, a computer program, or a book. This leads to a cycle of internalization and externalization of knowledge, motivated by the construction of an external object that has meaning to the individual and to those that surround him/her.

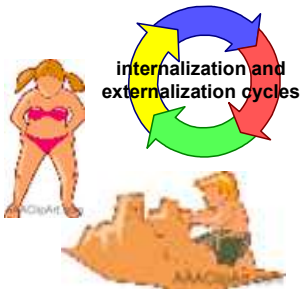


Figure 2.5: Knowledge internalization and externalization cycles motivated by the construction of a sharable and concrete artifact

2.3.4 Paulo Freire: Dialogue enables Learning

Paulo Freire is one of the most prominent educators of our century. His experience has included teaching children and adults from different backgrounds and social levels. He has emphasized that the learning process is not instructional but rather dialogical (Freire, 1970). In this sense, instructor and pupil should be viewed at the same time as educators and learners, respecting each other by what they know, and gaining new knowledge together by negotiating content and strategies. Given that KM is informal rather than curriculum-based, his claim is especially relevant for our purposes.

Freire has proposed that the main purpose of gaining new knowledge is to enable critical thinking and the active engagement of the learner in the

construction and enhancement of his own reality (Freire, 1970). For that, he has argued that the educational process should have its foundation in the consciousness of the day-to-day situations lived by the learners. Thus selecting themes that are related to the pupils' daily lives can lead to better results than choosing subjects that have no attachment to their reality.

One of Freire's particular claims regards the real gains of motivating learners to ask questions. According to his Pedagogy of Question (Freire and Fagundez, 1992), a question is the first "knowledge sparkle". By this means, one is able to externalize some of his knowledge at the same time as reflecting about what information he/she possesses in contrast to what is missing. In fact, it is through question and answering that we solve many of our problems in daily life, being it at work, at family encounters, or in other endeavors. And besides, questions are also the beginning of any scientific work, in which finding relevant research questions express the maturity of the one seeking for a scientific breakthrough. It is important to note that questioning and answering here are not merely part of an intellectual game, but rather an important ingredient of the word-action-reflection triad. In this sense, reflection leads to the expression of doubts that are directly connected to action.

Different cultural aspects can impel or constrain the act of questioning. A democratic environment is more appropriate than an authoritarian one. According to Freire and Fagundez (1992), authoritarianism prevents curiosity, since questioning can be seen as challenging authority. In addition to that, dialog requires an environment where mistakes are tolerated and even valued as a means for learning and improving performance.

2.4 The Agent-oriented Paradigm

Winograd (1995) claims that in the mid 1990's, there was a shift in software development from a programming to a design centered view. By then, software engineers had realized that it was more difficult to understand which functionality a system should exhibit than correctly codifying functionality

in a programming language. The biggest effort should then be put on modeling the interactions between system and stakeholders, rather than on coding and debugging software. This observation triggered the adoption of many different software development approaches and methods, from the object-oriented software engineering paradigm through participatory methods and prototyping practices.

The agent-oriented paradigm comes again as one more step in the evolution of software engineering approaches (Jennings et al., 1998) (Parunak, 2000). Jennings et al. (1998) claim that agents have become so attractive due to their ability to naturally and easily characterize a variety of applications. We agree with this view. Agents are more able than objects to represent active entities of a domain or a system. In fact, our world is composed of active and passive entities. For instance, in an organizational setting, an employee is an active entity, while the resources he uses for his work are passive. Similarly, we can conceive a system composed of active entities (agents) that manipulate a number of passive resources or information entities (objects) to accomplish their tasks. Therefore, especially with respect to organizational software, the agent paradigm presents much more powerful abstractions to analyze and model the complexities and idiosyncrasies of the organizational setting. It allows the view of organizations, humans and software systems as intentional entities that interact on the pursuit of both common and individual goals, and on the execution of tasks.

The main characteristic that distinguish an agent from an object is its capacity to act autonomously (Jennings et al., 1998) (Parunak, 2000). This work takes the definition of autonomy proposed by Jennings et al. (1998). According to this definition, to say that agents are autonomous means that (to some extent) they have control over their behavior and can act without the intervention of humans. Objects do have control over their state, but not of their behavior. In other words, once an object B invokes a method of object A, the method is executed. In agent-based systems, actions are executed by request, i.e. an agent B must issue a request to agent A, which then decides whether or not the action fits its own internal motivations, before executing it.

The agent paradigm has been shaped by developments from several research areas, such as distributed computing, object-oriented systems, software engineering, artificial intelligence, economics, sociology, and organizational science (Jennings et al., 1998). Still today, two main viewpoints on agents can be identified in the agent research community.

- The *Software Engineering perspective* takes agent as a powerful metaphor to develop software. In this perspective, the software system can be thought of as a group of active entities (agents), each one having its own goals and behavior. The sum of this "more simple" behavior gives the multiagent systems (MAS) overall complex behavior. Consequently, MAS are referred to as more than the sum of its parts.

- The *Artificial Intelligence* perspective emphasizes the intelligent and flexible behavior of agent, characterizing it as an autonomous entity, capable of both reactive and proactive (go-driven) behavior, and social ability.

Although we acknowledge the importance of the Artificial Intelligence research, and especially those targeted at understanding the mentalistic concepts attributed to agents, such as beliefs, desires, intentions (Rao and Georgeff, 1991) and commitments (Castelfranchi, 1995), here we take a Software Engineering view. This choice is justified by our main concern of exploring the potentials of the agent-oriented approach for the analysis, design and development of KM enabling systems.

2.4.1 Agents' Definitions and Attributes

There is not a consensus on the definition of agents and their attributes. Weiss (1999) states that agents are autonomous computational entities, which can be viewed as perceiving their environment through sensors and acting upon their environment through effectors. Figure 2.6 shows an abstract top-level view of an agent. It shows that the agent takes sensory input from the environment, and produces actions that affect this environ-

ment as output. The interaction is usually an on-going, non-terminating one (Wooldridge, 1999).

Figure 2.6: An agent interacting with the environment (Wooldridge, 1999)

As there is no agreement on the definition of agent, the same happens regarding intelligence. For Wooldridge (1999), an agent is called intelligent if it can act flexibly towards achieving its goals, which mean that the agent must be reactive, pro-active, and have social ability. These three characteristics are described as follows:

- reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it;
- pro-activeness: intelligent agents are able to exhibit goal-directed behavior by taking the initiative;
- social ability: intelligent agents are capable of interacting with other agents.

Other researchers, such as Sen and Weiss (1999) think that a system being considered as intelligent is also expected to be able to learn. According to them, learning can be defined as the acquisition of new knowledge and motor and cognitive skills and the incorporation of this knowledge and skills in future system activities, provided that this acquisition and incorporation is conducted by the system itself and also leads to an improvement in its performance. In other words, saying that an agent has learning ability means that the agent is going to improve its future behavior, based on knowledge and skills it has acquired in past experiences.

Agents may also exhibit other attributes, such as mobility, veracity, benevolence and rationality. This does not mean that every agent will have all these characteristics. This depends on the nature of its tasks and should be decided by the designer. These attributes are defined as follows (Wooldridge and Jennings, 1995):

- mobility: is the ability of an agent to move around electronic networks;
- veracity: is the assumption that an agent will not knowingly communicate false information;
- benevolence: is the assumption that different agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it;
- rationality: is the assumption that an agent will act in order to achieve its goals and will not act in such a way as to prevent its goals from being achieved.

A system composed of two or more agents that interact in order to achieve a common goal is called a *multiagent system* (MAS) (Wooldridge and Jennings, 1995). In a MAS, it is possible to encounter agents with different levels of intelligence. Depending on the agent's goals, it is necessary to provide it with knowledge and an inference mechanism, so that it can reason and decide how it should act. On the other hand, other agents might have goals that do not require much intelligence, but require that they should be mobile, for example. As mentioned before, the decision about the attributes that an agent must have depends on its internal requirements.

"A MAS can be seen as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver. The characteristics of MAS are: a) each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint; b) there is no global system control; c) data is decentralized; and d) computation is asynchronous." (Jennings et al., 1998, pg. 285)

Note that such characteristics of MAS apply for artificial or human societies. Thus, we infer that a MAS composed only by humans, or by a mix of humans and artificial agents inherit the same kinds of problems described by Jennings et al. (1998) regarding purely artificial MAS. These problems regard, among others, the coordination of actions, the conciliation of multiple intentions, the allocation of limited resources, and the guarantee for a noiseless communication.

2.4.2 Agent-oriented Software Engineering Methodologies and Languages

Agent Technology has received a great deal of attention in the last few years and, as a result, the industry is beginning to get interested in using this technology to develop its own products. The role of agent-oriented methodologies is to assist in all the phases of the life cycle of an agent-based application (Iglesias et al., 1999), although different ones emphasize one or more development activities. In this section, we present some languages and methodologies specifically tailored for the analysis and design of agent-based systems. Whilst a comprehensive review of all methodologies and languages is beyond the scope of this thesis, we here attempt to provide some of the most prominent approaches.

The Gaia Methodology

According to (Wooldridge et al., 2000), the extended approaches based on the object-oriented view fail to adequately capture agent's flexible, autonomous problem-solving behavior, the richness of agents' interactions , and the complexity of agent systems' organizational structures. For these reasons, the Gaia methodology has been specifically elaborated for the analysis and design of agent-based systems. Analysis and design can be understood as a process of developing increasingly detailed models of the system to be constructed. Figure 2.7 shows the Gaia models applied in each of these development activities.

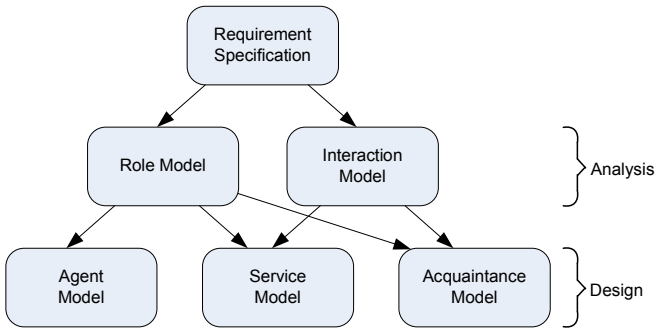


Figure 2.7: The Gaia models (Wooldridge et al., 2000)

For Gaia, a multi-agent system can be viewed as an organization, in which the agents assume different roles. These roles have a certain relationship among themselves and each one of them will participate in systematic interactions with the others. The analysis stage is dedicated for the understanding of this organization, and this is made with the aid of two models: the roles model and the interaction model.

A role is defined by four attributes: responsibilities, permissions, activities and protocols. The responsibilities define the functionalities of the role. The permissions are the "rights" associated with the role to allow it to perform its responsibilities. In other words, the permissions identify the resources that are available for the role, typically information resources. So, a role can be associated to the ability of reading, modifying or generating a certain information. Activities are computations that can be executed by the role alone, without interaction with other roles. On the other hand, protocols define the way the role interacts with other roles, while executing its responsibilities. The roles model can be precisely defined as a set of role schemata, one for each of the system's roles.

There are dependencies and relationships between several roles in a multi-agent organization. This is central in the way the system works. For this reason, the interactions must be captured and represented in the analysis stage. In Gaia, these links between roles are represented in the interaction model.

The interaction model consists of a set of protocol definitions, one for each

role interaction. Here, the focus is in the essential nature and the purpose of the interaction, and not in a precise order of a particular exchange of messages. The protocol definition determines a textual description (proposal) of the interaction, its initiator and the responder, its inputs and outputs and a brief textual description of its process.

The aim of a "classic" design is transforming the abstract models derived during the analysis stage into models at a sufficiently low level of abstraction that they can be easily implemented. This is not the case with agent-oriented design, however. Rather, the aim in Gaia is to transform the analysis models into a sufficiently low level of abstraction that traditional design techniques (including object-oriented techniques) may be applied in order to implement agents. To put it another way, Gaia is concerned with how a society of agents cooperate to realize the system-level goals, and what is required of each individual agent in order to do this. Actually how an agent realizes its services is beyond the scope of Gaia, and depends on the particular application domain.

The Gaia design process involves generating three models: the agent model; the services model and the acquaintance model. The agent model is concerned to documenting the various agent types used in the system, also assigning one or more roles to each type. The services model identifies the services associated to each role. And the acquaintance model defines the communication links that exist between agent types.

The design process can be summarized as follows:

1. creating the agent model: a) aggregating roles in agent types; b) refining it to create an agent hierarchy; and c) documenting the instances of each type;

2. developing the services model, examining activities, protocols and responsibilities of each role;

3. developing an acquaintance model, derived from the interaction and agent models.

ROADMAP

ROADMAP (Juan et al., 2002) focuses on building open systems and emphasizes the societal aspects of an agent system. This methodology extends Gaia by introducing use-cases for requirement gathering, explicit models of agent environment and knowledge, and interaction model based on AUML interaction diagrams. Figure 2.8 presents the models adopted by this methodology.

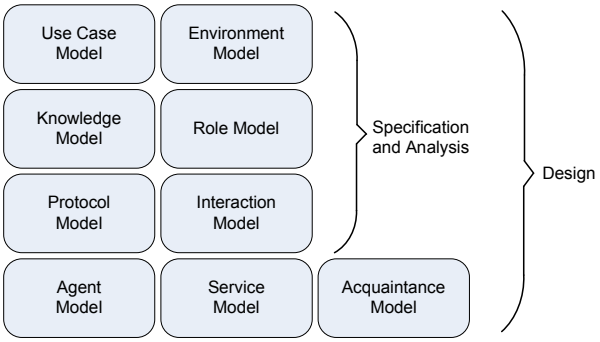


Figure 2.8: The ROADMAP models (Juan et al., 2002)

Traditional *UML Use Case Diagrams* are applied to gather system requirements. These case diagrams are then used to generate two other models: the environment and the knowledge models, which present a general view of the system's environment and knowledge respectively.

The *Environment Model* consists of a tree hierarchy of zones in the environment, and a set of zone schema to describe each zone in the hierarchy. A zone schema includes a text description of the zone, and the following attributes: objects, constraints, sources of uncertainty and assumptions.

The *Knowledge Model* consists of a hierarchy of knowledge components, and a description for each knowledge component. Besides identifying and decomposing knowledge components disposed in a hierarchy, this model is a result of the analysis of these components' life cycles, determining how the knowledge components are generated, consumed and stored.

The *Role Model* extends its correspondent model in Gaia with a role hierarchy. The role hierarchy is represented as a tree of roles, in which leaf nodes of the tree are atomic roles, while the others are composite roles. The

atomic roles retain their original definition and represent characteristics of individual agents. The composite roles are defined in terms of other roles, whether atomic or composite. Roles are defined as in Gaia, having permissions and responsibilities. Besides this, ROADMAP Role schema has two additional attributes: sub-roles and knowledge. The former lists the sub-roles of a composite role, representing the local organization structure. The latter represents local social knowledge, emerging from the interaction of sub-role knowledge. Roles can be changed at run-time given the correct authorization (i.e. a permission to access and modify the definition of other roles). Instead of immutable contract of behavior, roles should be considered as long-term agreement of behavior that can be reasoned and changed. This difference allows a computing organization modeled in roles to be more flexible.

The old Gaia interaction protocols are here subsumed by the *Protocol Model*, which are then refined into AUML interaction diagrams (refer to section 2.4.2), which compose the *Interaction Model*. In this respect, the only additional feature is the representation of zones in the interaction diagrams. With the adoption of AUML interaction diagrams, ROADMAP aims at providing a more flexible and dynamic way to model agent's interactions.

No addition is made in the design models related to the ones proposed in Gaia. Like previously, all models from the analysis are carried out into design, where the *Agent, Service* and *Acquaintance* models are designed for the system.

The ROADMAP methodology provides strong support for engineering complex open systems, but is less suitable for application not requiring these properties. In cases of systems having static nature, simple environment and lack of knowledge, creating the models prescribed by ROADMAP simply causes extra overhead (Juan et al., 2004).

OperA

The OperA methodology (Dignum, 2004a) allows for the formal specification of agent societies (i.e. a system is seen as an organization or society of

agents). OperA at the same time, facilitates the discussion with domain experts that are not knowledgeable in agent theory, and is based on a formal semantics that make verification possible.

OperA modeling approach consists of two main phases: the first phase is dedicated to build an *Organizational Model*, which comprehends the definition of the organizational structure and global behavior of the system; in the second phase, the organizational structure is actually populated by agents, and specific conditions are agreed for their enacting of the organizational roles. This second phase is accomplished through two different models, namely the *Social* and the *Interaction* models.

The *Organizational Model* specifies the organizational characteristics of an agent society in terms of four structures:

- Social structure: specifies objectives of the society, its roles and what kind of model governs coordination.
- Interaction structure: details interaction moments (scene scripts) that represent a society task that requires the coordinated action of several roles; and gives a partial ordering of scene scripts, which specify the intended interactions between roles.
- Normative structure: describes society norms and regulations in terms of role and interaction norms.
- Communicative structure: specifies the ontologies for description of domain concepts and communication illocutions.

In the *Social Model*, the enactment of roles by agents is fixed in social contracts that describe the capabilities and responsibilities of the agent within the society, that is the agreed way the agent will fulfill its role(s). Because the society designer does not control the design and behavior of individual agents, there is a need to verify the actual behavior of a society population. This is done by analyzing the agreements specified in the social contracts. The use of contracts to describe activity of the system allows, on one hand,

for flexibility in the balance between organizational aims and agent desires, and, on the other hand, for verification of the outcome of the system.

In the *Interaction Model*, concrete interaction scenes are dynamically created by role-enacting agents, based on the interaction scripts specified in the OM. Role enacting agents negotiate specific interaction agreements with each other. Such interaction commitments are fixed in interaction contracts. As in the Social Model, interaction contracts allow on one hand for flexibility and personalization of the organizational design, and on the other hand, for the verification of design and activity. That is, it can be verified whether the interaction agreements between a specific population satisfy and are sufficient for the organizational interaction aims specified in the Organizational Model.

The choice for modeling the systems in these two distinct phases, one for modeling the society and the other to enable the population of the society makes OperA very suitable for situations that involve the integration of agents developed by several parts. In fact, OperA assumes that the agents composing the system are already designed beforehand. Thus, OperA consists of a methodology to combine these agents into a system that exhibits coherent behavior and meets the needs defined by the system requirements (Organizational Model), rather than proposing a design methodology which develops the agents from scratch to compose a system.

Tropos

Tropos is an agent-oriented software development methodology for engineering distributed systems (Bresciani et al., 2004). The methodology adopts a model-driven approach, i.e. it guides the software engineer in building a conceptual model, which is incrementally refined and extended, from an early requirements model, namely a representation of the organizational setting where the system-to-be will be introduced, to system design artifacts. Indeed, a distinctive feature of the methodology with respect to current agent-oriented methodologies is that of filling the gap between requirements analysis and system architecture design, by adopting an uniform notation

and an uniform analysis technique to model business goals, system requirements and system architecture.

Tropos uses a conceptual modeling language derived from the i* framework (Yu, 1995), which provides a graphical notation and a set of techniques for goal analysis. This notation has been extended in order to allow for informal and formal specifications. Basic constructs of the conceptual modeling language are those of actor, goal, plan, softgoal, and resource:

- an actor can represent a stakeholder in a given domain, a role or a set of roles played by an agent in a given organizational setting;
- a goal represents the strategic interests of actors. Two basic types of goals are considered, namely hardgoals and softgoals, the latter having no clear-cut definition and criteria as to whether they are satisfied. This difference is captured in (Chung et al., 2000), which suggests to say that (hard) goals can be satisfied, while softgoals can be satisficed. Softgoals are useful to represent how a state of affairs should be reached, that is they can represent goal/plan qualities and non-functional requirements.
- a plan (or task) specifies a particular way of doing something, i.e. a particular course of action that can represent a means for satisfying a goal or for satisficing a softgoal;
- a resource is a physical or informational entity used in a given task or to achieve a certain goal.

A dependency link between pairs of actors allows the analyst to model the fact that one actor depends on another in order to achieve a goal, execute a plan, or acquire a resource. The former actor is called the depender, while the latter is called the dependee. The object (goal, plan resource) around which the dependency centers is called the dependum. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals. In this sense, the depender becomes vulnerable due to its dependency links. This type of information can be graphically

depicted in an actor diagram, a graph whose nodes represent actors (circles) and whose arcs represent dependencies (a couple of arrows linked by its dependum).

The process of model building in Tropos has been specified in (Bresciani et al., 2004) in terms of a non-deterministic concurrent algorithm, here we give a qualitative description. Model building begins with the definition of a number of actors, each with a list of associated main goals (or softgoals). Notice that at the beginning, the minimum set of actor goals which relates to the analysis purpose is explicitly modeled. Throughout the refinement of the model, further goals may be included.

Each root goal is analyzed from the perspective of its respective actor and depicted in a sort of balloon, called the goal diagram. For instance, goal means-end analysis proceeds by refining a goal into sub-goals, plans, and resources that provide means for achieving the goal (the end). Contribution analysis allows the analyst to point out goals and softgoals that can contribute positively or negatively in reaching the goal being analyzed. Decomposition allows for a combination of AND and OR decompositions of a root goal into sub-goals, thereby refining a goal structure. The generated sub-goals are delegated to other actors, or remain a responsibility of the actor itself. Sometimes new actors need to be introduced, to whom some goals and/or tasks are delegated. For instance, in order to represent the role of technology at support of the organization's processes, new actors are introduced, refining a model of the organization's needs into a model of the requirements for an information system able to meet these needs. Softgoal analysis is typically used to drive the choice of a particular choice among different alternatives that may emerge during OR-goal decomposition (Chung et al., 2000). Modeling is complete when all goals have been dealt with to the satisfaction of the actors who pursue them.

Prometheus

The Prometheus (Padgham and Winikoff, 2002) methodology focuses on building agents using BDI platforms and on providing explicit and detailed

processes and deliverables suitable for use by industry practitioners with limited agent experience, or by undergraduates. Having in mind the development of large industrial applications, this methodology is based on an incremental development, which allows constant refinements in artifacts and documentation. Prototyping of skeleton code is enabled during these refinements, allowing for a better understanding of the system's behavior before its actual complete implementation.

Prometheus methodology consists of three activities: system specification, architectural design, and detailed design. The *system specification* activity focuses on identifying the basic functionalities of the system, along with inputs (percepts), outputs (actions) and any important shared data sources. Use case scenarios, adapted from UML, are created to provide a more general view of the interaction between actions, percepts and functionalities.

Architectural design uses the outputs from the previous activity to determine which agents the system will contain and how they will interact. During this stage of the design, it is important to identify the events that the agent will respond to. Agent messages are also identified, forming the interface between agents. Possible shared data objects should also be identified at this stage.

Detailed design looks at the internals of each agent and how it will accomplish its tasks within the overall system. The focus is on defining capabilities (modules within the agent), in terms of internal events, plans and detailed data structures.

Prometheus supports the engineering of conventional closed systems with controlled and trusted agents. It specifically supports the BDI framework, and focuses on functionalities. However it lacks support for advanced properties such as openness and is not suitable for systems requiring these properties (Juan et al., 2004).

AUML

AUML presents an agent as an extension of active objects, exhibiting both dynamic autonomy (the ability to initiate action without external invocation) and deterministic autonomy (the ability to refuse or modify an external request). Other capabilities, such as BDI mechanisms, mobility and explicit modeling of other agents can also be added as extensions to the basic AUML agents (Odell et al., 2000).

The AUML proposal takes UML, which is an accepted formalism in academic and commercial environments and extends and adjusts this language to the context of agents. According to AUML proponents, UML provides tools for modeling many of the concepts regarding agents, such as interaction protocols and internal behavior (Odell et al., 2000). In other cases, the authors suggest UML extensions that support additional concepts. The proposed modifications include (Wooldridge and Ciancarini, 2001):

- support for expressing concurrent threads of interaction, thus enabling UML to model such well-known agent protocols as the Contract Net;
- a notion of "role" that extends that provided in UML, and in particular, allows the modeling of an agent playing many roles.

It is important to point out that, like UML, AUML is rather a modeling language than a methodology. This means the proposed models can be used in different ways, according to the adopted methodology.

MessageUML

MESSAGE (Methodology for Engineering Systems of Software Agents), also known as MessageUML, is an agent-oriented methodology that builds upon current software engineering best practices covering analysis and design of MAS (Caire et al., 2001). MESSAGE uses a notation that is based on UML whenever appropriate. More specifically, MESSAGE's modeling language is related to UML as follows: a) it shares a common metamodeling language (meta-metamodel) with UML and MOF; and b) it extends the UML

metamodel with 'knowledge level' agent-oriented concepts. The main UML behavioral concepts that are used to define the 'physics' of the MESSAGE worldview are: action, event and state.

The main contributions of MESSAGE are its proposed agent knowledge level concepts and diagrams for viewing these concepts in the analysis model. Most of the MESSAGE knowledge level entity concepts fall into the main categories: ConcreteEntity, Activity, and MentalStateEntity.

A ConcreteEntity can have the following types:

- Agent: is an atomic autonomous entity that is capable of performing some (potentially) useful function. SoftwareAgent and HumanAgent are specializations of Agent.
- Organization: is a group of agents working together to a common purpose. It has structure expressed trough power relationships (e.g. superior-subordinate) between constituents, and behavior/coordination mechanisms expressed through interactions between constituents.
- Role: the distinction between role and agent is analogous to that between interface and object in UML.
- Class: describes the external characteristics of an agent in a particular context.
- Resource: is used to represent non-autonomous entities such as databases or external programs used by agents. Standard object-oriented concepts are adequate for modeling resources.

An Activity can have the following types:

- Task: is a knowledge-level unit of activity with a single prime performer.
- Interaction and Interaction protocol: the MESSAGE concept of Interaction borrows heavily form Gaia (refer to section 2.4.2). An Interaction by definition has more than one participant, and a purpose

which the participants collectively must aim to achieve. An InteractionProtocol defines a pattern of Message exchange associated with an Interaction.

MESSAGE assumes an architecture that separates an inference mechanism from a knowledge base and a working memory. The knowledge base contains fixed or slowly changing domain or problem-solving knowledge in a declarative form. The working memory contains more transient sense or derived information. This working memory is viewed as an abstract database holding instances of MentalStateEntities, and its contents define the Agent's mental state.

Two other simple but important concepts used in MESSAGE are: a) InformationEntity: is an object encapsulating a chunk of information; and b) Message: the agent-oriented concept of Message differs from the object-oriented one in a number of respects. In UML, a message is a causal link in a chain of behavior, indicating that action performed by one object triggers an action by another object. In MESSAGE, a message is an object communicated between Agents. The attributes of a message specify the sender, receiver, a speech act (categorizing the message in terms of the intent of the sender) and the content (an InformationEntity).

MESSAGE defines a number of views that focus on overlapping sub-sets of entity and relationship concepts: organizational view, goal/task view, agent/role view, interaction view, and domain view. The existence of different system views is aimed at providing flexibility to the analyst, i.e. he/she can choose an appropriate strategy, based on the combination of two or more of these different views. A possible modeling approach starts with a top level of decomposition, referred to as level 0, which is subsequently refined to provide a complete understanding of the system.

AORML

The Agent-Object-Relationship (AOR) modeling approach, applying the Agent-Object-Relationship Modeling Language (AORML) (Wagner, 2003), is based on an ontological distinction between active and passive entities,

that is, between agents and objects. The agent metaphor subsumes both artificial and natural agents. Thus, the users of the information system are included and also considered as agents in AOR modeling.

AOR distinguishes between agents and objects according to these two main points: 1) while the state of an object in OO programming has no generic structure, the state of an agent has a 'mentalistic' structure: it consists of mental components such as beliefs and commitments. 2) while messages in object-oriented programming are coded in an application-specific ad-hoc manner, a message in Agent-Oriented Programming is coded as a 'speech act' according to a standard agent communication language that is application-independent (Labrou et al., 1999).

In AORML, an entity is either an agent, an event, an action, a claim, a commitment, or an ordinary object. Agents and objects form, respectively, the active and passive entities, while actions and events are the dynamic entities of the system model. Commitments and claims establish a special type of relationship between agents. These concepts are fundamental components of social interaction processes and can explicitly help to achieve coherent behavior when these processes are semi or fully automated.

Only agents can communicate, perceive, act, make commitments and satisfy claims. Ordinary objects are passive entities with no such capabilities. Besides human and artificial agents, AOR also models institutional agents. Institutional agents are usually composed of a number of human, artificial, or other institutional agents that act on its behalf. Organizations, such as companies, government institutions and universities are modeled as institutional agents, allowing to model the rights and duties of their internal agents.

There are two basic types of AOR models: external and internal models. An external AOR model adopts the perspective of an external observer who is looking at the (prototypical) agents and their interactions in the problem domain under consideration. In an internal AOR model, AORML adopts the internal (first-person) view of a particular agent to be modeled. External models typically have a focus, that is an agent, or a group of agents, for which

we would like to develop a state and behavior model. Figure 2.9 shows the elements of an external AOR model, in which the language notation can be seen.

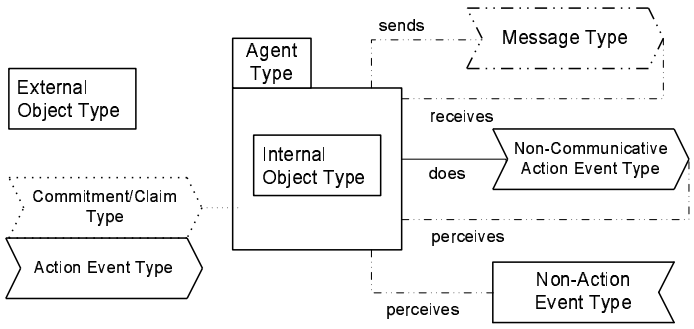


Figure 2.9: The core elements of AOR external models

Object types belong to one or several agents (or agent types). They define containers for beliefs. If an object type belongs exclusively to one agent or agent type, the corresponding rectangle is drawn inside this agent (type) rectangle. If an object type represents beliefs that are shared among two or more agents (or agent types), the object type rectangle is connected with the respective agent (type) rectangles by means of an UML aggregation connector. As can be seen in Fig. 2.9, there is a distinction between a communicative action event (or a message) and a non-communicative action event. Also, AOR distinguishes between action events and non-action events. The figure shows in addition that a commitment/claim is usually followed by the action event that fulfills that commitment (or satisfies that claim).

An external model may comprise one or more of the following diagrams:

- Agent Diagrams (ADs), depicting the agent types of the domain, certain relevant object types, and the relationship among them. An AD is similar to a UML class diagram, but it also contains the domain's artificial, human and institutional agents.

- Interaction Frame Diagrams (IFDs), depicting the action event types and commitment/claim types that determine the possible interactions between two agent types (or instances).

- Interaction Sequence Diagrams (ISDs), showing prototypical instances of interaction processes.

- Interaction Pattern Diagrams (IPDs), focusing on general interaction patterns expressed by means of a set of reaction rules defining an interaction process type. Reaction rules are the chosen component by AOR to show the agent's reactive behavior and it can be represented both graphically and textually.

As AUML, AORML is rather a modeling language than a methodology, not prescribing a specific modeling process. Instead, this should be defined case by case by the system analyst and designer.

MAS-CommonKADS

The MAS-CommonKADS methodology extends the models defined in CommonKADS (Schreiber et. al, 1994), adding techniques from object-oriented methodologies (OOSE, OMT) and from protocol engineering for describing the agent protocols (SDL, MSC96) (Iglesias et al., 1998).

The methodology starts with a conceptualization activity that regards an informal phase for collecting the user requirements and obtaining a first description of the system from the user's point of view. For this purpose, the use cases technique from OOSE is used, and the interactions of these use cases are formalized with MSC (Message Sequence Charts). The methodology defines the models described below for the analysis and the design of the system:

- Agent Model: describes the main characteristics of the agent, including reasoning capabilities, skills (sensors/effectors), services, goals, etc.

- Task Model: identifies the tasks (goals) carried out by agents, and task decomposition, using textual templates and diagrams.

- Expertise Model: describes the knowledge needed by the agents to carry out the tasks.

- Coordination Model: focuses on the conversations between agents, that is, their interactions, protocols and required capabilities.
- Organization Model: models the organization in which the multi-agent system is going to be introduced and the organization of the agent society.
- Communication Model: details the human-software agent interactions, and the human factor for developing these user interfaces.
- Design Model: collects the previous models and is subdivided into three sub models: application design: composition and decomposition of the agents of the analysis according to pragmatic criteria and selection of the most suitable agent architecture for each agent; architecture design: designing of the relevant aspects of the agent network; and platform design: selection of the agent development platform for each agent architecture.

This methodology has been successfully applied in several research projects in different fields, as intelligent network management, and the development of hybrid systems with multi-agent systems.

Comparing the Presented Methodologies

The methodologies described have different proposals for modeling an agent and a multi-agent system. However, they do have things in common. All of them, in a way or another, attempt to model an agent as an autonomous entity, and also address the interaction between this agent and the others in the agent society. Table 2.2 provides a comparative scheme among all described approaches, presenting the concepts each of them model in each development activity.

AUML, AORML and MessageUML are more easily adaptable in an industrial setting, since they extends UML, an already accepted standard in industrial environments. In a sense, these approaches extend object-oriented modeling techniques to model agents. This can be justified for a number

of reasons, for example, the fact that the object-oriented programming languages have been considered a natural framework for agent's implementation (Iglesias et al., 1999). On the other hand, as agents and objects are not the same thing, this can be a risky approach (Iglesias et al., 1999) (Wooldridge and Ciancarini, 2001). From these three approaches, only MessageUML provides methodological guidelines, the others limiting themselves on providing a modeling notation. AUML has been the choice of FIPA[3], the strongest standard body in the agent's community. Most efforts regarding AUML have been concentrated on expressing different kinds of coordination protocols. On the other hand, this methodology lacks support for information modeling, and more sophisticated behavior and interaction modeling mechanisms. Such tools are offered in AORML, which also offers the possibility of combining objects and agents. This seems to be very interesting, since not all entities in a system are active, thus being more adequately modeled as agents. Other than AORML and Prometheus, which mention the combination of agents and objects, Tropos and MessageUML allows the differentiation of passive and active entities, introducing the concept of resource to represent the former, while agent models the latter.

Gaia is also inspired in object-oriented analysis and design (in FUSION, to be more specific) (Wooldridge and Ciancarini, 2001). No example of successful experience using this methodology has been advertised until this moment. In addition to that, although Gaia claims to offer a design methodology, this is still a high-level design. Several steps are necessary to get the design to the point of actually being implemented. For this detailed design, another notation (often UML) is usually needed. Extending Gaia, ROADMAP presents some special features to model open systems, allowing the structure of the system to change at runtime. This capability is exclusive to ROADMAP, lacking in all other approaches here described.

Other methodologies have been developed with particular types of system in mind. Prometheus, for instance, is particularly suitable for BDI agents, i.e. agents are modeled based on their beliefs, desires and intentions. OperA, on the other hand, presents a suitable approach to accommodate agents

[3]http://www.fipa.org

designed by several parties. This tends to become more and more common with the developments in the SemanticWeb (Davies et al., 2003b), which envisions an Internet populated with agents which can be used individually or in combination to perform a variety of functions to the user.

While ROADMAP and Prometheus claim to support requirements specification, Tropos is the methodology that more consistently supports this activity. The first two approaches limit themselves at describing use case scenarios, while Tropos allows the analyst to systematically go from an organizational model to the elicitation of requirements for a system to support this organization.

Two of the presented approaches provide support to the definition of some kind of contract between the agents, namely OperA, with its normative structure, and AORML, with the use of commitments and claims to regulate agent's interaction. While a dependency in Tropos define some kind of commitment between two agents, this construct provides more limited support to contracts, as it only appoints the existence of commitments, rather than enforcing them at design time.

The MAS-Common KADS methodology, rather than being inspired by object-orientation, is closer to the knowledge engineering methodologies (Iglesias et al., 1999), and introduce a bigger number of details in the system's modeling. Although these methodologies might be adequate for modeling reasoning agents, it might add unnecessary complexity for the case of more general applications.

2.5 Building Blocks for Constructivist Knowledge Management

In this thesis, we claim that KM can be more effectively supported if viewed through a Constructivist perspective. Constructivism explains how individuals build knowledge in a natural and informal way. The Constructivist paradigm is often confused with anarchism and disorder (Mahoney, 2004). However, it is important to note that emphasizing individual autonomy and

Methodology	Development Activities			
	Early Requirements	**Late Requirements**	**Architectural Design**	**Detailed Design**
Gaia	-	Roles and interactions	Agents, services and acquaintances	-
ROADMAP	-	Functionalities, environment model (zones, percepts and actions); knowledge, roles and interactions	Agents, services and acquaintances	Interactions in more details
OperA	-	Organizational model (roles, coordination models, norms, ontologies, interaction scripts)	Social model (agents, social contracts), interaction Model (agents, interaction scenes)	-
Tropos	Organizational actors, goals, softgoals, tasks, resources, dependencies	Organizational actors (including systems), goals, softgoals, tasks, resources, dependencies	Sub-actors detailing system architecture	Capabilities and interactions modeled with AUML
Prometheus	-	Actions, percepts and functionalities	Agents and assignment of functionalities to agents, events and messages (interaction)	Capabilities, plans

Methodology	Development Activities			
	Early Requirements	**Late Requirements**	**Architectural Design**	**Detailed Design**
AUML	-	-	-	Activities and interaction
Message/UML	-	Domain concepts, organization, roles, agents, resources	Roles, agents, resources, tasks, interaction goals	Roles, agents, resources, tasks, interaction goals
AORML	-	Agents, objects and relations	Agents, objects and relations	Interactions and internal behavior
MAS-CommonKADS	-	Functionalities, agents, tasks, events, interaction, knowledge (domain, agent, and environment), organization model and society of agents	Network facilities (e.g.: yellow/white pages, agent name service etc.), knowledge facilities (e.g.: ontology servers), coordination facilities (e.g. coordination protocols), agent architecture	Selection of software platform

Table 2.2: Comparing methodologies

active participation in knowledge creation does not imply that no planning should be required, and that knowledge inadvertently emerges. In fact, this section is dedicated to understanding which would be the conditions for knowledge creation, according to the presented Constructivist theories.

The KM theories presented in section 2.2.4 have been selected due to the congruence between their principles and the Constructivist views (even if this has mostly not been consciously noted by these KM theories' proponents). Here, we would like to discuss the most significant contributions of these KM theories in light of the presented Constructivist theories (refer to section 2.3). This discussion will grant us with the means to draw the requirements for the development of what we will call from now on Constructivist KM.

A clear connection between the described KM theories and Constructivism is given by the fact that all these theories are highly based on the *active and autonomous* engagement of the knowledge holder on the process of acquiring and sharing knowledge. Each person has a particular way of expressing him/herself, sharing what they know, as well as seeking for new knowledge and abilities. If individuals are restrained, they may feel uneasy about sharing knowledge with others. While contributing to motivation, autonomy may also lead to unexpected situations of knowledge creation and innovation. Individuals should then be impelled to share instead of retain knowledge. Organizations should actively create opportunities for sharing and collaboration. But employees should also be allowed to find their own ways to exchange knowledge while engaged at work, so as to avoid sharing to become bureaucratic or as an obligation. In this sense, despite of the organizational power structure, knowledge exchange should be *non-hierarchical*, i.e. sharing processes should not be constraint by organizational roles or hierarchical positions. Instead, all organizational members should be recognized as possible knowledge contributors, each one being valued for their competence and expertise in their particular fields of action.

All theories presented here value the role of *social interaction* to the construction of knowledge, recalling the social cognitive theory of Vygotsky. Humans are social beings, and knowledge only emerges as a result of their interaction. In the Knowledge Spiral, social interaction is paramount for

knowledge sharing, especially for the processes of socialization and externalization, which respectively lead to the communication of tacit and explicit knowledge. Situated Learning makes direct reference to Vygotsky's theory, especially when exploring the benefits of apprenticeship, which leads a newcomer to learn by collaborating with an expert, being directly engaged in his practices. What makes CoPs so desirable for KM is exactly the types of dynamic environment it creates on the basis of its members interaction when involved in the community's practices. Through these interactions, meanings are negotiated and the community's practices are themselves constantly reshaped. Finally, DKM states that while locally managing their knowledge assets, knowledge holders should get involved in a rich exchange of knowledge, in a way that their local interpretations are dynamically negotiated.

Piaget's theory emphasizes the importance of a *perturbation* in the individual's cognitive system as the trigger for learning. This has also been noted by Nonaka and Takeuchi, who indicate that fluctuation and creative chaos are important characteristics of an environment that supports knowledge creation. In a sense, this is also implicitly included in Paulo Freire's Pedagogy of Question. This theory states that a question is the initiator of dynamic processes comprehending discussion and reflection in action, which are relevant for the generation of innovation.

The Constructionism proposed by Papert has a lot in common with the Knowledge Spiral, since both focus on the knowledge externalization and internalization cycles that occur when people interact, ultimately leading to knowledge creation and learning. In addition to that, Papert has emphasized the importance of an external and *socially meaningful artifact* for knowledge creation. This artifact is part of what Wenger calls shared repertoire, in the context of Communities of Practice, i.e. a resource that is created as a result of shared practice, or used in action by community members, and which is meaningful in this social environment. Knowledge artifacts are also important in the context of DKM, in which knowledge exchange is attributed to the constant sharing of locally managed knowledge artifacts.

Finally, Paulo Freire argues about the need to provide *contextualized environments* for the proper emergence of knowledge. In other words, he claims

that learning is much more productive in environments that are connected to the learner's reality. In this sense, Situated Learning and knowledge sharing in CoPs affirm the same, as they propose that people learn by being involved in real practices (i.e. learning by doing), instead of acquiring a body of abstract knowledge that can supposedly be transported and applied in posterior situations.

Summarizing the above discussion, Fig. 2.10 presents the building blocks for Constructivist KM, i.e. the conditions for it to emerge.

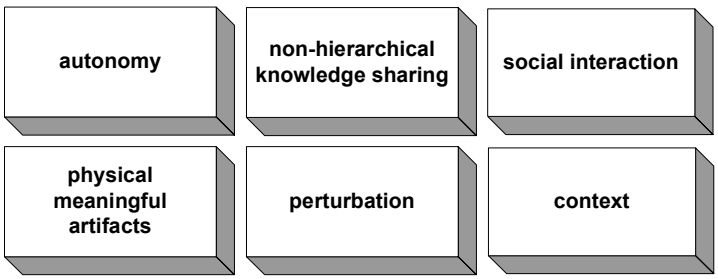


Figure 2.10: The building blocks for Constructivist Knowledge Management

In summary, Constructivist KM claims that more attention should be given to the knowledge holders. They should be the focus when proposing a specific process or system to support KM. This does not mean that organizational objectives will be forgotten. Rather, organizational global goals (or what Nonaka and Takeuchi call organizational intention) and individual goals of the organization's members should both be considered. Constructivist theories explain the processes that lead to the emergence of knowledge. Supporting these processes is the key to the generation of new knowledge and innovation.

Having established this work's philosophical basis, it is important to understand how to take Constructivist KM into practice. Constructivism has been mentioned in connection to computer science in different ways. For example, it has been used as a supporting theoretical framework for work on Computer Supported Collaborative Learning (CSCL). In an innovative approach to Artificial Intelligence, Drescher (1991) has built a computational learning and concept-building mechanism referred to as schema mechanism

inspired in Piaget's constructivist theory. Our work is more closely related to the former than to the latter, as constructivism here is also seen as a theoretical framework to guide the development of KM practices and information systems.

More specifically, we propose that, aiming at understanding these processes and verifying if the conditions for Constructivist KM hold, an analysis of the organizational setting must take place. In the analysis, several knowledge holders from different levels within the organization structure should be consulted, so that a global understanding can be built. The analysis should be able to negotiate the global strategies of the organization and the individual goals of the knowledge holders. As a result of the organization analysis, an internal process may be redesigned, or a new KM system may be developed, following the principles of Constructivist KM. In this way, the proposed solution is prone to find less resistance in the organizational setting, targeting some of the challenges described in section 2.2.3

2.6 Towards Agent-oriented Constructivist Knowledge Management

In this work, we claim agents are suitable abstractions for modeling KM domains and systems. Some reasons for this can be found in early section 2.4. To be more precise, in KM settings, agents may represent not only artificial beings, but also the human users and the organizations involved in a given scenario. Agents' inherent characteristics (such as autonomy, reactiveness, proactiveness and social ability) and cognitive concepts (e.g. intentions, beliefs, commitments/claims) enable the explicit capture of important aspects of the organizational setting. For instance, they allow reasoning about organizational members' beliefs and perceptions, their interactions, and the commitments they establish with each other on the organization's behalf. The understanding about these aspects help the analyst to create a clear picture of the organizational culture, besides understanding how knowledge flows within the organization. As a result, besides inserting new technology,

the business processes applied in the organization may be changed in order to enhance these knowledge flows. In addition to that, if a technological solution is needed, agents enable legacy systems to be considered in the analysis, allowing the new solution to be based on approaches of integration of old and new components. This may lead to more satisfaction to end users, who are already familiar with the interface and methods applied in the systems in use. This is compliant with the idea of Constructivist KM, which emphasizes the need to provide more attention to knowledge holders, while at the same time, trying to consider organizational general goals and strategies when proposing a KM solution.

Agents are specially powerful to assist in the capture and reasoning of the identified Constructivist KM principles. For example, as explained in section 2.4.1, autonomy and social ability are inherent characteristics of agents. By exploring them, we are able to identify how much *autonomy* is granted to the agents of a KM setting, and how well they collaborate and interact (refer to *social interaction* building block). Examining the structure of the agent society, understanding the relations between its populating agents, gives us the possibility to grasp if knowledge flows in a hierarchic way or not within this society (verifying the compliance to the *non-hierarchic knowledge sharing* building block). While pursuing their goals, agents apply information resources. These passive system entities are captured by constructs of most of the described engineering approaches, being modeled as objects, resources or permitted information items. The analysis of the information resources created, maintained and exchanged between agents enables the identification of the *physical meaningful artifacts* that mediates knowledge creation and sharing. Finally, by focusing on the characteristics of the environment where agents inhabit, we are able to analyze whether or not such environment provides *context* for knowledge sharing, and to which extent constructive *perturbations* are explored.

The lack of a systematic agent-oriented methodology has been mentioned as one of the biggest obstacles for the large-scale adoption of the agent paradigm Jennings et al. (1998) Parunak (2000). Although much work targeted this topic in the past few years, there is still room for debate in the

area. Nevertheless, it seems that this discussion has reached a point of maturity. If before the focus was the proposal of different modeling abstractions and methods, today the attention has shifted to comparing methodologies in order to understand for which application types each approach is more suitable (Juan et al., 2004) (Dam and Winikoff, 2003) (Sabas et al., 2002) or for combining different approaches (Henderson-Sellers, 2005) (Bernon et al., 2004) (Juan et al., 2003) in order to maximize their advantages and minimize their drawbacks. Perhaps, at some point, this debate will culminate with the proposal of a standard approach, following the paths of the object-oriented community that has proposed UML as a standard for object-oriented analysis and design.

This is also the approach taken in this thesis. Our objective is to combine existing work on agent-oriented software engineering to tailor a suitable methodology for Constructivist KM settings. In our view, important characteristics of an adequate methodology are the following:

- supporting crucial engineering activities, allowing the developer to first understand the analyzed environment and its inherent problems, and then consistently lead the developer to the design of the solution;
- offering a suitable set of concepts and constructs for the targeted system development activity;
- providing a visual language besides textual descriptions, thus facilitating the communication between stakeholders and analysts, and among analysts and system designers;
- being relatively accessible and not requiring too much overhead in the sense of extra work from the part of the analyst in understanding and using the given language and method.

The first item above regards the particular development activities and life cycle supported by the methodology. To guarantee that a clear understanding is achieved before a solution is proposed, our methodology should start with a detailed analysis of the domain, leading to the elicitation of

system and/or process requirements. And then, in case an information system is proposed, the methodology should cover architectural and detailed design activities, enabling the designer to achieve the final stages of system implementation.

With respect to the second issue above, it is important to find, among the existing approaches, the ones that support all entities of the domain, namely organizations, organizational members, communities, systems, and other resources. Besides, the applied notation should be able to systematically represent relations between these entities, model their interactions, and design their internal behavior. In addition to that, we should focus on the right choice of agent mentalistic characteristics to be applied in the different activities of the development cycle. Concepts such as agent's *beliefs*, *goals*, *commitments* and *plans* are vastly discussed in literature and different models have been proposed (Rao and Georgeff, 1991) (Wooldridge and Ciancarini, 2001). However, it is hard to know how to effectively consider these concepts while analyzing a domain and designing a system. Particularly, it is difficult to materialize these constructs as concrete elements of an information system.

The discussion about adequate concepts, models, and life cycles is resumed in the next chapter, where we propose and describe our methodology. We specifically explore the combination of two agent-oriented approaches, complying with the third and forth items above. The resulting notation provides the user with a comprehensive set of diagrams and analysis methods that are visually rich, thus facilitating reasoning and communication about the created models. Furthermore, the two combined approaches are well-known in the agent-oriented community. This makes the effort of analysts and designers adopting our methodology much diminished, in comparison with cases in which a total new set of concepts, constructs and methods are adopted and need to be learned from scratch.

Chapter 3

The ARknowD Methodology

"Our brains are essentially
model-making machines."
Vilayanur S. Ramachandran

Having seen in the previous chapter, what are the characteristics of a KM environment and what are their most pressing needs, the next step is proposing an approach that taking such characteristics into account, attend the main highlighted needs. For this purpose, we introduce a methodology we name Agent-oriented Recipe for Knowledge Management System Development (ARKnowD). ARKnowD is aimed at analyzing the current state of an organization, trying to identify the previously defined Constructivist KM building blocks. As we have seen, the degree to which the organizational setting meets these building blocks directly conditions KM support. We believe that Constructivist KM especially leads to a higher degree of acceptance level of organizational members towards information systems and practices adopted by the organization.

In this chapter, we start by introducing the main assumptions underlying ARKnowD (chapter 3.1), one of which is the applicability of existing agent-oriented methodologies that in combination, may meet the demands of our approach. Then, we describe in depth the scenarios of applicability of our methodology (3.2) and its underlying activities and life cycle (3.3). Next, we discuss existing work using agent's cognitive concepts (3.4) and based on

such concepts, develop an ontology comprehending all ARKnowD's modeling constructs (3.5). This ontology is then used as a reference model to guide the evaluation of the two notations applied in ARKnowD, making the necessary adjustment to facilitate their merge into one language (3.6). Moreover, in order to combine these two existing agent-oriented approaches, we provide an MDA-inspired transformation method, described in detail in section 3.7. Following, section 3.8 focuses on a working example that serves as a context for some important methodological guidelines that should aid the analyst and designer to undertake the task of building an organizational model using ARKnowD. Next, we provide a discussion on automated support (3.9) which is succeeded by the description of a few approaches having affinity with ours (3.10). Section 3.11 finally concludes this chapter.

3.1 Introduction

As discussed in the previous chapter, KM has lately become the focus of much attention and great investments within organization settings. This stems from a general realization that the biggest assets owned by businesses and corporations lie within the minds of the people who constitute them, or work on their behalf. As a consequence, knowledge is usually embedded in the organization's routines and products. The understanding of how knowledge flows within an organization is paramount for supporting KM. This understanding enables knowledge to be elicited, besides providing insights on how to support KM in a specific organizational scenario.

As argued in section 2.6, agents are suitable entities to model human and artificial organizations due to their autonomous, reactive and proactive nature, besides other cognitive characteristics. This can support domain analysts and system designers in understanding the current organizational setting before proposing the development or adoption of particular solutions. However, having an appropriate abstraction is not enough for guaranteeing the development of adequate solutions for the organization. For that, a consistent engineering methodology is needed to transform these abstract

notions in useful tools to enable problem solving in real scenarios.

In this thesis, we propose a methodology named Agent-oriented Recipe for Knowledge Management System Development (ARKnowD), preliminarily introduced in (Guizzardi et al., 2005). ARKnowD is an integrated agent-oriented methodology to develop KM solutions, which represents as agents all humans, organizations and information systems of the domain. This enables the analyst to understand their relations and interactions, guiding him/her on finding appropriate solutions to target the idiosyncrasies of that particular environment. Note that our conceptualization of 'system' is general, including but not being restricted to that of information system. In this work, system is defined as a general set of interacting entities (Bunge, 1979), thus comprehending artificial and non-artificial entities (such as humans, organizations and organizational units). This opens the possibility to consider several outcomes resulting from the application of our methodology, such as: changing organizational structures, modifying processes, and adopting technological or non-technological tools.

Benefits as a result of the application of ARKnowD may be attributed to our choice of using the proper agent cognitive characteristics in the different development activities. Concepts such as agent's beliefs, goals, and plans are vastly discussed in literature and different models have been proposed. However, there is a gap between knowing their definition and actually applying them effectively in practice. In this respect, our work attempts to provide an answer to the following questions: Should these concepts be considered all at once in system development? If not, when are *goals* suitable, and when should the developer start considering agent's *beliefs*, for example? And, perhaps, the most frequent question of all: How can these concepts be materialized in practical elements of an information system? Although there is no final answer for such questions, we aim at contributing to clarify these important issues.

Another strong characteristic of ARKnowD is the adoption of visual modeling for supporting requirements analysis and design. A model here can be defined as an abstraction of the reality. It usually focuses on a particular perspective and thus, on specific concepts that are important for a given model-

ing activity. Visual modeling supports reasoning both about the domain and the proposed solution. More specifically, visual models provide analysts and designers with means for visualizing, specifying and communicating about elements of the domain and the solution. Besides, visual modeling allows groups of developers to work in physically disperse environments, aiding the integration of the different components of an information system, developed with basis on models made with the same notation. Finally, communication with the stakeholders is also facilitated by the use of visual models. In this sense, we emphasize the appropriateness of agent-oriented concepts, such as goals, plans, beliefs and commitments. These concepts are much closer to the reality of common stakeholders than technology-oriented terminology, such as tables, SQL query, middleware and threads.

Given the current stage of research on the agent-oriented paradigm, and the vast availability of methods and languages for agent-oriented analysis and design, the methodology presented here is built over existing work. One of the principles of our methodology is granting analysts and designers with freedom to select the appropriate tools from a vast 'library' of methods and languages, depending on the specific case at hand. It is our belief that not one method possesses all the right properties. Instead, these properties can often be attained by combining different approaches. This view is compliant with the method engineering approach adopted in the OPEN metamodel (Henderson-Sellers, 2005), which prescribes the reuse of fragments of different agent-oriented methods according to a given situation. Figure 3.1 illustrates different possibilities of combining existing agent-oriented languages and methodologies, already described in section 2.4.2.

Fig. 3.1 depicts several options one could select when developing a system. For instance, analysts that are familiar with the Gaia methodology (Wooldridge et al., 2000) could start with the definition of roles and interactions and, then, refine these models respectively into OperA's roles and scenes (Dignum, 2004a) (path 1). This would result in a more detailed and formalized analysis model. Another possibility is given by the combination of Tropos (Bresciani et al., 2004) and OperA (path 2). As viewed in table 2.2, Tropos is the approach that gives more attention to the requirements

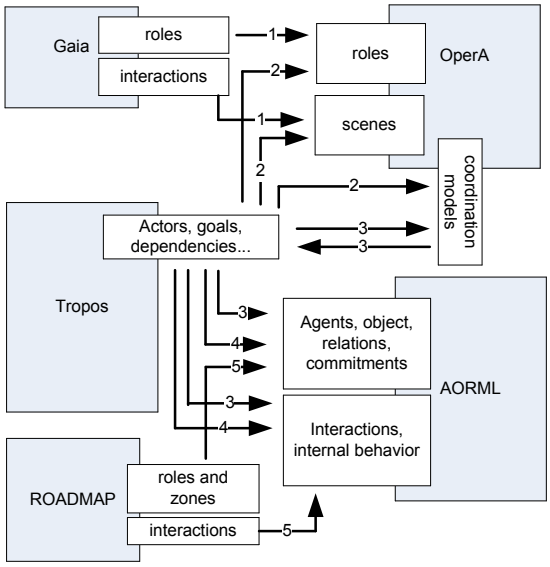


Figure 3.1: Combining different agent-oriented approaches

analysis activity, which can be highly beneficial for KM settings. Combining Tropos and OperA allows the analyst to take advantage of Tropos's requirements analysis propensity at the same time as generating a formalized analysis model. However, if a formalized model is not necessary, Tropos can be combined with AORML (Wagner, 2003) (paths 3 and 4), generating a methodology that covers all activities of system development, coming from early requirements analysis to detailed system design (this last activity is not covered by OperA). Following path 3, the analyst still applies one of OperA's resources, namely the coordination models, to guide them on the creation of a suitable system architecture (Dignum, 2004a), before using AORML for detailed design. Path 4 is indicated for cases where the designer uses his previous experience for generating the solution's architecture, so AORML can be applied directly after Tropos. Finally, in path 5, ROADMAP (Juan et al., 2002) and AORML are combined. ROADMAP is specifically tailored to enable the development of open systems. This methodology usually applies AUML (Odell et al., 2000) for detailed design. Here, however, we suggest AORML as an alternative design language.

In this work, we explore the possibility given by path 4, i.e. the combina-

tion of Tropos and AORML. Both approaches use the notion of agent and related cognitive concepts in all software development activities, from early requirements analysis down to the actual implementation. However, as an extension of UML, AORML is rather a modeling language than a methodology, though some methodological directions on how to use AORML for software development have already been identified in (Wagner, 2003).

Another difference is given by the different strength of each of the approaches for the different system development activities. Tropos gives a crucial role to the early requirements analysis activity that precedes the prescriptive requirements specification of the system-to-be. Although AORML has been proposed for domain modeling (Guizzardi et al., 2004a), it does not provide specific diagrams for requirement's specification. Even if traditional UML use cases diagram may be applied, the Tropos's notation is much more rich and appropriate for agent-oriented modeling. On the other hand, contrarily to AORML, Tropos does not cover detailed design, adopting AUML for supporting this activity.

The adoption of AUML has its drawbacks. For instance, AUML does not provide a rich model of the domain and system entities such as AORML (information modeling). Using the AOR agent diagram, the designer is able to represent all agents and objects considered in the domain, along with their properties. In AUML, information modeling does not receive much attention, and traditional class diagrams are often used, although they are not made for agent-oriented modeling. The result is that only agents are allowed (no objects are considered), or else the same construct is used to represent both agents and objects. The latter leads to construct overload, which may seriously undermine the understanding of the resulting models (Guizzardi, 2005). Another advantage of using AORML is the availability of three different diagrams of modeling interactions, compared to one offered by AUML. This includes the AOR Interaction Pattern Diagram, which models the internal reactive behavior of an agent while interacting with others.

The differences between Tropos and AORML suggest that these two approaches can be rather complementary than competing. The concepts adopted in Tropos can be consistently mapped to AORML constructs, al-

lowing them to be carried out farther from the requirement analysis to the design activity. Section 3.7 presents more details on the transformation between these two notations.

We believe the combination Tropos/AORML is profitable in both directions. Specifically in respect to organizational or KM Systems modeling, Tropos may benefit from the following strengths of AORML: 1) the fact that 'mentalistic' (or cognitive) concepts of agents, such as beliefs and commitments, are explicitly considered in system design supports the analyst/designer to reason about and to model the behavior of agents, both internally and in interaction with other agents of the system; 2) although norms and contracts are not directly supported by AORML, it provides deontic modeling constructs such as commitments and claims, which form the basis for the establishment of such norms and contracts; 3) it captures the behavior of agents with the help of rules. Besides these strengths, since AORML is an extension of UML, preserving its principles and concepts, it is an accessible language, and it allows the use of UML constructs whenever an extension is not provided, thus providing a comprehensive set of tools for the analyst/designer. On the other hand, the explicit use of Tropos's goals and plans provide a rich conceptual framework for modeling the intentional dimension of the organization. This includes a preliminary view of how user's interact, without however adding unnecessary protocol details in the early stages of requirements analysis. Such concepts of goals and plans are missing in AORML.

3.2 Scenarios of Applicability

We envision three possible application scenarios for ARKnowD, illustrated in Figure 3.2.

a) Proposing changes in the organizational structure to accommodate/enhance KM practices.

b) Modifying business processes to accommodate/enhance KM practices.

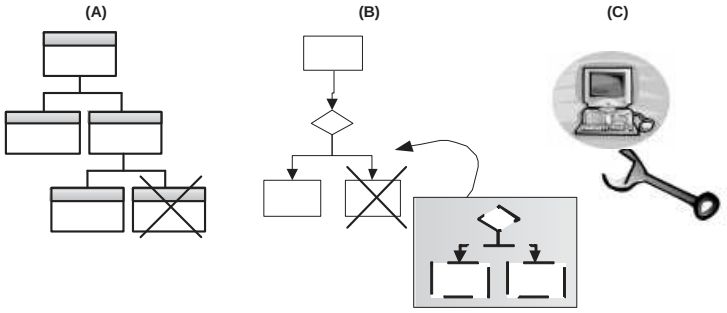


Figure 3.2: Three application scenarios for ARKnowD

c) Adopting an enabling tool, technological or not, to support KM. In case an information system is needed, there are three possibilities:

- Developing a KM System based on legacy systems currently in use.
- Developing a KM System from scratch.
- Adapting an existing KM System (out of the shelve) in a current business process, making the necessary processual changes accordingly.

Technology has been cited as only one part of the solution to enable KM (Orlikowski, 1992b). At times, enhancing the knowledge flow within the organization does not require the adoption of any new tool, but rather modifying the structure and the business processes underlying the organization.

Organizational reengineering has often been mentioned as a solution to create more conducive environments for knowledge sharing (Orlikowski et al., 1995) (Wiig, 1994). This can be achieved by creating or extinguishing organizational units or departments. As an interesting example, we cite the case of the Dutch insurance company Achmea reported in (Dignum and van Eeden, 2003), which created a new division (the KM division) to propose projects and take care of all matters related to KM. Other cases include periodic rotation of personnel, which may enhance the flow of knowledge from one department to the other. New knowledge sharing opportunities and synergy accrue from such personnel rotation, as recognized by Nonaka

and Takeuchi (1995) and Perini et al. (2004).

Several are the cases reporting KM enhancements that emerge as a result of changes in process. Nonaka and Takeuchi (1995) report a case of this sort, involving a team of technicians from Matsushita that were involved in the construction of the first fully automatic electronic bread-making machine for home use, released in 1987. After a few frustrating experiences, especially to automate the process of kneading the dough, Matsushita achieved encouraging results after an internship that a senior developer made with a baker. In observing and participating in the activities of making bread, he realized the right way a dough should be worked on, posteriorly embedding a mechanism in the machine that imitated the movements of the baker. In this case, no new department was created and no new tool was adopted. Instead, the routine of an employee (the software developer) was radically changed, enabling him to capture tacit knowledge embedded in the practices of a specialist, much as advocated by Lave et al. (1991) (refer to section 2.2.4 for a full discussion on this).

An effective knowledge flow requires the right set of tools or instruments. These tools can be based on information technology or not. In some situations, a simple flip-chart may be required to register new ideas in a brainstorming session, allowing the exploration of analogies and metaphors, advocated as drivers for tacit knowledge sharing (Nonaka and Takeuchi, 1995). For other purposes, information systems as the ones described in section 2.2.2 may be needed to enable and enhance the capture, structuring and dissemination of knowledge throughout the organization. Finally, a combination of physical tools and information systems' capabilities may result in the creation of novel supporting instruments, following the trend of ubiquitous computing (Weiser, 1994).

ARKnowD supports three distinct situations regarding the adoption of information systems. The system may be developed from scratch. For that, ARKnowD analyzes how the new system fits in relation to the members of the organization, and how it changes the current processes. Following, the inner-structure of the system is detailed and developed. However, organizations normally have some legacy system in place that although obsolete,

offer a few functionalities that are essential to the organization. Agents have been cited as especially suited for enabling development of new systems by integrating legacy systems (Jennings et al., 1998) (Wooldridge, 2002). In this case, ARKnowD includes legacy systems as agents in the analysis of the organizational settings. The interaction of these systems with the human agents of the organization is detailed and finally, new functionalities may be proposed. This can be done either by wrapper-agents that add new levels of functionality over the legacy systems (Wooldridge, 2002), or by new agents that simply interact with them as well as with human agents. Finally, there is a third possibility of information system adoption. This refers to the case when an organization wants to adopt a specific system that is already a finished product. This is more common than one might imagine, as a result of hype or market pressure regarding a specific kind of technology. Here, ARKnowD supports the insertion of the new information system within the organizational setting, by fitting it into organizational processes, making eventual necessary adjustments to better integrate the new solutions within organizational practices.

3.3 Activities and Lifecycle

The proposed methodology comprehends the following activities:

1. *Requirements elicitation.* Requirements elicitation is a basic activity of all software engineering processes. Before starting to build a new information system, it is necessary to grasp what stakeholders (i.e. future users of the systems and other influenced parties) really need. This is exactly the main aim of Requirements elicitation (Nuseibeh and Easterbrook, 2000) (Goguen and Linde, 1993). In ARknowD, however, this activity takes a more general notion, since the organization may not need a new information system, but solely a structural or process change. Thus, requirements here refer to any need for change in terms of organizational structure, process and tools, reflecting the scenarios described in section 3.2.

In any case, one thing is certain: the way people work within an organiza-

tion is bound to change. These changes may bring about positive outcomes in respect of knowledge sharing. Changes generate perturbations in the organizational environment, which as claimed by Constructivist KM, naturally lead to learning and performance enhancement (for an in depth discussion about this topic, refer to the previous chapter, sections 2.2.4, 2.3.1 and 2.5). However, there is no point in making unecessary changes, or even hampering current best practices within the organization. Thus the importance of gathering the right set of requirements before jumping into the development of a new solution.

As the term *elicitation* suggests, requirements are not simply out there, waiting to be collected (Nuseibeh and Easterbrook, 2000). The activity of requirements elicitation is, thus, ill-structured and complex. Several techniques have been propose to support this activity, as described in (Goguen and Linde, 1993) and (Nuseibeh and Easterbrook, 2000). Traditional techniques rely on individually answered questionnaires or interviews, while others propose special activities to be performed with groups of stakeholders. Ethnographic techniques have recently gained attention in recognition for the complexity of requirements elicitation. Ethnomethodology usually requires that the analyst is immersed in the organizational setting, performing active observation and trying to understand this setting from the perspective of a real member of the organization. ARKnowD does not prescribe any specific technique for requirement elicitation. However, our work recognizes that observation of people in action, as proposed by ethnographic techniques, should be combined with interviews and questionnaires, leading to more consistent requirements.

2. *Requirements analysis.* Requirements analysis refers to the activity of modeling and reasoning about organizational requirements. Our methodology models requirements as goals. This view has been largely acknowledged by the Requirements Engineering community. Note, for instance, that Zave cited by (Nuseibeh and Easterbrook, 2000) defines Requirements Engineering as *"the branch of software engineering concerned with real-world goals for functions of, and constraints on software systems"*. This definition emphasizes the importance of viewing the stakeholders' real goals as motivators

for choosing a particular solution over another. For a discussion on RE methodologies that apply goal analysis, refer to (Kavakli and Loucopoulos, 2005).

The adoption of goals is also compliant with KM theories. According to the Knowledge Management Spiral (Nonaka and Takeuchi, 1995) (see section 2.2.4), for example, one of the main drivers of knowledge creation is the organization's intention, defined as "an organization's aspiration to its goals". Nevertheless, these authors mainly focus on the organization top management's intention for facilitating KM initiatives. In contrast, we consider the goals of all stakeholders, trying to understand the relations and possible discrepancies between their goals. This view is aimed at providing autonomy in knowledge sharing, as prescribed by Constructivist KM, and also emphasized by Dignum (2004a).

ARKnowD particularly supports the analysis regarding the extent of the analyzed setting's compliance to the Constructivist KM building blocks earlier defined in section 2.5. In other words, the methodology's supported concepts and techniques allow the analyst to understand:

- how much *autonomy* is given to each organizational member to share knowledge the way he/she sees fit;
- if the creation and sharing of knowledge happens in a bureaucratic way, obeying hierarchical structures within the organization of if it is rather *non-hierarchical* and natural, motivating each one to contribute with his/her share of knowledge despite organizational position or experience;
- how well organizational processes favor *social interaction*, considered here as an essential ingredient for the disambiguation of tacit knowledge, and thus for the generation of innovation;
- what kind of *meaningful artifacts* are exchanged among organizational members, cross-cutting divisions and communities and in this way, carrying knowledge throughout the organization;

- how constructive *perturbations* are generated and coped with within the organization, triggering the dynamics that motivate employees to constantly self-improve;
- what kind of contexts emerge or are actively planned by the organization for knowledge creation, integration and sharing.

We claim that the presence of the highlighted characteristics within the organization's environment lead to more effective support to KM. Thus, a deeper understanding of how much the analyzed setting complies with these principles gives the analyst the means to assess how well the organization currently supports KM. Consequently, such principles may be used as guidances, serving as a kind of checklist for the domain analyst. Understanding the presence or absense of such principles allows limitations in the organization's environment to be corrected and appropriate KM solutions to be proposed.

Borrowing Tropos's approach, requirements analysis is divided in two subactivities, namely early requirements and late requirements analysis (Bresciani et al., 2004). In early requirements analysis, modeling focuses on understanding the organization's and stakeholders' goals as they carry out their work and responsibilities. This activity culminates with an understanding of the rationale behind stakeholders' needs of a specific solution (in terms of tools, changes in the organizational structure, or process changes). Finally, in late requirements, we focus on the requirements for the solution, being able to trace them back to the fulfillment of the social and individual goals previously analyzed.

3. *Design.* The design activity is responsible for providing the solution in as much detail as to enable it to be developed in practice. At design time, we should be able to identify all agents that take part in the structure of the solution, as well as their relations. Moreover, processes are fully modeled, making clear the interactions among agents, besides their external and internal behaviors that should also be explicitated.

In the case of the development of an information system, the design can be viewed as two distinct sub-activities: *architectural design* and *detailed*

design. As the name suggests, architectural design is concerned with the architecture of the system under development. At this point, all agents of the system should be identified, along with their individual goals. In addition, the resources and plans used by the agents to achieve their goals are modeled. Note that the transformation between Tropos and AORML happens exactly at this point. Finally, in the detailed design, the information structure of the system is detailed, along with processes and agent's behavior. The final result of detailed design should be a system model that can be implemented using a programming language and/or framework. ARKnowD does not commit to a specific programming infrastructure. Chapter 6 exemplifies the detailed design of a system implemented using the JADE framework (Java Agent DEvelopment Framework) [1].

Figure 3.3 illustrates ARKnowD's life cycle, summarizing our discussion on the supported activities.

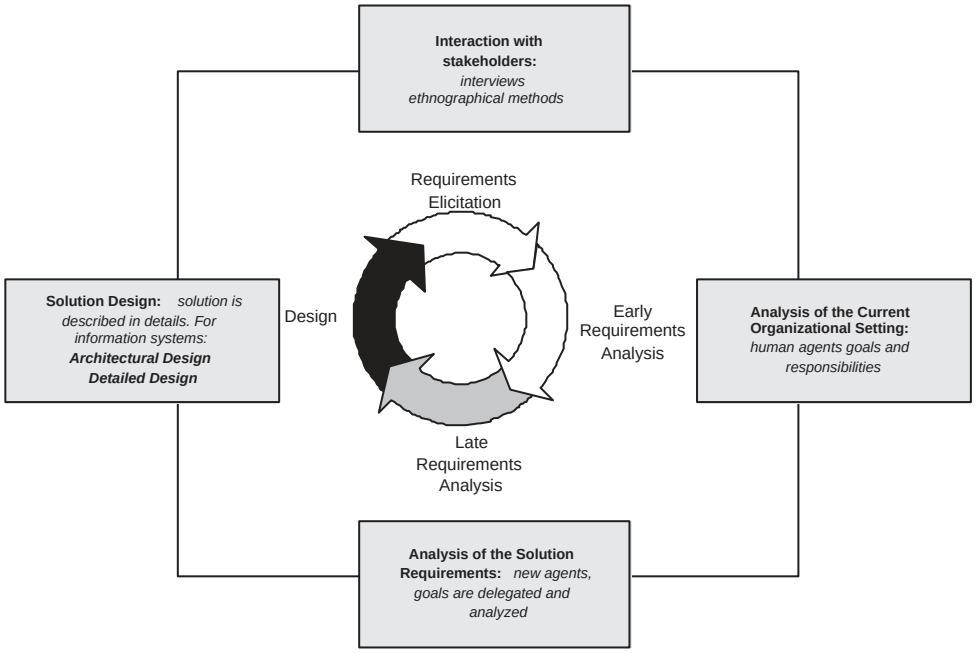


Figure 3.3: ARKnowD's lifecycle

Note that this chain of activities may be performed several times, in an iterative process. The development of a solution commonly requires several

[1] http://jade.tilab.com/

cycles, each one performing some or all activities to a certain extent. The first cycles are characterized by the focus on the clarification, negotiation and agreement of requirements, thus requirements elicitation and analysis are iterated several times. Then, the focus slowly shifts to the development of the solution, although each design cycle may still require the elicitation and further analysis of new requirements. To clarify this point, we borrow from the Rational Unifying Process (RUP) the concept of phases (Kruchten, 2000). RUP divides the development process in four phases, defined as follows:

- *Inception*: targets the specification of the vision of the solution and its business case, as well as the definition of the scope of the project.
- *Elaboration*: focuses on planning the necessary activities and required resources, besides specifying and designing the architecture.
- *Construction*: aims at building the solution and evolving the vision, the architecture, and the plans made in the previous phase until the solution is ready for delivery to the targeted community.
- *Transition*: concerns the adjustment of the solution to the stakeholders. In case the solution is non-technological, it usually involves training and monitoring the stakeholders within their new division, or performing new working processes If an information system is part of the solution, besides training and monitoring, it also involves deploying, supporting and maintaining the system.

Organizational top and/or middle managers are usually responsible for the project's vision (Nonaka and Takeuchi, 1995), elaborated during inception. However, Constructivist KM emphasizes the importance of involving all organizational members in this debate as much as possible. Inception is also a time for advertising the project's vision, gaining support from the stakeholders, especially those individuals that are going to be involved in the activities of requirements elicitation. In fact, this phase may even indicate who are the best candidates to be included in the elaboration plans for future observation or interviews.

During transition, the applied solution should be assessed, and minor problems should be corrected. Assessment typically takes a long period, as the results of early and late employments of new processes and tools may be very different (Orlikowski, 1992a). Meanwhile, organizational requirements are liable to change, and the solution may need to be consequently updated. This initiates another phase of inception, triggering new iterations of ARKnowD's main activities. Note that, in this case, information systems and new organization divisions are now included as new agents in the requirements analysis phase.

3.4 The Use of Agent Mentalistic Concepts

As stated in section 2.4, an AI perspective of agents characterize it by its cognitive (or mentalistic) properties, such as beliefs, goals and commitments. Although a software engineering view on agents emphasize its potential for information system development, without too much care for mentalistic or cognitive notions, we here argue that such properties may indeed favor both domain and system analysis and design. This is especially motivated by the fact that we apply agents as metaphors to model members of organizations. Thus, it may be interesting to consider these properties to explicitate particularities of human relations, along with their interaction with information systems. In this sense, notions such as *goals*, *plans* and *dependency* adopted by Tropos, along with *belief*, *perception* and *commitment* supported by AORML made these notations particularly attractive for modeling our view of human organizations. Particularly for Constructivist KM support, these cognitive notions assist the analyst in the task of capturing the peculiarities of the organization's environment and culture. Understanding agent's goals, perceptions and beliefs lead to a deeper comprehension of the values and strategies adopted in the organization, thus contributing for the conception of more suitable practices and information systems to enable Constructivist KM.

Several agent cognitive models are proposed in AI literature, the most

well-known of them being the BDI model (Rao and Georgeff, 1991). This approach models agents by focusing on the three basic mental components of *belief*, *desire* and *intention*. Belief refers to knowledge the agent has about the environment and about the other agents with whom he/she interacts. Desire refers to the particular "will" of the agent towards a specific state of affairs of the world (goal), although he/she might never actually pursue these goals. And finally, intention leads to specific plans and commitments to achieve specific goals. Among the practical use of such framework, Rao and Georgeff (1991) present a possible-words formalism for BDI-architectures, while the InteRRaP architecture proposed by Fischer et al. (1996) takes a less formal view of this model, realizing it through a set of three layers that aim at explaining agent behavior and supporting system design.

A different model characterizes the state of an agent as a combination of mental components such as beliefs, capabilities, choices, and commitments (Shoham, 1993). However, according to (Wagner, 2003), both this and the BDI model fail to recognize two fundamental elements for the design of agent-oriented information systems: *perceptions* of environmental events and messages of other agents, which form the basis for the agent's reactive behavior; and *memory* of past events and actions.

Notions such as beliefs, goals and commitments have been largely used to model multiagents' cooperation and teamwork (Pynadath et al., 1999). For instance, CAST (Yen et al., 2001) has been developed to simulate and support teamwork within mixed human/agent teams. The petri-net based formalism applied in CAST enables it to represent besides *belief* about the world, also belief about the current goals and activities of others in the team. In addition to that, CAST generates representation of *roles* and *responsibilities*, along with individual and team *plans*. Based on the shared mental model, CAST agents are able to decide on the fly how to accomplish desired *goals*, how to select responsibilities to *commit* to or *delegate*, how to proactively assist others in the team, and how to effectively communicate within the team.

CAST applies a knowledge representation language called MALLET (a Multi-Agent Logic-based Language for Encoding Teamwork), which models

beliefs, roles, responsibilities, and capabilities (Yin et al., 2000). Furthermore, Mallet considers both plans and goals, having plans being decomposed to fulfill agent goals. Other work on cooperation that deal with goals and plans, linking them to the notions of beliefs, desires and intention include (Boella et al., 1999).

Currently, research in this area has shifted its focus from the individual characteristics of an agent to a view of multiagent systems as organized bodies. In this sense, coordination and control should be addressed as organization-centric instead of agent-centric, as earlier (Sichman et al., 2005). This has given life to a new research area known as *Agent Organizations*. Research in this area is much in line with our aim at modeling human organizations, our work being particularly focused on the application area of KM. Thus, within the work on agent organizations, we are particularly interested in the proposed modeling frameworks.

Several such frameworks have been proposed. Among them, we have already mentioned OperA (Dignum, 2004a), which besides being a software engineering methodology, has been specifically tailored for modeling agent organizations. As described in section 2.4.2, the organizational model proposed by OperA views an organization as a set or *roles* that abstractly describe the functional position agents will later occupy. Besides, this model: a) defines the coordination structure followed by the organization (such as hierarchy, market or network); b) models interaction moments through *scenes*' description; and c) prescribes *norms*, associated to roles and interactions. Besides OperA, AGR (Ferber et al., 2004), MOISE+ (Hubner et al., 2002) and ISLANDER (Esteva et al., 2002) are among the most prominent examples of agent organizations frameworks. AGR focuses on a structural view of the organization, modeling it as a set of *agents*, *roles* and *groups*. Agents are active, communicating entities playing (multiple) roles within (several) groups. Besides a structural view similar to that of AGR, MOISE+ also presents functional and deontic aspects. On one hand, functional aspects concern *organizational goals*, which are decomposed in *plans* and assigned to agents through *missions*. On the other hand, deontic aspects describe roles' *permissions* and *obligations* concerning missions. In MOISE+, roles may be

related by communication and authority links. In ISLANDER, the organization's roles and their relationships are presented in the *dialogic framework*, which also prescribes a common ontology that guides agent's communication and knowledge exchange. Interactions in ISLANDER are modeled as *scenes*, which are then related through *performative structures*. And finally, ISLANDER also models norms through the definition of commitments, obligations and rights of participating agents.

It is worth mentioning that much work related to philosophy and cognitive sciences have supported the definition of the cognitive notions mentioned here, guiding its practical use for modeling and developing multiagent systems. Among these is the early work of Bratman (1987) on goals, beliefs, intentions and related mental models, and the contribution of Castelfranchi and colleagues on commitments (Castelfranchi, 1995), dependency (Castelfranchi et al., 1992), and delegation (Castelfranchi and Falcone, 1998). In addition to these, work on conceptual formalization through the use of ontologies have also provided valuable contribution in this respect (Guizzardi, 2005) (Bottazzi and Ferrario, 2005).

Because the definition and treatment of such cognitive concepts are different in each work, we found important to provide our own conceptualization. Hence, in the sequence, we present an ontology of agent and related concepts. In this work, we try to merge the different existing views, sometimes combining them, sometimes compromising one idea in favor of another. It is important to add that here, we particularly apply this ontology for guiding the understanding and the evaluation of the notations adopted by ARKnowD (i.e. those of Tropos and AORML). Furthermore, we aim with such initiative, to give a step forward in the direction of clarifying agent-related concepts, thus contributing to the state of the art in the area.

3.5 Towards an Ontology for the Domain of Agents

An ontology can be defined as a specification of a representational vocabulary for a shared domain of discourse (Guizzardi, 2005). In other words, an ontology is a domain model, composed of a set of concepts and relations. An ontology is useful to provide a clear understanding about a certain domain. Its concepts and relations provide the precise meaning of the domain's concept. Here, an ontology is applied to clarify ARKnowD's concepts, inherited both from Tropos and AORML.

We base our *Agent Ontology* on the foundation ontology previously defined by Guizzardi and Wagner (2005). According to these authors, a foundation ontology, or upper level ontology *"defines a range of top-level domain-independent ontological categories, which form a general foundation for more elaborated domain-specific ontologies"* (Guizzardi and Wagner, 2005, pg. 346). These foundation ontology is divided into three incrementally layered compliance sets: 1) UFO-A defines the core of UFO, excluding terms related to perdurants (i.e. processes) and terms related to the spheres of intentional and social things; 2) UFO-B defines, as an increment to UFO-A, terms related to perdurants; and 3) UFO-C defines, as an increment to UFO-B, terms related to the spheres of intentional and social things. This section briefly describes UFO-A and UFO-B, focusing only on the concepts that are important for the complete understanding of our ontology. Following, we present UFO-C in detail, extending it to create our ontology.

3.5.1 UFO-A: Endurants and Perdurants

Figure 3.4 shows an excerpt of UFO-A. UFO-A distinguishes between two kinds of individuals: **endurants** and **perdurants**. This distinction can be intuitively understood in terms of the distinction between "objects" and "processes", respectively. An **endurant** does not have temporal parts, and persists in time while keeping its identity. Examples of endurants include a house, a person, a hole, the (objectified) color of an apple, and an amount

of sand. A **perdurant**, conversely, is composed of temporal parts. A storm, a heart attack and a business process are three examples of perdurants.

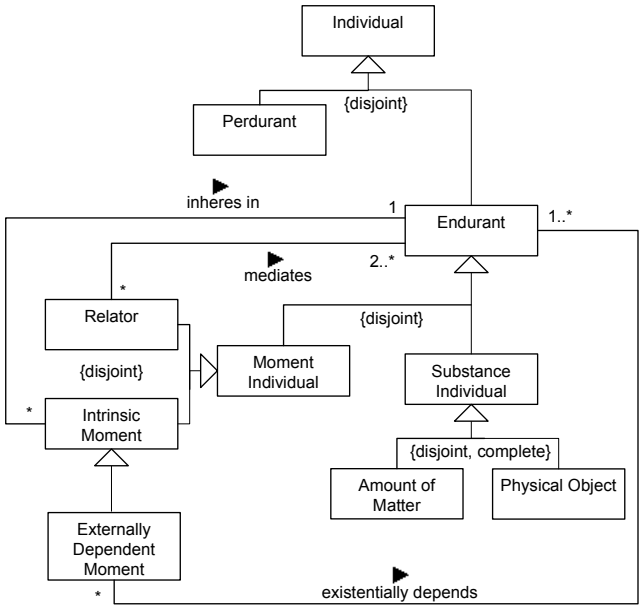


Figure 3.4: Different kinds of individuals in UFO-A

Endurants are further specialized into **substance individual** and **moment individual**. The former refers to an endurant that possesses direct spatio-temporal properties and can exist by itself, i.e. substance individuals are not existentially dependent on other endurants, except possibly on some of its parts. A building, a person and a dog are examples of substance individuals. A **moment individual**, however, is an endurant that cannot exist by itself; that is, it existentially depends on other individuals (e.g. the age of a person, a belief of an agent). Making an analogy with the object-oriented software engineering domain, we can understand the difference between *substance* and *moment* comparing them respectively to *object* and *(objectified) property*.

A moment individual can be either an **intrinsic moment** or a **relator (or relational moment)**. An **intrinsic moment** is a moment individual that is existentially dependent on one single individual (e.g., the color of an apple depends on the existence of the apple itself). Meanwhile, a **relator** is a moment individual that is existentially dependent on more than

one individual (e.g., a marriage, an enrollment between a student an educational institution). In other words, a relator is an individual capable of connecting or **mediating** entities (Guizzardi, 2005). For example, we can say that John is married to Mary because there is an individual marriage relator that existentially depends on both John and Mary, thus, mediating the two. Likewise, we can say that Lisa works for Xerox because there is an employment relator mediating Lisa and Xerox.

We can say that endurants **bear** moments, or inversely, that a moment **inheres in** an endurant. The relation of *inherence* is a special type of existential dependence relation between moments and their bearers. Formally, besides existential dependency, inherence implies the so-called *non-migration principle* (Guizzardi, 2005), i.e., if a moment X inheres in an individual Y, then there is no individual Z distinct from Y such that X inheres in Z. In other words, inherence is a functional existential dependence relation. This way, Fig. 3.4 particularly emphasizes that an intrinsic moment inheres in one single endurant. An **externally dependent moment** is a special kind of intrinsic moment which although inhering in a specific endurant, also existentially depends on another one. The employee identifier is an example of externally dependent moment, since although inherent to the employee, is also dependent on the organization where this employee works. A relator R mediating the individuals A and B inheres in the individual composed of A and B (the so-called mereological sum of A and B)(Guizzardi, 2005) and, due to the aforementioned non-migration principle, this individual cannot change. In other words, R inheres in (and, thus, is existentially dependent on) exactly that specific collection of individuals formed by A and B.

A substance individual is further specialized into **amount of matter** and **physical object**. A **physical object** satisfies a condition of unity, for which certain parts can change without affecting its identity (e.g. a house, a person, the moon). Conversely, an **amount of matter** is a substance individual that does not satisfy a condition of unity, typically referred to by means of mass nouns (e.g. a lump of clay, a pile of bricks, an amount of sand).

In Fig. 3.4, we emphasized that all specializations are disjoint, meaning

that if an individual is an instance of one specialization class, it can not be instantiated by another specialization class with the same parent. All specialization relations described in this section have this nature. Hence, we refrain from providing such details in the subsequent pictures in order to simplify the models. The above information presented for the individual level may be also replicated for the type level. Figure 3.5 shows an **entity** may be either an **individual** or a **type**, the former instantiating the latter. So for example, the substance individuals John, Mary and Lisa instantiate the substance type Person.

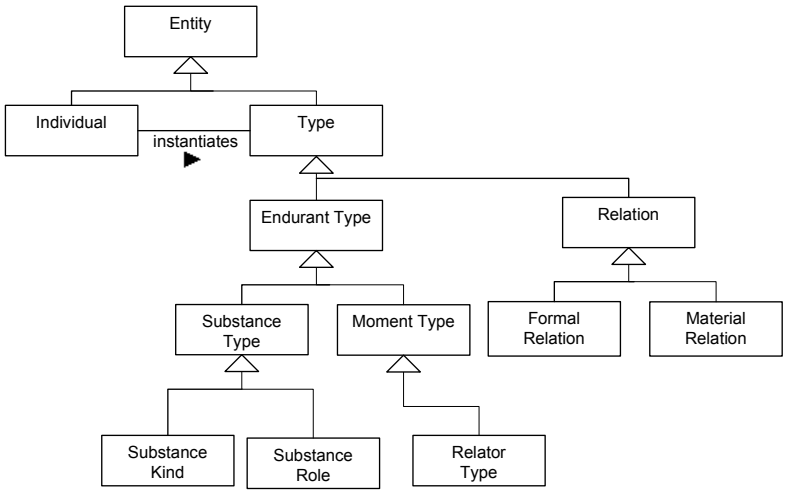


Figure 3.5: UFO-A differentiating between Kind and Role

Fig. 3.5 shows that for the category of substance types, UFO-A makes a further distinction based on the formal meta-properties of **rigidity** and **anti-rigidity**. In simple terms, a type T is said to be rigid if every instance x of T is necessarily (in the modal sense) an instance of T. In other words, x cannot cease to instantiate T without ceasing to exist. Conversely, a type T is anti-rigid if every instance x of T is possibly (in the modal sense) not an instance of T, i.e., if x can cease to instantiate T without ceasing to exist (Guizzardi, 2005). A stereotypical example highlighting this distinction is given by the types person and employee, both instantiated by the individual Lisa in a given circumstance. Whilst Lisa can cease to be an employee of Xerox (and there were periods of time in which Lisa was not one), she cannot

cease to be a person. In this thesis, a substance type that is rigid is named a **Kind**. In contrast, a substance anti-rigid type is named a **Role**.

Besides highlighting this important difference within the category of substance types, Fig. 3.5 also presents other entities. A **relation** is a type whose instances are tuples of connected elements. For instance, taking Lisa's example presented above, the 'works at' relation connects Lisa to Xerox. There are two types of relations: **formal relation** and **material relation**. A **formal relation** holds between two or more entities directly, without any further intervening individual (Guizzardi, 2005). Examples of formal relation include Lisa 'is older than' Mike, and John 'is taller than' Mary. As pointed out in (Guizzardi, 2005), the relata of these relations are in fact moments and not substance individuals. To say that Lisa 'is older than' Mike is to say that Lisa's age is bigger than Mike's age. Moreover, the relation between Lisa and Mike exists without the need for any real connection between the two. To put it differently, these relations between substantials are reducible to purely formal relations between intrinsic moments of the involved relata. Guizzardi (2005) also points out that instantiation, inherence and existential dependency are all types of formal relations.

Conversely, **material relations** are founded on the existence of a relator. Thus, Lisa 'works at' Xerox because there is an employment relator connecting the two. This employment can be composed, for example, of all commitments and claims associated with the role Lisa plays at that organization, and vice-versa (i.e. by all commitments and claims associated to the organization towards Lisa). Later in this section we provide a more extensive discussion on commitments and claims. Likewise, John 'is kissing' Mary because there is an individual kiss connecting the two. In summary, differently from formal relations, material relations are not reducible to relations between intrinsic moments of the involved relata.

3.5.2 UFO-B: an Ontology of Perdurants

Figure 3.6 presents UFO-B, in which the concept of **perdurant** coming from UFO-A is further specialized into **state** and **event**. A **state** is a perdurant

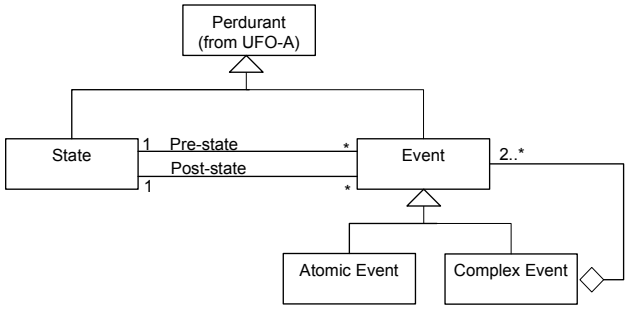


Figure 3.6: UFO-B: understanding perdurants in details

whose temporal parts belongs to the same state type as the whole. An **event**, on the other hand, is a perdurant that is related to exactly two states (its pre-state and its post-state). Pre-state and post-state are shown in the relations between event and state in Fig. 3.6.

An event is then specialized into **atomic event** and **complex event**. The former refers to an event that happens instantaneously, that is, an event without duration, for instance: an explosion, or a message reception. The latter is an event that is composed of other events by means of event composition operators. Examples of complex event comprehend a parallel occurrence of two explosions, a storm, a heart attack, and a work meeting. A *process* can be understood as a synonymous of complex event, i.e. an event that is composed of two or more events as shown in Fig. 3.6.

3.5.3 Extending UFO-C

UFO-C is based on the concepts of **physical object** and **moment individual** coming from UFO-A, and on the concept of **event** specified by UFO-B. Our extended version of UFO-C is depicted subsequently, starting from Figure 3.7 until Figure 3.10.

Fig. 3.7 shows that the UFO-A concept of **physical object** is here specialized into **physical agent** and **non-agentive object**. A **physical agent** is a physical object that creates **action events**, perceives **events** (possibly created by other **physical agents**), and to which we can ascribe a mental state. Here are some examples of physical agents: a man, a cat, a

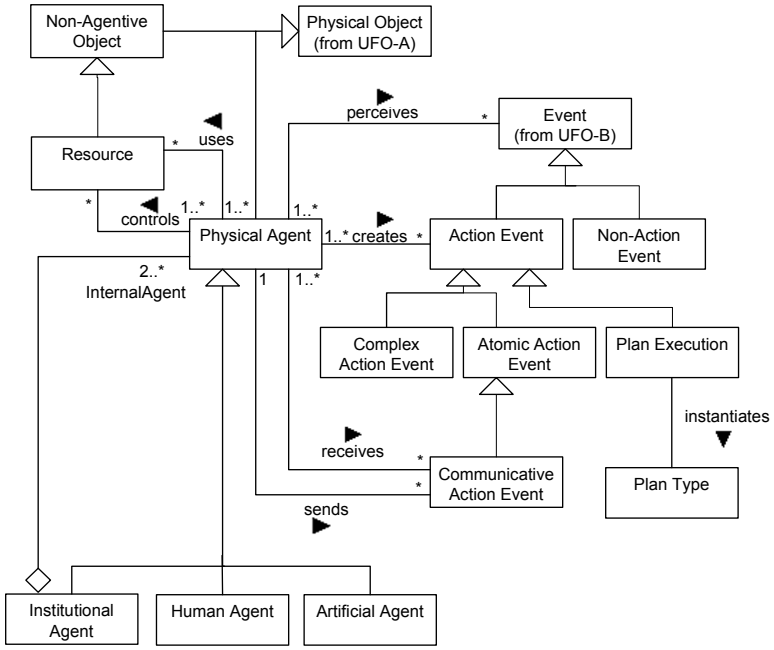


Figure 3.7: Extending UFO-C from the UFO-A concept of physical object and the UFO-B concept of event

robot. A **non-agentive object** is a physical object that is not a physical agent (e.g. a book and a tree). A **non-agentive object** can be a **resource**, meaning that such object is used by a **physical agent** with specific purposes, and typically owned or controlled by this or other **physical agent** (relation **owns** and **controls** outcoming from physical agent).

A distinction is made between **human agent**, **artificial agent** and **institutional agent** (all three sub-kinds of physical agent), to differentiate humans agents, software (or hardware) agents, and agents representing organizations or organization sub-parts (such as departments and divisions). Institutional agents are composed of several internal agents, which may be any kind of physical agent (human, artificial or institutional).

Most agent-oriented approaches only focus on agents, disregarding the presence of objects in the modeled scenario. We consider this a limitation and thus, acknowledge the existence of these two distinct entities. Especially in KM settings, some connections can be intuitively identified between the *knowledge artifacts* and *objects*, between the *KM system users* and *human*

agents, and between a KM supporting *organization* as *institutional agents*. Besides, the KM system itself can also be composed of multiple software agents that apply non-agentive objects as resources, mediating the processes of knowledge creation, integration and sharing.

Action event and **non-action event** are two types of **event** (concept from UFO-B). The former refers to an event that is created through the action of a physical agent, for instance, 'sending a message to another physical agent', 'writing a book', and 'reviewing a paper'. The latter is an event that is not created through an action of a physical agent (e.g. 'a deadline is achieved', and 'it becomes dark'), although it may be perceived by him/her. This differentiation is essential in agent-oriented approaches as modeling the environment populated by agents is paramount. Therefore, non-action events are typically events generated by the environment itself and perceived by the agents living in it. A **plan execution** can be defined as an intended execution of one or more actions, being in this way a special kind of action event. In other words, a plan execution may be composed by one or more sequentially ordered action events, targeting a particular outcome of interest to the agent. These action events may be triggered by both action and non-action events perceived by the agent. Besides, a plan execution is connected to a **plan type**, which is a general description of the action sequence that a physical agent should execute.

Analogously to an UFO-B atomic event, an **atomic action event** is an action event that happens instantaneously, such as 'picking a book in a shelve' and 'sending a message'. In fact, 'sending a message' can also be seen as a subtype of atomic action event, referred to as **communicative action event**. Physical agents both send and receive communicative action events. Communication is one of the most important aspects of agent-oriented systems as this triggers one agent to adopt **goals** or to execute **action events** on behalf of another. As already pointed out in section 2.4, unlike objects that simply execute actions when requested, an agent reasons over another agent's request before agreeing on a particular course of action (Wooldridge, 1999). Communication may be also required to inform an agent about changes in one's course of action or in the environment itself,

thus altering the agent's beliefs.

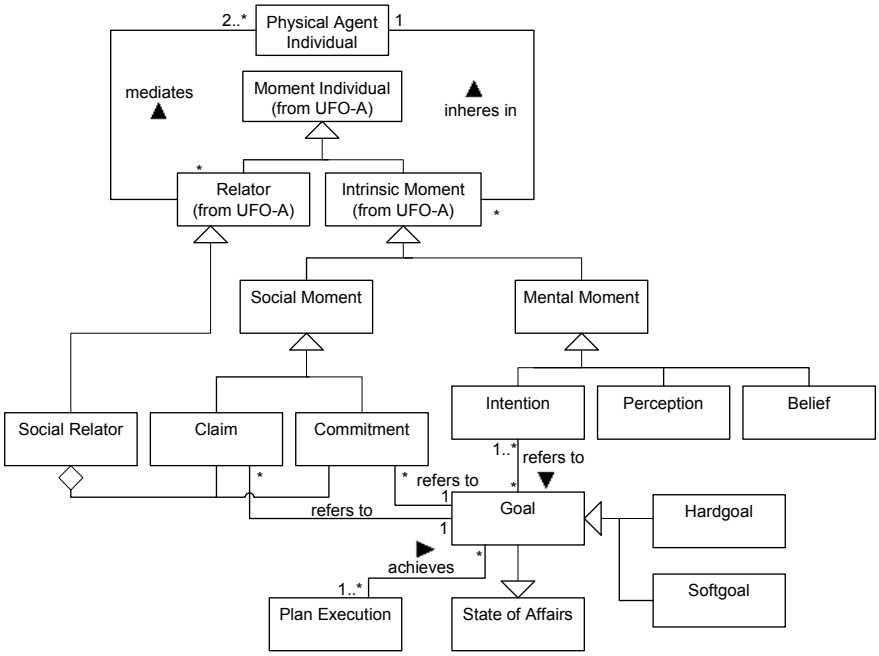


Figure 3.8: Extending UFO-C from the UFO-A concept of moment individual

In Fig. 3.8, the **intrinsic moment** concept of UFO-A is specialized into **mental moment**, which denotes an intrinsic moment that is existentially dependent on a particular agent, being an inseparable part of its mental state. Examples of mental moments include a thought, a perception, a belief, and an individual intention. We can then say that a **mental moment** inheres in a **physical agent** (relation **inheres in**). Here, we choose the mental moments that we find more useful for modeling agent-oriented systems. **Perception** is a relevant concept to express the relation of agents to events sensed from the environment and from other agents. **Belief** regards information the agent has about the environment and about other agents. In KM settings, belief and perception are highly conditioned by organizational culture, i.e. the views, values and behavior socially accepted within the organization's boundaries.

Organizational culture may also constrain the action of agents, and this is highly related to another important type of agent's mental moments, namely,

intention. Agent's intention directly leads to the adoption of certain goals and objectives. Taking this into account, we here specify **goal** as a particular **state of affairs** (i.e. condition or state of the world) related to an intention inhered in a **physical agent** (see the **refers to** relation between **goal** and **intention**). A goal may be specialized into **hardgoal** and **softgoal**. Here, we adopt Tropos' definitions that state that a hardgoal is associated a specific condition for verifying whether it has been satisfied or not. A softgoal, instead, has no clear-cut satisfaction condition. Due to this fuzzy nature of softgoal, we decide not to go forward with the analysis of this concept here, leaving it as future work. However, since we apply this concept in the analysis we make in this thesis, we found it important to include it in the ontology.

Social moment is an specialization of the UFO-A concept of **externally dependent moment**, including the concepts of **commitment** and **claim**. When two physical agents agree to accomplish goals to one another, a commitment/claim pair is generated between them. These deontic concepts are highly important to regulate the social relations between members of an organization. In KM environments, agents may have several commitments and claims towards one another. On one hand, a consultant might commit to his colleague to pass on some valuable information about a past case that he was involved in, which is similar to a present task of his colleague. On the other hand, the colleague can claim this knowledge transfer from the consultant. A pair commitment/claim constitutes a **social relator**, which is a particular type of UFO-A **relator**. Fig. 3.8 also shows that a commitment and claim refer to a goal (**refers to** between commitment and goal and between claim and goal). In other words, when a physical agent A commits to a physical agent B, this means that A adopts a goal of B. Conversely, the social relator created between A and B state that B has the right to claim the accomplishment of this specific goal to A. Castelfranchi (1995) made an important contribution on the understanding of commitments (and consequently also on the clarification of claims). In one of his work, he cites Searle, who claims that "a commitment is a right producing act" (Castelfranchi, 1995), highlighting that it is much more complex for an

agent to disengage from commitments towards other agents (social commitments, in Castelfranchi's term) than to dismiss his own intentions (which Castelfranchi calls internal commitments).

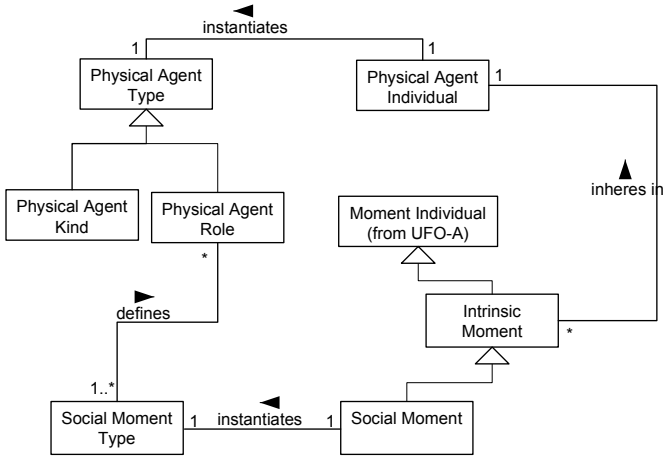


Figure 3.9: Pointing out the difference between physical agent type and physical agent role

Fig. 3.9 emphasizes the difference between **physical agent type** and **physical agent individual**. Furthermore, it also depicts the difference between rigid and anti-rigid agent types, here **physical agent kind** and **physical agent role**. While person is an example of a physical agent kind, physical agent roles are specifically suited to model organizational roles (e.g. secretary, manager) as well as other roles performed by agents in specific situations that can be played independently of the position someone has in an organization (e.g. 'coffee maker' or 'book reader'). As previously clarified in UFO-A, a person cannot cease to be a person while a secretary can be promoted into manager, or can assume another organizational position. This distinction is characterized by the rigidity of **physical agent type** and the anti-rigidity of **physical agent role**.

Still aiming at clarifying the concept of agent role, Fig. 3.9 shows that an agent role is defined by **social moment types**, which describe what the set of *general commitments* and *general claims* a physical agent playing that role has. This is again based on the work of Castelfranchi, who defines a general commitment as a commitment an agent makes towards a set of goals of the

same type. For example, when agreeing to perform the organizational role of a 'secretary', one is automatically committing oneself to 'writing letters' and 'making appointments' on behalf of one's boss. Conversely, this person also has some claims a priori, such as receiving a certain salary in the end of the month and having a suitable working place. Bottazzi and Ferrario (2005) reminds us that an agent's autonomy within an organization is restricted by the set of general commitments and claims he/she has, as a result of playing a specific role, also highlighted in (Dignum, 2004a).

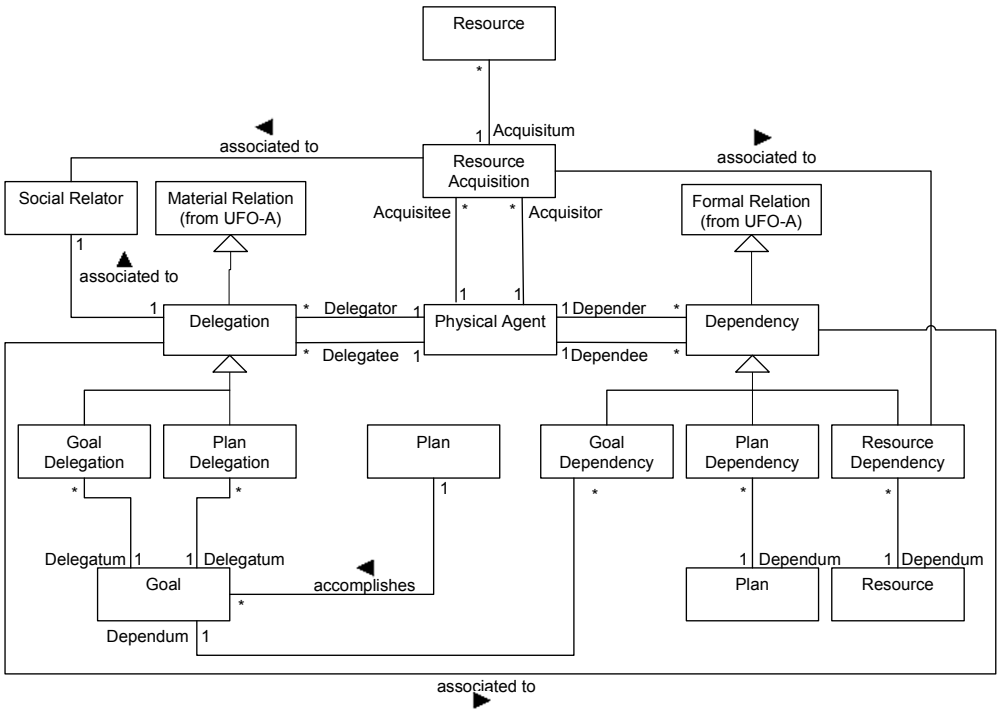


Figure 3.10: Distinguishing between dependency, delegation and acquisition relations

Figure 3.10 finally concludes our UFO-C extension, depicting the important distinction between the concepts of dependency and delegation. The first difference regards the fact that while a **dependency** constitutes a formal relation, a **delegation** consists of a material relation, following the definitions of UFO-A. Let us examine this difference in further detail. The figure shows that a dependency connects two physical agents (a **depender** and a **dependee**) and a **dependum**, whose nature defines the type of

dependency. Thus, a goal dependum indicates a goal dependency, a plan dependency is created around a plan, and a resource is a dependum of the resource dependency. An agent A (the depender) depends on an agent B (the dependee) regarding a goal G if G is a goal of agent A, but A cannot accomplish G, and agent B can accomplish G. Here, the fact that an agent cannot accomplish a goal may mean that this agent either does not have the ability to achieve it. Or else, it may denote that this agent's pursuit towards this goal may interfere with his/her other intentions, such that he/she decides not to pursue this goal after all. This may well be a reason why agent A decides to delegate such goal accomplishment to agent B. A delegation is thus associated with a dependency but it is more than that. As a material relation, it is founded on something more than its connected elements. In this case, the connected elements are two physical agents (**delegator** and **delegatee**) and a goal (**delegatum**), and the foundation of this material relation is the social relator (i.e. a commitment/claim pair) established between the two physical agents involved in this delegation. In other words, when agent A delegates a goal G to agent B, besides the fact that A depends on B regarding G, B commits him/herself to accomplish G on behalf of A. Goal and plan delegation refer to what Castelfranchi defines as open and close delegation (Castelfranchi and Falcone, 1998), meaning that the former leaves the decision regarding the strategy towards goal accomplishment to the depender. The latter rather prescribes a specific strategy (i.e. a plan) the depender should adopt towards achieving the delegated goal.

To illustrate the difference between dependency and delegation, consider the following case. Suppose John is a program committee member of a certain conference and that he received from Paul (the conference program chair) an article X to review. Suppose that John cannot review this article by himself, since there are some aspects of the article which are outside his field of competence. Now, suppose that George is a colleague of John who is knowledgeable exactly in those aspects that John needs to review article X. In this case, we could say that John *depends on* George to review article X. Notice, however, that this relation between John and George can be reduced to relations between the goals and capabilities of these individual

agents. Moreover, this relation does not even require that the related agents are aware of this dependence. This is certainly not the case for the relation between Paul and John. As the program committee chair, Paul depends on John to review article X. However, in this case, not only they are both aware of this dependence but there is the explicit commitment of John to Paul to review article X. In other words, the delegation of Paul to John to review article X cannot be reduced to relations between their intrinsic moments, but it requires the existence of a certain relator (a commitment/claim pair) that founds this relation. Not explicit in the diagram of Fig. 3.10 is the concept of *socially can achieve*, or *socially can execute*. In the paragraph above, when we say that a certain agent can achieve a goal, this means that such agent is able to do it him/herself or can delegate to another agent that can accomplish it on his/her behalf. In the example above, if John can review part of article X by himself and can delegate a remaining part to George, we could say that John *socially can achieve* the goal of reviewing article X.

Similarly to delegation, **resource acquisition** is also a material relation associated with the same concepts of dependency and social relator. We created this as a different concept because when agent A needs access to a resource R controlled by agent B, it is awkward to say that agent A delegates resource R to agent B. Moreover, this relation is differentiated as follows: an agent A acquires a resource R from agent B is equivalent to say that agent A needs to use resource R, agent A does not control resource R, agent B controls resource R, and agent B commits him/herself to give agent A access to resource R. In an alternative formulation we can say that if agent A acquires resource R from agent B then: a) there is a resource dependence from A to B w.r.t. R; b) A and B are mutually aware of this dependency; c) B socially commits to give A access to R.

3.6 Evaluating ARKnowD's Notation

When conceiving a novel modeling language, one should obviously be concerned with its quality. This quality traduces in how well the language is able to represent phenomena in its domain of discourse, and on how clearly the language is able to communicate such phenomena to the eventual readers of the model. ARKnowD combines the notations of two different modeling languages, the ones of Tropos and AORML. It is thus important to verify the quality of these languages individually, but especially the consistency in their combination to generate ARKnowD's language.

3.6.1 Evaluation Method

Guizzardi (2005) provides a framework for evaluating modeling languages. This framework verifies how clear and expressive a language is, by focusing on its notation, but also evaluates how well this language is able to represent the state of affairs for which it is proposed (also referred in this work as *domain appropriateness*).

"The domain appropriateness of a language is a measure of the suitability of a language to model phenomena in a given domain, or in other words, of its truthfulness of a language to a given domain reality. (...) Comprehensibility appropriateness refers to how easy is for a user a given language to recognize what that language's constructs mean in terms of domain concepts and, how easy is to understand, communicate and reason with the specifications produced in that language."(pg. 28)

The proposed framework is based on the construction of a domain ontology to describe the conceptual domain of discourse. This ontology is then used as a type of 'mirror' for the modeling language, i.e. for verifying how well this modeling language is able to represent the concepts and relations represented in the ontology. This verification results is a measure of the quality of the domain appropriateness of the given language.

Given the ontology elaborated and described in the previous section, we intend to apply this method to evaluate ARKnowD's language. The eval-

uation criterion is based on four properties, namely: *lucidity, soundness, laconicity* and *completeness*.

A language is considered *lucid* according to a conceptualization if each of its constructs can represent at most one entity of this conceptualization. Although not exactly the same, lucidity is closely linked to construct overload, i.e. having a single language construct representing two or more ontological constructs. As stated by Guizzardi (2005, pg. 31), *"Construct overload is considered an undesirable property of a modeling language since it causes ambiguity and, hence, undermines clarity. When a construct overload exists, users have to bring additional knowledge not contained in the specification to understand the phenomena which are being represented."*

Soundness refers to the property of a language of representing solely the entities of the domain conceptualization. Having one construct that does not map to any ontological construct is also known as construct excess. The presence of this extra construct should be avoided since it undermines the understanding of the specification. In other words, a specification is clear if the reader is able to link the language constructs to the entities of the domain of discourse. Consequently, only the entities of this domain (represented in the domain ontology) should be modeled with the use of language constructs.

A language is said *laconic* if it possesses only one construct to represent each phenomenon in the domain or discourse (i.e. each entity in the domain ontology). Conversely, the same conceptual entity may be represented by two or more constructs in a specification, consequently adding confusion to the meaning of the model. A reader may ask himself, for example, if the two constructs are actually the same or if there is any semantic distinction between them. Laconicity is then related to construct redundancy, which besides turning more difficult the understanding of specifications, adds unnecessary complexity to the modeling language.

A modeling language is said to be *complete* if every concept in a domain conceptualization is covered by at least one modeling construct of the language. This is directly linked to the expressivity of the given language. In other words, if a language is incomplete, it fails to represent all phenom-

ena in the given domain of discourse. The result of this incompleteness is either an incomplete specifications or construct overload, which are both undesirable for deteriorating the clarity of the specifications produced with the given language.

3.6.2 Evaluation

Taking the ontology presented in section 3.5 and based on the method described above, we have found a few problems in the current Tropos and AORML notations. Consequently, we have decided to make a few adjustments in order to proceed with their integration into ARKnowD. It is important to point out that these notations are here considered in combination with one another, so for example, if one comprehends a set of ontological concepts, the lack of these same concepts in the other is not considered incompleteness. This decision is motivated by the fact that in ARKnowD, each notation is used in a separate activity, for which one concept or another may be more appropriate.

In Tropos, there are one case of lack of *laconicity*, one case of unsoundness, two cases of *incompleteness*, and one case of missing *lucidity*. First, let us address the *lack of laconicity* and the *unsoundness* cases together. In Tropos, besides the concept of agent and role, corresponding to our ontological concepts of physical agent type and physical agent role, there are two other concepts: actor and position. Figure 3.11 depicts these concepts and their corresponding notations.

The concept of *position* is considered solely with the purpose of aggregating different roles. However, for not being a domain concept, position prevents Tropos from being considered laconic. Let us analyze this concept a bit further in order to make sure that it is really not present in the domain. As stated in section 3.5.3, a physical agent role is defined by the set of social moments, i.e. commitments and claims a physical agent playing such role agrees to. As a role aggregation, a position is defined as two or more set of social moments, which can be finally characterized as a social moment. Referring to rigidity or antirigidity, position is an antirigid concept as well

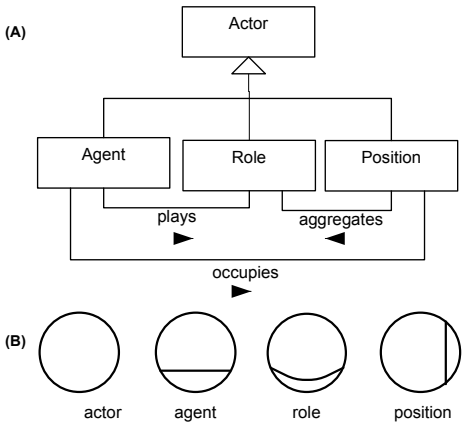


Figure 3.11: (A) an excerpt of the Tropos's metamodel showing the concept of actor and its specializations and (B) corresponding notations

as role. We conclude that there is no real differentiation between role and position, thus not justifying the use of two ontological concepts instead of one.

Still referring to Fig. 3.11, we note that *actor* is a general concept which can refer to an agent, to a role or to a position. However, we find no reason why to consider such a concept, as from the start of a domain analysis, the analyst is able to identify the type of each domain participant, considering the rigidity and anti-rigidity properties explained in section 3.5.1. Thus, the concept of actor leads to unsoundness and we prefer not to consider it in ARKnowD. Dismissing the position and actor concept, the analyst identifies from start, if a domain participant is an agent or a role. For representing an agent, ARknowD adopts Tropos's actor notation (empty circle) for being the most simple form, maintaining the Tropos's role notation as it is.

Going forward with the evaluation of the Tropos language, we address the *incompleteness* and *missing lucidity* issues by considering the case illustrated in Figure 3.12.

Fig. 3.12 illustrates the following situation. The department manager of an organization relies on the department secretary to make an appointment for a meeting with all the employees of the department. The secretary, on her turn, depends on a specific calender system named eDate. The secretary is

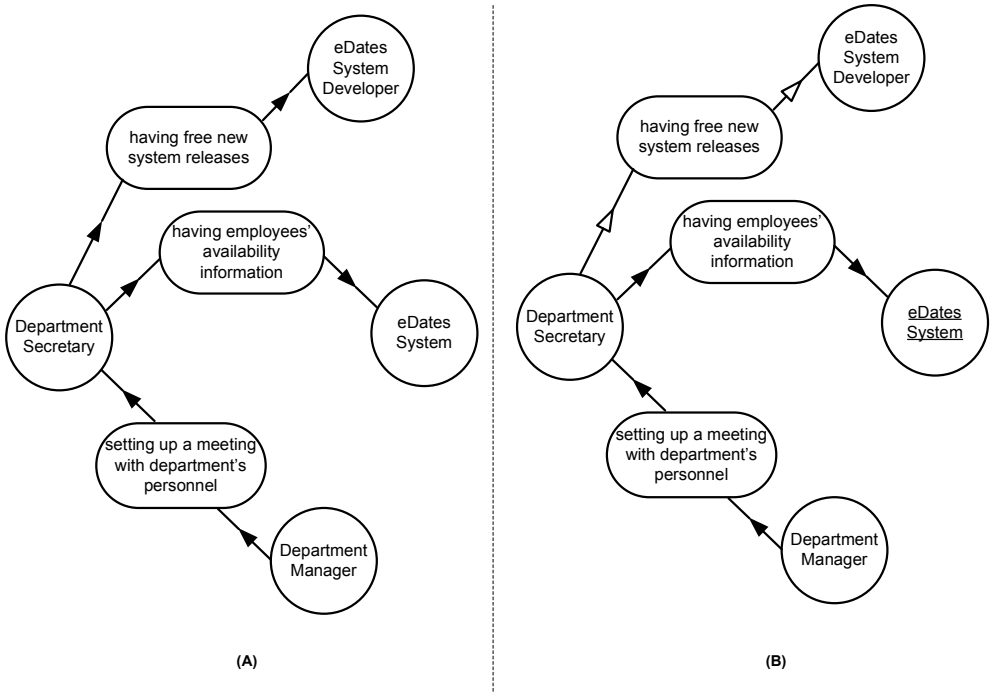


Figure 3.12: Correcting two cases of incompleteness

always checking for free new versions of the eDate system in the developer's website, aiming at profiting from new functionality and enhancements in this system. Department manager, department secretary, eDate, and eDate system developer are example of Tropos agents, which correspond to the UFO-C concept of **physical agent**.

In part (A), however, it is not possible to differentiate between **physical agent type** and **physical agent individual**. For instance, it is not clear if we talk about a specific secretary and a specific system, or general ones. Representing both ontological concepts using only one language construct is understood as construct overload (leading to missing lucidity). This may be in the way for a clear understanding of the modeled setting. We have therefore provided in (B), a way to differentiate these two entities. Inspired by UML, we chose to underline the name of **physical agent individuals** to point out the difference between them and **physical agent type**, thus imitating the way UML differentiates between instances and classes. Our choice provides a good connection between Tropos and AORML notation,

which already adopts this UML strategy. Following such strategy, part (B) depicts eDate as an individual and all other agents as types. The choice of making the others as types is to maintain a level of generality for the model, for instance this case would typically hold even if the secretary was changed. The new one would continue to be responsible for setting up meetings on behalf of his/her boss, and using the same system to do so.

A case of incompleteness that can be noted in Fig. 3.12 (A) refers to the lack of language expressivity of the Tropos language to model the concept of **dependency**. What Tropos usually terms **dependency** is actually a case of **delegation** according to our ontology. As we have seen, the latter is stronger than the previous, as besides dependency it also involves commitment from the dependee in relation to the depender. In part (A) of the figure, the delegation depicted between the secretary and the eDate system developer is actually only a dependency. The secretary does not know the developer, who on his turn has no way to commit specifically to her on releasing new system's versions. In other words, if the developer decides to stop providing free releases and rather to start charging for new eDate versions, the secretary does not have a claim towards him and will just have to live with this new situation. To correct this expressivity problem, we created a new symbol to distinguish dependency from delegation. This is a similar arrow as used before, however empty headed to denote the lack of commitment. This new symbol is illustrated in Fig. 3.12 (B).

The other case of incompleteness, actually refers to the concept of resource acquisition. However, in this case, we simply refer differently to what Tropos formerly termed resource dependency. We also do not see a point in changing the notation in this case, as resource acquisition, goal and plan delegations may be differentiated as shown in Figure 3.13. This notation maintains uniformity regarding the ones used for goal and plan delegation, showing that analogously to these relations, a resource acquisition is a resource dependency added by a commitment (in this case, the commitment of the acquisitee to provide the acquisitor with access to the acquisitum).

In AORML, we have found one case of missing lucidity. This regards the notation used at the same time to model a non-agentive object and a belief.

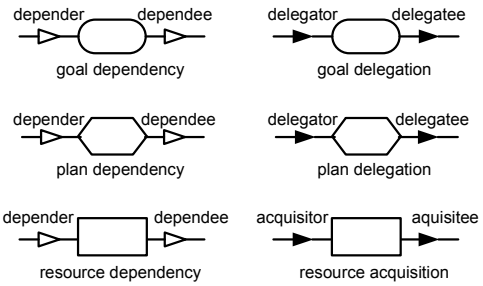


Figure 3.13: Differentiating the three types of dependencies, goal and plan delegation, and resource acquisition

To correct this problem, we use stereotypes, an UML construct commonly used to extend this language, differentiating old and new entities. Such construct is already applied in AORML, for example to distinguish between human, institutional and artificial agents. Figure 3.14 shows our proposed solution, depicting a typical situation involving a library institutional agent and a borrower human agent. The library uses an information system to organize its book collection. The borrower borrows books, having his own internal beliefs related to these books. The figure differentiates the actual books from the agent's internal beliefs by stereotyping the belief class.

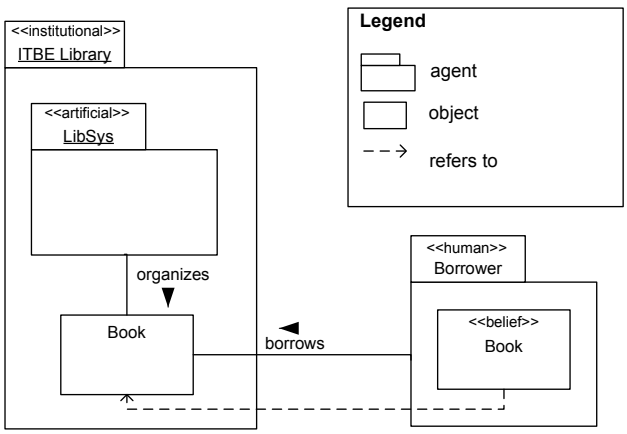


Figure 3.14: Distinguishing beliefs from non-agentive objects in AORML using stereotypes

3.7 MDA-inspired Transformation Method

Recent progress in the development of distributed systems include OMG's efforts towards the definition of a Model Driven Architecture (MDA)[2]. Although aimed at developing object-oriented systems, a few elements of this work may be equally valuable for agent-orientation. This is the case of the concept of *viewpoints* and the idea of *model transformation*. This section discusses both topics and describes how this is applied in the context of ARKnowD.

3.7.1 The Model Driven Architecture Viewpoints

MDA has been developed to enable flexible design of distributed software systems. It provides an open, vendor-neutral approach to avoid problems arising from business processes and technological changes. In this respect, MDA proposes the separation of the specification of the operation of a system from the details of the way that system uses the capabilities of its platforms. In other words, for each system under development, MDA proposes the definition of a platform-independent model (PIM) that can then be transformed into one or more platform-specific models (PSMs). This allows the system to be implemented in different platforms, while still maintaining the same PIM. Besides, benefits of this approach stem from the possibility to partially automate the model transformation process. In this way, development costs may be reduced and software quality may be improved. In addition to that, this approach facilitates integration, evolution and migration of software solutions, hence contributing to the limitation of maintenance costs for distributed applications.

Computational-independent, platform-independent and platform-specific are known as different viewpoints in MDA. *"A viewpoint is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system. Here*

[2]MDA Guide Version 1.0.1, omg/2003-06-01, available at http://www.omg.org/docs/omg/03-06-01.pdf

'abstraction' is used to mean the process of suppressing selected details to establish a simplified model." (MDA Guide Version 1.0.1, pg. 2–3). A computation-independent viewpoint focuses on the environment or domain of the system. At this point, the system's requirements are hidden or undetermined. The result of applying this viewpoint is the development of a Computation-independent model (CIM), also known as domain model or business model. A platform-independent viewpoint focuses on the general functionality of the system, without including the details that are specific of a given platform. In other words, this viewpoint presents part of the system specification that does not change from one platform to another. The platform-specific viewpoint, conversely, is targeted at adjusting the PIM specifications to a certain platform, providing details on how such a platform implements the PIM specifications. The already mentioned PIM and PSM are elaborated when respectively taking a platform-independent and a platform-specific viewpoint of the system.

When a system is developed, its CIM, PIM and PSM must be consistent. In other words, in an MDA specification of a system, CIM requirements should be traceable to the PIM and PSM constructs that implement them, and vice-versa. For maintaining consistency between models, enabling a smooth transition from one viewpoint to another, MDA proposes the use of transformation processes, i.e. processes that convert one model to another model of the same system.

The MDA Guide Version 1.0.1 describes several transformation methods. Here, we limit ourselves on describing the metamodel transformation, which is the one applied in this work. Figure 3.15 illustrates this type of transformation.

As depicted in Fig. 3.15, first, a PIM is specified, using a platform-independent modeling language. Then, a particular platform is selected for implementing the system. At this point, a transformation specification to convert the notation used in PIM to the platform-specific language is already available. This specification maps the metamodels of the language applied in the PIM to the one used in the PSM. Consequently, a PSM may be produced by following the guidelines of the transformation specification. Similarly, a

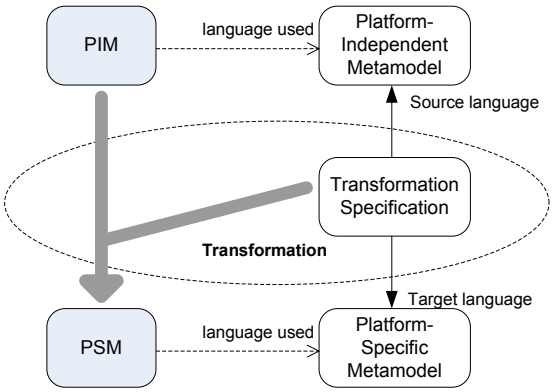


Figure 3.15: MDA metamodel transformation (MDA Guide Version 1.0.1)

CIM may be converted into a PIM.

The described transformation method offers a systematic approach to convert the models elaborated taking different MDA viewpoints. Following, we explore how this is used in the context of the ARKnowD methodology.

3.7.2 ARKnowD's Viewpoints and Models

In this work, we adopt the MDA viewpoints approach, aiming at suppressing unnecessary details according to different abstraction levels, which results in an appropriate separation of concerns regarding system analysis and design. Table 3.1 shows ARKnowD viewpoints framework.

Table 3.1 shows for each abstraction level (or viewpoint), which models are used, according to each modeling aspect, i.e. the interaction, information and behavior aspects. These three aspects are, in general, targeted in every system analysis and design models. The *information aspect* comprehends the entities composing the system and the relations existing among them. A preliminary view of such entities and relations can be obtained in the CIM with the use of Tropos's actor and goal diagrams, which depict the agents and resources (later objects) of the domain and a high-level view of how they relate. However, a complete information model may only be attained through the use of AOR Agent Diagrams (modeling agent and object classes) and UML Class Diagrams (exclusively modeling object classes),

later in the PIM. The *interaction aspect*, as its name suggests, deals with the dynamic aspects of the system, modeling the interactions among the agents composing it. In the CIM, Tropos's actor and goal diagrams, although not providing a detailed model of these interactions, provide an initial view of the interaction relationships among agents through the dependencies, delegations and acquisition links they show. Such interactions are modeled in detail in the PIM with the use of the three types of AOR interaction diagrams. Although characteristic of the PIM, these diagrams may be also applied in the CIM, in case any process of the domain must be analyzed in further details. Finally, the *behavior aspect* focuses on the internal behavior of each system component. Once more, in the CIM, Tropos only provides a high-level view of such behavior, specifically with the use of goal diagrams, which depict the internal perspective of a single agent of the domain. The behavior of the system agents may be better understood in the PIM, by applying AOR pattern and activity diagrams, which provide details about the internal reasoning and choices made by the agents.

As for the PSM, the used diagrams closely depend on the choice of the platform in which the system is finally implemented. Here we exemplify possible models used for each aspect if a Java platform is selected for implementation. In this case, UML Class Diagrams providing more details than the one designed in the PIM are used both for information and behavior modeling. Such diagrams contain all object classes with their respective attributes typed according to the platform, and depict all methods to be executed by an object. Next to this, UML Sequence and Deployment Diagrams model the interaction aspect, respectively providing details on the interaction of objects and the distribution of objects among the system hardware components (i.e. clients, servers, etc.). This thesis focuses more carefully on the CIM and PIM, giving less strength to the PSM, although chapter 6 illustrates how a system may be implemented based on a PSM that refines a specific PIM designed in chapter 5.

The division in three abstraction levels provide us with an interesting view, showing us that we should naturally target the modeling task from different perspectives: the domain model (CIM), a design model which can

Abstraction Level	Viewpoint Aspects		
	Information	*Interaction*	*Behavior*
Computation-independent Model (CIM)	Tropos Actor Diagram, Tropos Goal Diagram	Tropos Actor Diagram, Tropos Goal Diagram	Tropos Goal Diagram
Platform-independent Model (PIM)	Tropos Actor Diagram, AOR Agent Diagram, UML Class Diagrams	Tropos Actor Diagram, Tropos Goal Diagram, AOR Interaction Sequence Diagram, AOR Interaction Pattern Diagram, AOR Interaction Frame Diagram	Tropos Goal Diagram, AOR Interaction Pattern Diagrams, AOR Internal Activity Diagrams
Platform-specific Model (PSM)	UML Class Diagrams, others	UML Sequence Diagrams, UML Deployment Diagrams, others	UML Class Diagrams, others

Table 3.1: ARKnowD's viewpoints

be reused, meaning that it is independent of the implementation platform (PIM), and finally a design model that depends on the implementation platform of our choice (PSM). Referring to section 3.3, we are able to link the models generated as a result of the focus on the different viewpoints explored in table 3.1 with ARKnowD's modeling activities. The models resulting from the requirements analysis activities are typically CIM. Next, the architectural design and the initial detailed design activities generate a PIM. Finally, the design is detailed even further, resulting in a PSM, which enables the system to be implemented using a particular platform and/or programming language.

3.7.3 ARKnowD's Transformations: Converting Tropos into AORML

As previously explained, it is necessary to provide a transformation method to convert the notation of the models of the different viewpoints. In ARKnowD, this is done by mapping Tropos concepts to AORML constructs. Table 3.2 depicts this mapping, previously presented in (Guizzardi et al., 2005).

An agent in Tropos models an entity that has strategic goals and intentionality within the system or the organizational setting. This concept directly maps to one of the three types of agents in AORML: human, artificial or institutional agent, depending on its nature. Tropos's plans may indicate paths for AORML's interaction modeling. In other words, for each plan in a Tropos model, there can be an AOR Interaction Sequence Diagram, modeling the interactions of the agents participating in this plan (i.e. agents having the plan, or being connected to it by a delegation link). Capabilities in Tropos may be seen as a set of plans and, therefore, could be mapped to the set of interaction modeling paths, representing the agent's plans. Analogously, resources that represent physical or information entities in Tropos become objects according to AORML conceptualization. Additionally, in Tropos, goal, plan and resource dependency between two agents indicate that one agent depends on the other in order to achieve some goal, execute

Tropos Concepts	AORML Constructs
agent	agent
plan	AOR Interaction Sequence Diagram
capability	set of AOR Interaction Sequence Diagram
resource	object
dependency	AOR Agent Diagram association relation
delegation	AOR Agent Diagram association relation/AOR commitment
resource acquisition	AOR Agent Diagram association relation/AOR commitment

Table 3.2: Mapping Tropos into AORML

some plan, or obtain some resource. Because such dependency link indicates a kind of relation between the two agents (depender and dependee), an association link may be depicted between these agents in an AOR Agent Diagram (AD), typically used for information modeling. Here, we consider the differences between dependency, delegation and resource acquisition pointed out in section 3.5. As mentioned in that section, besides involving dependency between agents, delegation implies that the delegatee has actually agreed to accomplish a goal or perform a task on behalf of the delegator. Thus, a commitment is established from the delegatee regarding the delegator (or a claim emerges from the delegator towards the delegatee). Therefore, goal and plan delegations leads to the establishment of AORML commitments/-claims between agents, usually depicted in interaction modeling, using one or more types of AOR interaction diagrams. Resource acquisition is treated analogously to goal and plan delegation, since as previously discussed in section 3.5, these concepts have similar nature. In other words, also in the case of resource acquisition, an association link in the AOR Agent Diagram, and a commitment/claim link are assumed to exist between the two agents (the acquisitor and the acquisitee).

Note that one of the most important constructs in Tropos, the concept of 'goal', is not mapped into AORML. This relates to the fact that ARKnowD applies goal modeling exclusively for requirements elicitation and analysis. At design time, all goals have already been dealt with. Goals may have been fulfilled or abandoned. But most commonly, goal analysis leads to the delegation of unsolved goals to new or old actors, who are either part of the organization or a new information system. And finally, concrete plans are assigned to goals with the purpose of accomplishing them. Consequently, when the design activity starts, plans should be modeled rather than goals. As observed in table 3.2, plan modeling may be done through the use of AOR interaction sequence diagram, which details the protocol of communication between agents to realize a specific sequence of actions/interactions.

3.8 Working Example and Methodological Guidelines

In this section, we present a simple example of the use of ARKnowD, with the main purpose of illustrating the transformation between the notations of Tropos and AORML, as described in section 3.7.3. Here, a few modeling guidelines are also presented. More details on the use of the methodology are presented in chapters 4 and 5.

We find the conference review process an appropriate scenario to exemplify ARKnowD. First, this is a well-known setting for the academic community. Furthermore, it has been used elsewhere (Dignum, 2004a), thus enabling the comparison of our approach and notation with those of other methodologies. Figure 3.16 presents a Tropos actor diagram, depicting the main agents of the scenario, along with some goal and resource dependencies between them.

The diagram of Fig. 3.16 shows that the scenario involves the participation of four agents, namely the Conference Chair, the PC Chair, the Paper Author and the PC Member. For realizing the conference, the Conference Chair depends on the Paper Author to submit papers that will be selected

for presentation in the conference (submitting paper goal). For this papers selection, the Conference Chair delegates to the PC Chair the responsibility of selecting the best papers to be published in the conference proceedings (selecting proceedings' papers goal). The PC Chair and the Paper Author have a mutual relationship. While the PC Chair wants to acquire papers submitted by the Paper Author (submitted paper resource), the Paper Author delegates to the PC Chair the goal of having his paper reviewed as part of the papers selection process (having paper reviewed goal). However, the PC Chair does not review all papers on his own. For that, he relies on PC Members (reviewing papers goal). For accomplishing this goal, the PC Member must receive the papers assigned to them (assigned paper resource), along with the review form (review form resource) from the PC Chair.

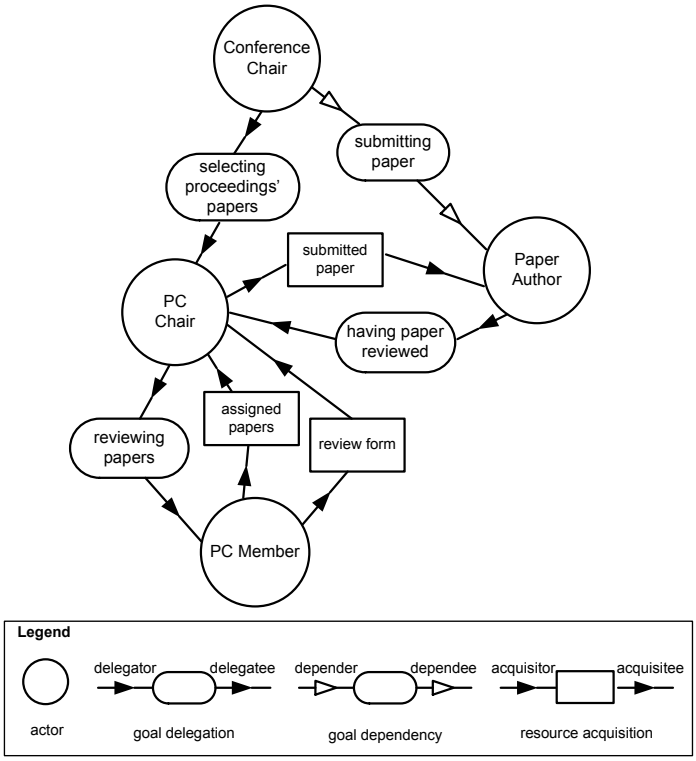


Figure 3.16: Tropos actor diagram depicting main agents and dependencies from the paper review scenario

Modeling is made on the level of 'classes' rather than on the level of 'instances', i.e. agents are depicted as PC Chair and Paper Author, instead

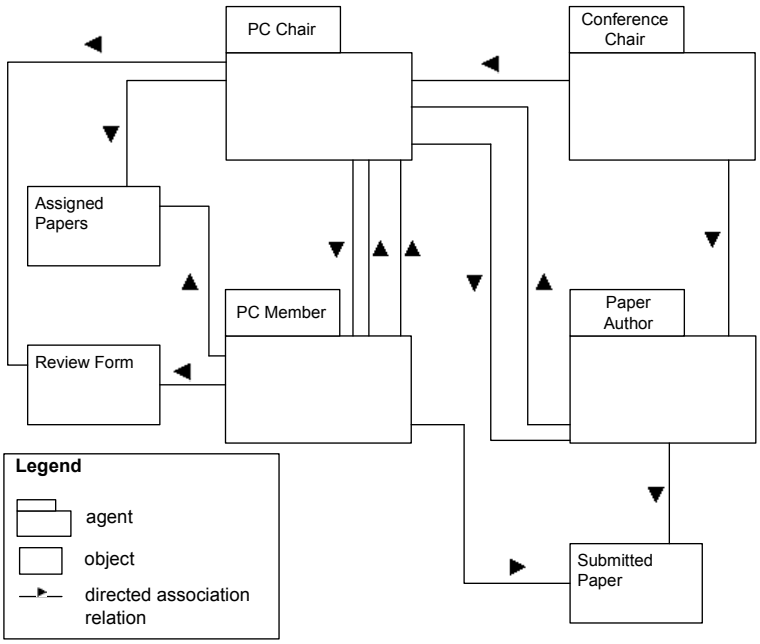


Figure 3.17: AOR agent diagram automaticaly generated from previous Tropos actor diagram

of providing their names. This aims at making the model more general, abstracting away from the details of one specific case. It is also apparent from the diagram depicted in Fig. 3.16 that cardinality is not provided in this model. For instance, there is only one PC Chair, while there are several PC Members. However, the agent names are kept in singular form, as representing the class of agents. Cardinality is subsequently presented in AOR Agent Diagrams.

At this point, we can already exemplify the first transformation. Figure 3.17 depicts an AOR Agent Diagram (AD) that can be automatically generated with basis on the goal diagram of Fig. 3.16, using the transformation rules described in table 3.2.

This figure depicts the agents and objects of the scenario, respectively transformed from the Tropos agent and resource constructs in the actor diagram of Fig. 3.16. Besides the scenario's entities, the diagram also depicts the relations between them, converted from the dependencies, delegations and acquisitions depicted in the previously presented Tropos actor diagram.

Note that the relations are directed. The direction of the relations between agents has been directly inferred from the directions of the dependency, delegation and acquisition links on the actor diagram. Moreover, the number of relations between two agents is given by the number of dependencies, delegations and acquisitions between these agents. For instance, between PC Chair and PC Member, there are three relations, corresponding to the two acquisitions and one delegation previously depicted in Fig. 3.16, and following the same directions of such links. Regarding relations between agents and objects, the direction is always the same: the relation comes from the agents to the objects. This is due to the fact that agents are active entities, while objects are passive. Thus, agents usually 'use', 'send', 'receive' objects, i.e. in this way, the relation can be more naturally nominated using the active voice.

Although this first automatic AD is truthful to our scenario, some modifications may be necessary for enabling its best use in practice. This diagram can then be revised and modified, given rise to the AD of Figure 3.18.

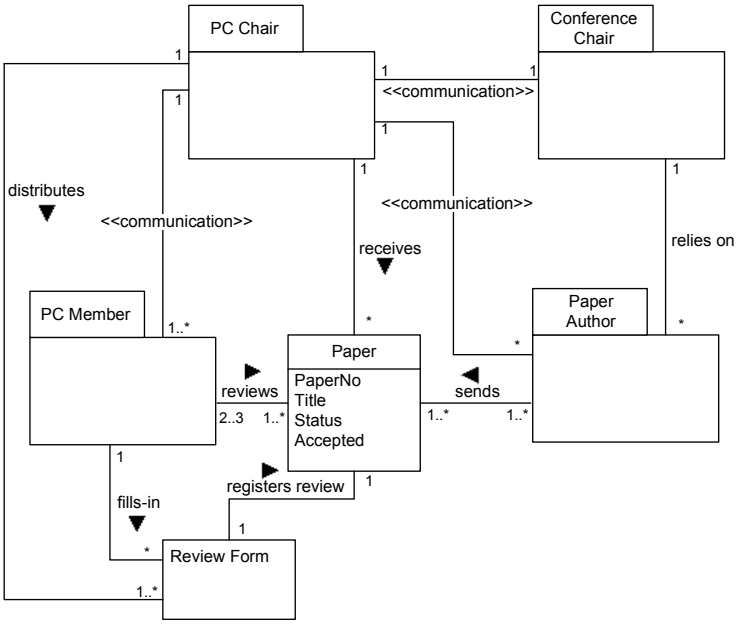


Figure 3.18: Final agent diagram

In the AD of Fig. 3.18, two objects from the previous AD, namely Sub-

mitted Paper and Assigned Paper have been merged into the Paper object. This comes from the realization that the previously depicted resources on the Tropos actor diagram actually referred to the same object, in two different states (i.e. 'submitted' and 'assigned'). Hence, the two objects have originated a single one, and such state is now given by the status attribute in the Paper object. Besides status, other three attributes now characterize the Paper, namely: PaperNo, which identifies the paper in the reviewing process; Title, i.e. the title of the paper; and the true/false Accepted attribute, which indicates whether the paper has been accepted for publication or not. In addition to that, multiple relations between agents have been reduced to one (as a result of a choice made by the designer. In other situations, multiple relations may be considered desirable, thus being maintained) and all relations have been named. Finally, we have created a specific type of relation between the agents, named *communication relation* (note the *communication* stereotype). Besides being related by associations, agents typically relate through communication relations, which indicate that they interact to accomplish their goals. Typically, communication relations will occur among agents that previously delegated goals or tasks, or acquired resources from one another. This is due to the fact that for a delegation or an acquisition to occur, agent A must explicitly interact with agent B, either to ask him/her to accomplish some goal or execute a task on his/her behalf, or to acquire a resource controlled by agent B. Details about such interaction are not presented in ADs, but rather in interaction diagrams.

Note also that the diagram of Fig. 3.18 presents the cardinalities of all agents and objects of the scenario. In the case depicted here, only association relations are necessary among the scenario's entities. In other cases, generalization and composition relations may be necessary (see, for instance, the case depicted in chapter 5). In general, all UML relations may be normally used in the AOR AD. This kind of diagram allows us to depict all entities of the scenario, presenting a comprehensive view of the domain being analyzed. The attribute of the other entities may be added as needed, describing the relevant properties of the scenario. When designing an information system, this type of diagram is particularly useful. Rather than

this one, which focuses on domain concepts, the AD is then used to depict entities that compose the system, both in terms of agents and objects. Some of these objects represent information entities that are later (on implementation time) converted into database tables or files to be read and written by the agents of the system (Wagner, 2003). This kind of use for the AOR AD is exemplified in chapter 6.

Proceeding with the analysis of the scenario, our next step is to specify the individual perspectives of the scenario's agents. For simplification, we here choose to exemplify this for only one agent, namely the PC Chair. Modeling the view of a particular agent is accomplished using the Tropos goal diagram. Hence, while Fig. 3.16 presents an overview of the scenario, Figure 3.19 focuses on the particular view of the PC Chair.

The goal diagram of Fig. 3.19 shows that the PC Chair has adopted the selecting proceedings papers goal previously delegated to him by the Conference Chair. At this point, we start refining this goal into sub-goals and analyzing which actual plans may be used to accomplish them. The selecting proceedings papers is here decomposed in two sub-goals that should both be accomplished (AND-decomposition) by the PC Chair, i.e. the having papers reviewed and deciding on paper's acceptance goals. Further analyzing the having papers reviewed goal, we note that the plan taking care of review is used to accomplish it. Next, this plan is decomposed in two sub-plans from which the PC Chair may choose one (OR-decomposition), namely the having paper reviewed by three PC Members and the participating in paper review plans. If the PC Chair decides do participate in the review of the paper, besides reviewing it (reviewing paper plan), he must send it to two other PC Members (having paper reviewed by two PC Members plan) Note that there are two softgoals that should be analyzed to understand why, at times, one choice is made over the other. The PC Chair wants, at the same time, to keep a fair load of work for him and the PC Members (sharing workload well softgoal) and to guarantee that the papers are fairly reviewed (being fair softgoal). On one hand, the sharing workload well contributes positively to the having paper reviewed by three PC Members, since in this way, the PC Chair is sharing his work with the PC Members. But on the other hand, the being fair softgoal

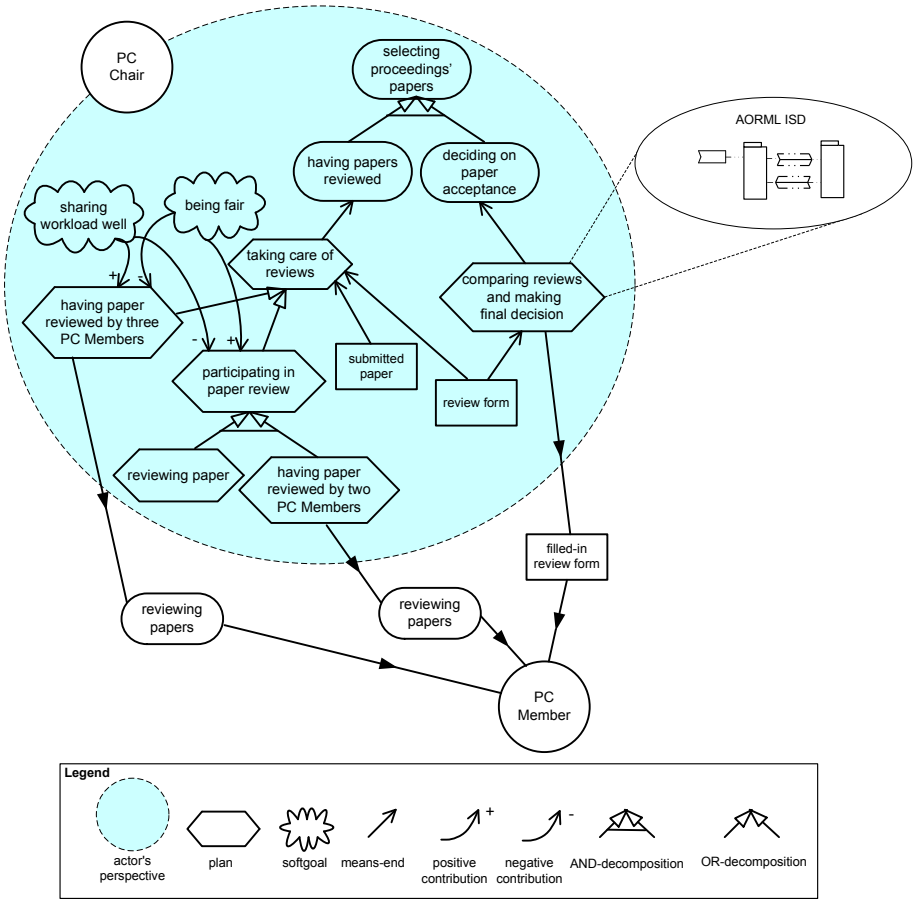


Figure 3.19: Tropos goal diagram specifying the point of view of the PC Chair agent

contributes negatively to this plan, since by loosing control of the review, the PC Chair is not a hundred-percent sure that the paper will be fairly reviewed (although by knowing the PC Members and their respective expertise may give an idea about this). In addition to analyzing goals and plans, the diagram also shows the resources used in the plans, i.e. the submitted paper and review form. Finally, the delegations to the PC Member are also included in this diagram. For executing the having paper reviewed by three PC Members and having paper reviewed by two PC Members plans, the PC Chair delegates to the PC Members the goal of actually reviewing the paper. And for performing the comparing reviews and making final decision plan, the PC Chair must acquire from the PC Member the review form, filled in with the data from the paper review.

The diagram of Fig. 3.19 illustrates several analysis methods of Tropos, namely *AND/OR-decompositions, means-end analysis* and *contribution analysis*. By examining it careful, we are able to present a few important guidelines. Decompositions, for example, are just allowed between entities of the same type. For instance, as illustrated in the diagram, one goal may be decomposed into two goals, or one plan may be sub-divided into two plans. Between entities of different kinds, means-end and contribution relations typically hold. However, we should note that semantically, these two relations are different. Means-end signify that an entity is actually used to achieve another. This is typically depicted between plans and goals (i.e. plans are means to achieve goals) and between resource and plans or resources and goals (i.e. resources are used in plan execution or in goal accomplishment). Contribution between a plan and a goal, or between goals are also possible. However, consider a plan contributing to a goal. In this case, a plan is not considered a complete strategy for accomplishing the goal, as in the means-end relation. Conversely, this plan partially accomplishes it, i.e. provides some sort of contribution for that goal achievement. In general, in a means-end relation, the plan has been tailored specifically or has the main purpose of achieving that goal. This is not the case for contributions, where plans that have other purposes may unadvisedly participate in goal achievement.

Another common use of contribution is on the analysis of alternatives, as

exemplified by the two softgoals presented in the previous actor diagram. In this case, goals may also be used instead of softgoals. For instance, a new goal named 'being on time' could be included, showing that it contributes positively for the PC Chair participation in paper review, while providing negative contribution to completely delegating it. In this case, the entity is a goal, and not a softgoal because it has a clear-cut satisfaction criteria. In other words, if the paper is reviewed before the review deadline has been expired, this goal is satisfied. And otherwise, it is not. The previous two goals (i.e. sharing workload well and being fair) are very subjective, thus the assessment regarding its satisfaction depends on the peculiar view of the PC Chair. For this reason, they are softgoals.

Much controversy has surrounded the application of Tropos softgoals. In ARKnowD, there are three possibilities for its use: a) adding a quality to a goal; b) representing a goal for which a particular domain agent does not have an objective criteria of assessment; or c) expressing a non-functional requirement of a system agent. The use expressed by a) refers to an add-on to a goal. For instance, a 'fairly' softgoal could be attached to the deciding on paper acceptance goal of Fig. 3.19. This qualifies that particular goal, reminding us that the PC Chair has this in mind while aiming at that goal. As an alternative, we could embed the adverb in the previously defined goal, generating the fairly deciding on paper acceptance softgoal. These are two ways of saying the same thing. Softgoals may also resemble a goal, not containing any specific adjective or adverb in its description. However, as indicated in b), if there is no objective criteria for assessing its satisfaction, it should still be represented as a softgoal. Finally, as reminded in c), softgoals are typically used for representing non-functional requirements of a system, such as 'security', 'high performance', and the like. In general, softgoals are constructs more present in the initial analysis of the scenario and system requirements. Their natural tendency is disappearing, or at least being related to goals that accomplish them. Even softgoals representing non-functional requirements shall have objective counterparts that actually realize such feature in practice. For example, as soon as a security mechanism is specified for a system, a 'security' softgoal may be achieved by a goal representing

this mechanism (means-end relation).

In Tropos, there is an important semantic distinction between resource acquisition, goal delegation and plan delegation, each one having its own particular application (analogously, this difference also holds for goal, plan and resource dependency). A resource acquisition characterizes a situation in which an agent A needs a specific resource owned or controlled by an agent B in order to accomplish a goal or execute a plan. This is illustrated by the acquisition link depicted in Fig. 3.19 between the PC Chair and the PC Member, representing that the former needed to obtain the filled in review form from the latter. As indicated in the diagram, this resource is specifically needed to enable the PC Chair to execute the comparing reviews and making final decision plan. Besides this resource acquisition link, this diagram exemplifies the use of goal delegations. This type of delegation refer to cases in which the accomplishment of one of the goals of agent A is conditioned to the accomplishment of a goal of agent B. In this case, agent A does not care about how such goal of agent B is achieved, leaving this decision to agent B. Conversely, the situation in which agent A wants agent B to follow a specific procedure characterizes a plan delegation. In other words, in plan delegation, agent A (the depender) specifies how agent B (the dependee) should act. As mentioned in section 3.5, goal and plan delegations refer to what (Castelfranchi and Falcone, 1998) respectively calls *open* and *close* delegation.

The previous goal diagram also includes a reminder that each of the plans can then be specified in more details using an AOR Interaction Sequence Diagram (ISD) (the other two types of AOR interaction diagrams may also be used to clarify some specific issues, as exemplified later in this chapter). The choice for which of the plans to detail and when is the responsibility of the analyst/designer. In any case, there is an important observation to be made here. In Tropos, plans could be indefinitely refined. For instance, the having paper reviewed by two PC Members plan could be refined into three sub-plans, such as: choosing the two best PC Members to review the paper, submitting the paper to the PC Members, reinforcing the review deadline. However, in AR-KnowD, we advise the analyst to keep the granularity of the plans in a level

where it can be then specified using AORML. This guideline is motivated by our realization that the AOR interaction diagrams are more appropriate than the Tropos goal diagram for modeling interactions. For instance, the ISD models agent's actions and communications events, besides also including non-action (or environment's) events, and commitments between agents. As an illustration, Figure 3.20 depicts an ISD that serve both to the having paper reviewed by two PC Members and having paper reviewed by three PC Members plans.

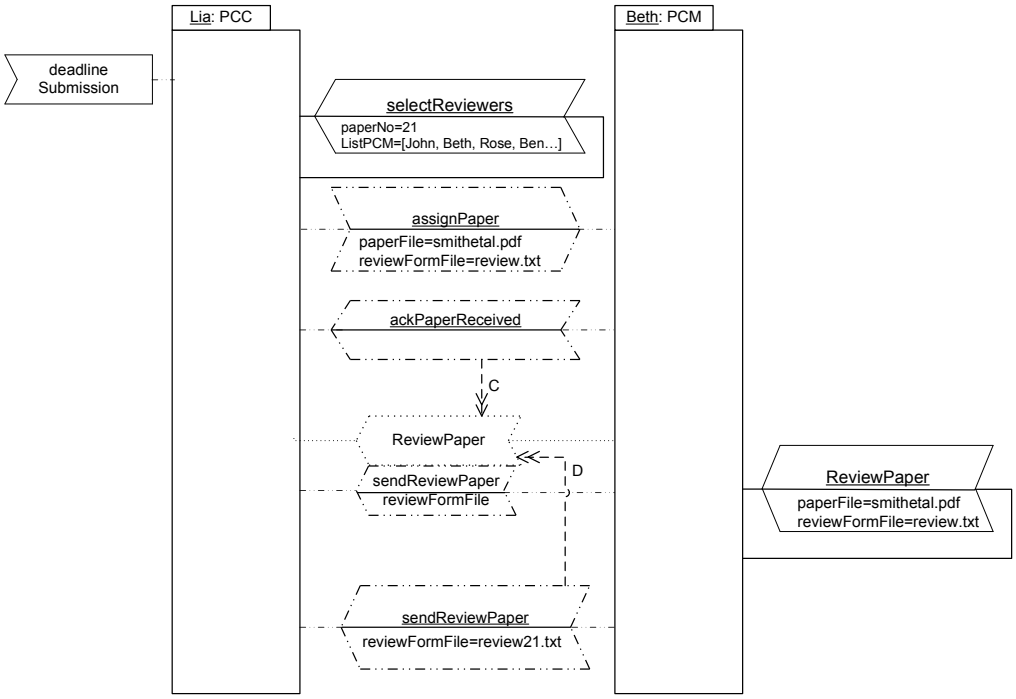


Figure 3.20: AOR Interaction Sequence Diagram

In the diagram of Fig. 3.20, the agent's interactions are triggered by a non-action event (i.e. part of the environment, independent of the agents). More specifically, the deadline for paper submission has come (dealineSubmission event). When Lia, the PC Chair (PCC) senses this, she starts distributing the papers to the PC Members. For reasons of space and simplicity, only one PC Member is depicted in this diagram. However, the interaction with other PC Members is analogous to the one exemplified here. Lia's first action is selecting the right reviewers for the paper, based on the area

targeted by the paper and on what she knows about the expertise of the reviewers (note that the SelectingReviewers action receives two parameters, the paper, identified by its number, and the list of available reviewers). Having identified that Beth is a good PC Member (PCM) to review the paper number '21', Lia submits the paper file, along with the review form to Beth (assignPaper message with paperFile and reviewFormFile parameters). Next, Beth acknowledges that she has received the message. At this point, Beth is committing to review the paper assigned to her (ReviewPaper commitment). Note the line coming from the acknowledgment message to the commitment, annotated with a "C", indicating the commitment has been "created" by that message. For Beth, a commitment refers to a specific action to be performed in due time, i.e. reviewing the paper. For Lia, this construct actually represents a claim regarding a specific action that should happen in the future. Note that the ReviewPaper commitment has a message attached to it (i.e. a sendReviewPaper message), indicating that this commitment is fulfilled if Beth submits a message of this kind to Lia. Otherwise, this commitment is broken, giving Lia the right to sanction. Fortunately, in this case, Beth has fulfilled her responsibility, sending back the review form, filled in with the paper review (sendReviewPaper message sent by Beth to Lia). Note the line coming from the message to the commitment, annotated with a "D", which stands for "discharge" (i.e. the message discharges the commitment). The commitment between Lia and Beth reflects the delegation between PC Chair and PC Member, depicted in the actor diagram of Fig. 3.16. As indicated in the transformation rules of table 3.2, Tropos delegations lead to the establishment of AOR commitments.

Although the AORML notation differs from UML, it does make use of various UML constructs. In fact, whenever AORML does not provide extensions, UML can be normally applied. Note for example that like UML, instances names are underlined (see the agents' and messages' names) while class names are not (e.g. PCC and PCM, respectively representing the PC Chair and PC Member agent classes). Note also that the name of the agent instances are followed by the name of their classes (e.g. 'Lia: PCC'). Differently from UML, though, ISDs show interactions among agents. Thus, the

semantics of the message construct is different. In object-oriented programming (and thus, in UML), a message typically refers to a method call. As previously mentioned in section 2.4, agents have control over their behavior, being requested to perform actions. Consequently, agents communicate using *speech acts*, which specify a sender, a receiver, a illocutionary act or performative (such as 'request', 'inform' and so on), and a message content (Labrou et al., 1999). For example, the assignPaper message represents a request, in which Lia is the sender and Beth is a receiver, and the content is given by the parameters of the message.

The greatest innovation in this diagram is given by the possibility of defining and controlling commitments established between two agents. According to the AORML author Wagner (2003, 11), *"commitments and claims are fundamental components of social interaction processes. Consequently, a proper representation and handling of commitments and claims is vital for automating business processes."* And indeed, other researchers on agent organizations have acknowledged this concern, if not directly representing commitments/claims, relying on the related deontic concepts of obligations, rights and responsibilities (Dignum, 2004a) (Esteva et al., 2002) (Hubner et al., 2002). In the case modeled in Fig. 3.20, the commitment serves to regulate the relationship between two human agents. However, commitments established between artificial agents are also highly useful. They may indicate for the information system designer an important point for exception handling. In addition to that, such constructs are especially indicated for cases in which a commitment exists between two agents developed by different parties, as in open systems developed via Internet, for traditional or virtual organizations. In this case, the services provided by external vendors can be regulated by contracts established by the commitments between agents. For a case in which AORML supports the design of a KM system using agents external to the organization, please refer to (Santos et al., 2005a) and (Santos et al., 2005b).

The delegation and acquisition links earlier depicted in Tropos give rise to commitments when the system is designed in further details. In general, in the early stages of the analysis activity, only a flavor of the relations exist-

ing among agents is captured, leaving the definition of commitments/claims for later design activities. Then, the commitments/claims are completely modeled, along with the actions that may fulfill them, and the applied sanctions in case they are not fulfilled. The choice for different level of details, supported by the diversity of concepts (in analysis, delegation and acquisition; in design, commitment/claim) provides the right level of abstraction for each activity. On one hand, in the analysis, details are overlooked and the analyst may focus on the big picture. On the other hand, during design, the designer is able to capture all details that lead to system automation.

As it becomes apparent from Fig. 3.20, an AOR ISD models (some part of) a prototypical instance of an interaction process. An interaction process is a sequence of action and non-action events, performed and perceived by agents. A protocol defines a particular sequence of action and non-action events. For example, deadlineSubmission is a non-action event, i.e. an event generated in the environment and perceived by the agent, Lia in this case. Both selectReviewers and assignPaper are action events, the latter being a communicative action event. To understand better the relations between agents and events events (including action and non-action events), one may refer back to the ontological distinctions discussed in section 3.5.

ISDs are generally made for several prototypical situations, to give the analyst/designer a clear idea of the possible outcomes of the agent's interactions. So, an alternative diagram to the one previously shown could be created, showing a case in which the PC Member fails to deliver the review of the paper, breaking the commitment established between him and the PC Chair. The alternative diagram could hence model the consequence of this break of contract, both to the PC Chair and the PC Member. In that case, the PC Chair could send a reminding message, as we will model in a moment, using another kind of AOR diagram (Figure 3.21). And after all, if the PC Member still fails to send the review, the PC Chair would typically have to review the paper himself. At the same time, the PC Member could receive a complaint message, or he could be annotated by the PC Chair as someone not to invite for program committees of future conferences.

The ISD specifically deals with agent's external actions and interactions,

but it does not concern how the agents behave internally. This may be important in different points of the analysis or design activity. AORML provides a special diagram to model internal behavior of agents, typically triggered by action or non-action events. This diagram is named Interaction Pattern Diagram (IPD), illustrated in Figure 3.21.

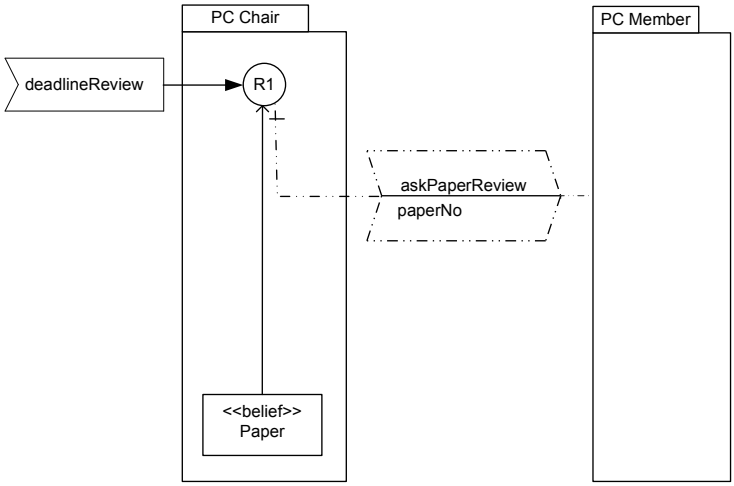


Figure 3.21: AOR Interaction Pattern Diagram

The diagram illustrates the PC Chair's behavior when the deadline for reviewing papers is achieved. The deadlineReview event triggers the R1 rule, representing the PC Chair's reactive behavior. This rule regards the verification if the papers have been revised or not, and may be written as shown in table 3.3. In case the paper has not yet been reviewed (indicated by the crossed line coming from the rule R1, the PC Chair submits a message to the PC Member that missed to send the review for the given paper (askPaperReview message).

As can be noted in the diagram of fig. 3.21, a reaction rule is visualized as a circle with incoming and outgoing arrows drawn within the agent rectangle whose reaction pattern is represented (the PC Chair, in this case). Each reaction rule has exactly one incoming arrow with a solid arrowhead, specifying the triggering event type. In our case, the agent's reaction is triggered by the deadlineReview event. Other ordinary incoming arrows representing state conditions (referring to corresponding instances of other entity types),

ON	Event	Perceive deadlineReview
IF	Condition	IsReviewed(?PaperNo) <> TRUE
THEN	Action	SEND askPaperReview(?PaperNo) TO ?PC Member

Table 3.3: Textual description of the rule R1 representing the PC Chair's reactive behavior

as the arrow coming from the Paper belief object class. There are two kinds of outgoing arrows: one indicating the performance of (either physical or communicative) actions and one for specifying mental effects (changing beliefs and/or commitment) resulting from the execution of the rule. In our example, only the former is illustrated, with the connector to the askPaperReview message. The latter (an arrow with a double arrowheaded) is exemplified at (Wagner, 2005).

Besides ISDs and IPDs, AOR still offers a third possibility with the Interaction Frame Diagrams (IFDs). An AOR IFD gives a static view of the possible interactions between two (types of) agents without modeling any specific process instance. It consists of various types of communicative action events, non-communicative action events, commitments/claims (coupled with the corresponding types of action events), and non-action events (Wagner, 2003). Figure 3.22 presents such kind of diagram, depicting all interaction possibilities between the PC Chair and the PC Member.

The diagram of Fig. 3.22 presents all messages exchanged by the PC Chair and PC Member in the two previous diagrams, besides the commitment established between them in the ISD of Fig. 3.20. This summary presents an overview of their interaction and is typically interesting between two artificial agents, because it clearly indicates the interface between them, facilitating coding.

3.9 Automated Support

One of the main advantages of using existing work on agent-oriented approaches comes from profiting from the already available modeling tools

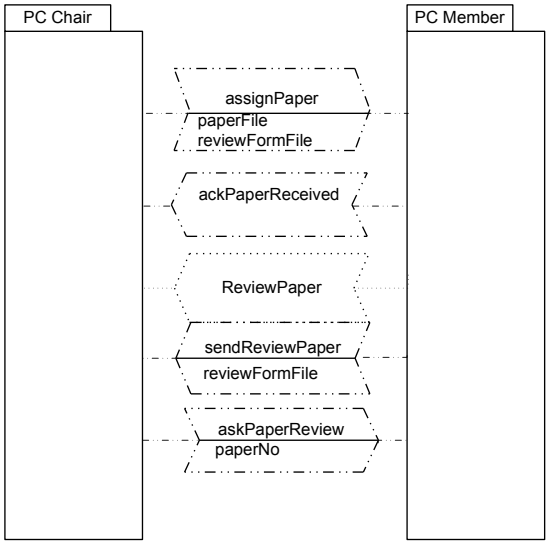


Figure 3.22: AOR Interaction Frame Diagram

developed to support such approaches. In this respect, there are several tools available to support Tropos modeling such as:

1. GR Tool [3]: provides particular support to goal analysis. In this tool, the goals of an agent are designed, and values that refer to the satisfaction or denial of the goals are established. Given these values, the tool supports both qualitative and quantitative relationships between goals, and can be used to perform two types of analysis. The first type (forward reasoning) answers questions of the form: Given a goal model, and assuming that certain leaf goals are fulfilled, are all root goals fulfilled as well? The second type of analysis (backward reasoning) solves problems of the form: Given a goal model, find a set of leaf goals that together fulfill all root goals.

2. T-tool [4]: allows the analyst to check the consistency of his/her models, by applying a formal specification language named Formal Tropos. This language supports the primitive Tropos concepts, but offers in addition, a rich temporal specification language inspired by KAOS

[3] http://sesa.dit.unitn.it/goaleditor/
[4] http://www.dit.unitn.it/~ft/ft_tool.html

(van Lamsweerde et al., 1991). Besides verifying model's consistency, T-Tool allows checking whether it respects a number of desired properties. Moreover, a specification can be animated in order to give the user immediate feedback on its implications.

3. TAOM4E [5]: targets a comprehensive agent-oriented modeling environment, supporting the analysis and design of agent-oriented systems. Its development has taken MDA recommendations into account. At the present development stage, TAOM4E mainly supports the Tropos modeling language. However, there have been some parallel initiatives of integrating AORML and AUML, allowing the analyst and designer to profit from transformations between Tropos and one of these two UML-based modeling languages. The integration of new languages is facilitated by the choice of developing TAOM4E as an Eclipse plug-in. Eclipse [6] is an open source initiative that allows the integration of different tools into a single application.

At the moment, design using AORML is made possible by using a Microsoft Visio Template [7]. However, in the future, it is desirable to have a tool dedicated to AORML, or one that includes it as a possible modeling language. To this end, we have launched an initiative to integrate AORML in TAOM4E. In this work, we apply the MDA transformation method described in section 3.7.1. More specifically, we have developed the transformation between a Tropos actor diagram into an AOR Agent Diagram, following the transformation rules depicted in table 3.2.

This work has been allowed by the use of Tefkat [8]. Tefkat is a prototype transformation engine developed by the Distributed Technology Centre (DSTC) of the National IT Research and Development Center in Australia. As TAOM4E, Tefkat has been developed as an Eclipse plug-in, which facilitates its integration with this given agent-oriented modeling environment. Figure 3.23 illustrates this integration.

[5] http://sra.itc.it/tools/taom4e/
[6] http://eclipse.org/
[7] available at http://www.informatik.tu-cottbus.de/g̃wagner/AORML/
[8] http://www.dstc.edu.au/Research/Projects/Pegamento/tefkat/

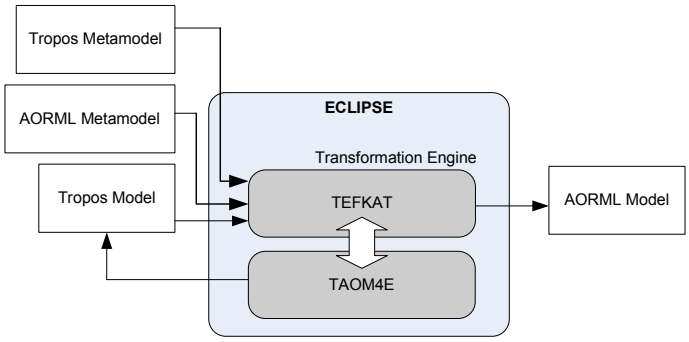


Figure 3.23: Transformation engine

Fig. 3.23 shows that for executing a transformation, Tefkat receives as input the metamodels of the two modeling languages (i.e. the metamodels of the Tropos language and AORML), along with the source Tropos model developed with the use of TAOM4E. The mapping between the two metamodels is directly implemented using Tefkat's declarative language. The result is an AORML model, as indicated in Fig. 3.23. The integration of Tefkat and TAOM4E is facilitated by their both being implemented on top of Eclipse, thus being compatible with the Eclipse Modeling Framework (EMF). Listing 3.1 gives an idea of the declarative language applied in Tefkat.

Following, we exemplify the transformation of the Tropos actor diagram of Fig. 3.16 into the AOR agent diagram of Fig. 3.17, using the implemented transformation plug-in. Figure 3.24 shows the given actor model, designed with the support of TAOM4E[9].

Fig. 3.24 shows that, for executing a transformation, you right click with the mouse on the Tropos diagram file and select *Tefkat*. At this point, the transformation is automatically executed, generating the XMI file depicted in Figure 3.25.

[9]As TAOM4E does not support the differentiation between delegation, dependency and acquisition adopted in ARKnowD, we here designed the diagram using the existing dependency construct. However, this does not result in any problem regarding our exemplification, as for the three types of relations, an association link is included in the AOR AD.

Listing 3.1: Tefkat's declarative language

```
TRANSFORMATION LinkingTropos2AORML: tropos -> aorml

IMPORT http://aorml.ecore
IMPORT http://taom4e/model/informalcore.ecore

CLASS ActorForAgent{
  FActor actor;
  Agent agent;
};

CLASS ResourceForObject{
  FResource res;
  Object obj;
};

CLASS DependencyForAssociation{
  FDependency dep;
  DirectedAssociation dass;
};

RULE Actor2Agent()
FORALL Actor at
MAKE Agent ag
SET ag.name = at.name
LINKING ActorForAgent WITH actor=at, agent=ag
;
```

By analyzing Fig. 3.25, we note that such XMI file corresponds to the AD of Fig. 3.17. We note that there are three agents corresponding to the agents of the previously modeled actor diagram. Besides, the previously depicted resources have been converted into three objects, presented in this file. And finally, all dependencies between agents in the Tropos actor diagram have been converted into directed associations between agents, or between an

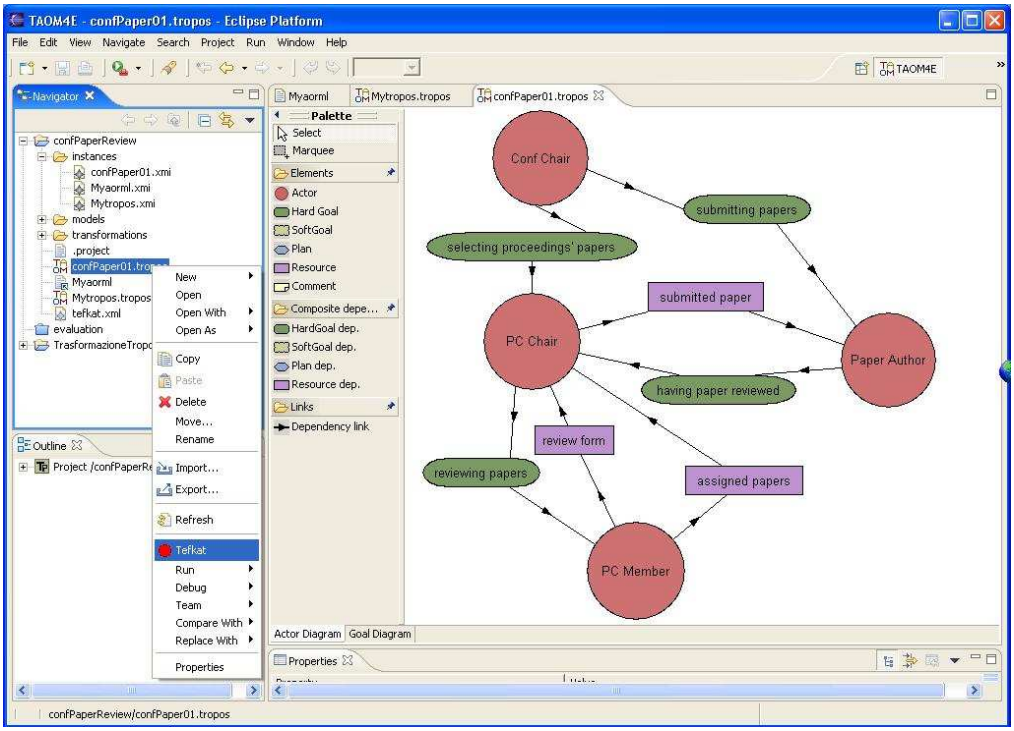


Figure 3.24: Actor diagram designed in TAOM4E

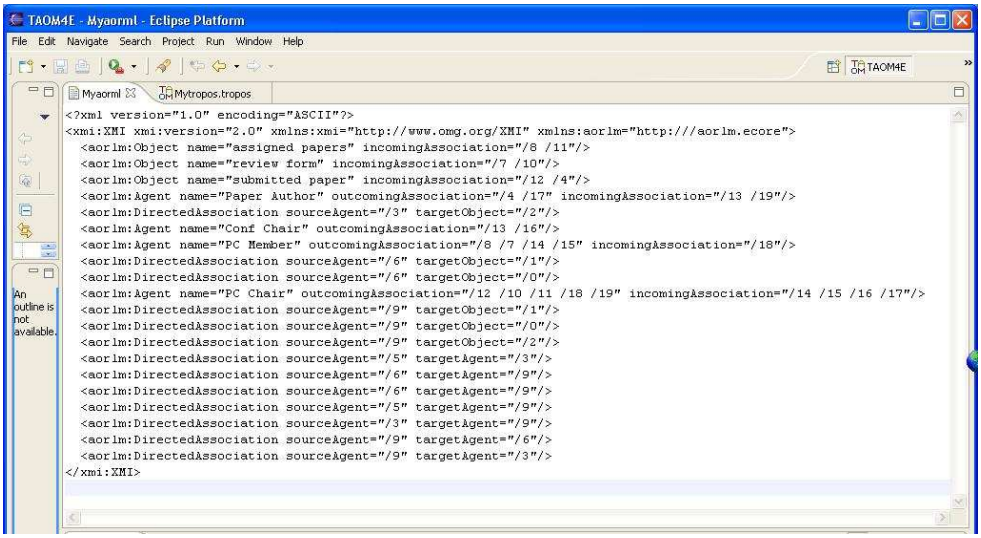


Figure 3.25: Transformation output file

agent and an object. The visualization of the graphical AD resulting from this file has not yet been developed in TAOM4E and remains future work.

3.10 Related Work

ARKnowD has emerged from the combination of two previous work directed towards the proposal of an agent-oriented methodology supporting KM, i.e. those of Perini et al. (2004) and Guizzardi et al. (2004a). The former supports the use of agents to model organizational processes, proposing a methodology for analyzing KM requirements based on intentional analysis, claiming that, in order to develop effective KM solutions, it is necessary to analyze the intentional dimension of the organizational setting, i.e. the interests, intents, and strategic relationships among the agents of the organization. Their methodology is based on the use of the i* framework (Yu, 1995), the same used as a basis for the development of the Tropos methodology. In our approach agent-oriented modeling is proposed as one of the techniques supporting a more complex analysis process that leads to a design activity, not targeted in their initiative. Moreover, having adopted the Tropos methodology allows using a more clear agent-oriented semantics of i* elements, which enables a smooth transition to our design approach. In (Guizzardi et al., 2004a), we have proposed the use of AORML for KM analysis and design, in the context of collaborative learning . Although we acknowledge the possibility of using AORML in domain modeling, we feel that this language lacks the concepts and constructs to support requirements analysis. As motivated in chapter 2, supporting KM depends on a clear understanding of the scenario, where requirements analysis comes as an essential step. In general, modeling with AORML starts with information modeling (like in UML class diagrams), jumping over the requirements analysis step. A common proposal for these initial activity is the use of UML Use Cases, however, we claim that our approach is more appropriate for focusing on goals, supported by the emphasis given by Nonaka and Takeuchi (1995) on intention (i.e. goals) as the basis of any KM project.

Dignum (2004a) describes two case studies of the application of OperA to support KM scenarios. In this same work, the author claims that the needs of such scenarios has been the main motivator for OperA's proposal. This stems from the recognition that, like multi-agent systems, KM environments can be seen as distributed systems where different actors, each pursuing its own goals, need to interact in order to achieve common targets and realize organizational objectives. Agents are considered appropriate for being endowed with social skills that comply with the needs to model the complex interaction and negotiation processes that characterize such environments. Moreover, agents are able to proactively change to cope with changes in this highly unpredictable business setting. In our work, we share these opinions, besides the use of similar concepts, such as agents, roles and goals. However, the modeling constructs applied are completely diverse, for instance, while OperA makes uses of scene scripts and provides a sound formal foundation based on temporal deontic logic, our proposal is much less formal, aiming at the support of the specification of KM environments through the use of a graphical language.

Related work may also be found in (Loucopoulos and Kavakli, 1999), where the authors propose a conceptual modeling approach to support enterprise KM. This work shares many similarities to ours. It also proposes the analysis of the goals of the stakeholders, allowing the establishment of dependencies and support relationships, which are similar to what Tropos refers to as contribution. It also assigns processes as goal operationalization, as in ARKnowD (starting by the definition of plans to fulfill goals, and then its description with AOR interaction diagrams). However, besides goal, resource and activity dependency (this last one being analogous to Tropos plan dependency), their approach models different kinds of dependency, such as authority dependency and coordination dependency. Another divergence is that for them, the process of acquiring and maintaining knowledge refers to the structure and processes underlying the targeted organization. The authors focus on eliciting and representing this knowledge in a sort of business process analysis. Rather than a KM systems, the result of this analysis is the proposal of an information system to automate the organization's processes.

Finally, the CommonKADS methodology has been recently proposed as a solution for KM settings (Schreiber et al., 2000). This methodology has an agent-oriented counterpart, namely the MAS-CommonKADS, earlier described in section 2.4.2. The CommonKADS approach consists of a requirements specification activity, focused on the construction of five models: *organization, task, agent, knowledge* and *communication models*, each one analyzing different aspects of the system-to-be. This is followed by a design activity, which defines the technical system specification in terms of architecture, implementation platform, software modules, representational constructs and computational mechanisms needed to implement the functions specified in the knowledge and communication models. The main differentiation between our approach and CommonKADS is that this methodology has been specifically built to develop knowledge-based systems, as the expert systems previously described in section 2.2.2. As a result of this knowledge engineering orientation, this approach places great strength on the business process (task and communication models) and knowledge artifacts models (knowledge model), giving less attention to the agents executing such processes and producing such artifacts. Agents are merely focused in the agent model, in which scripts are built indicating the knowledge, processes, responsibilities and constraints of each agent.

3.11 Conclusion

This chapter has described the ARKnowD methodology, along with its notation, activities and scenarios of applicability. For the main purpose of supporting KM, ARKnowD combines two distinct agent-oriented software engineering approaches, namely Tropos and AORML. For merging the two notations, we proposed a MDA-inspired transformation method, partially implemented in an agent-oriented CASE tool named TAOM4E, currently under development. In this chapter, we also focused on clarifying the semantics of the applied agent-oriented concepts, formalizing them with the means of an ontology. This same ontology is used for evaluating and assisting a consistent merge of the adopted notation. Furthermore, we provided

a few modeling guidelines, supported by a working example that illustrates the application of ARKnowD.

One of the main principles of ARKnowD is the realization that there is no silver bullet when pursuing an agent-oriented engineering methodology, so the best approach is combining existing work according to the given domain or situation. This tendency for comparing and combining existing methodologies has been already noted in the agent community (Dam and Winikoff, 2003) (Bernon et al., 2004) (Juan et al., 2003) (Juan et al., 2004) (Sabas et al., 2002), expressing that work in this area has matured in the past few years. Specifically for KM scenarios, we have found the combination of Tropos and AORML quite appropriate, for the reasons previously stated. The combination of these and other approaches for the application in different domains is an interesting work that should shed new light in this direction.

One important claim of our work is that more focus should be given to the initial phases of system development, aiming at grasping the requirements for an appropriate solution, both in terms of the individual perspective of the organizational members and the overall objectives of the organization. This is especially important in the KM context, which focuses on the effective use of human intellectual capital, since much of human knowledge is tacit and intangible (Nonaka and Takeuchi, 1995). Moreover, issues such as community and community's practices (Wenger, 1998) go much beyond those typically considered in the conception of traditional systems, and opens up many more ways to leverage information technologies to augment human and organizational capabilities and performances.

The methodology proposes the analysis of the goals of the system's stakeholders and their inter-dependencies as the initial steps towards understanding the requirements for a KM system. The main strengths of this approach can be summarized as anticipating the concerns of all participants of a given scenario, focusing on the stakeholders' aims while abstracting from unimportant issues, until the domain is well understood and the analyst is ready to propose a solution (either by changing organizational structure, current processes, or by applying technology). Nevertheless, a consistent methodology should be provided, going beyond the analysis and moving towards the

design of this solution. In this sense, ARKnowD allows agent's cognitive concepts such as beliefs and commitments to be designed and later materialized in practical elements of a system. Here, we particularly rely on a clear model of the entities of the scenario and their relations, and the detailed description of their interaction and behavior. The adopted notation allows the combination of agents and objects in a single model, which is particularly suitable for KM settings. In this respect, knowledge artifacts are represented as objects, and the stakeholders as human agents. If an information system is proposed, the system itself can also be composed of multiple agents, which manipulate different objects in order to mediate the processes of knowledge creation, integration and sharing. In the end, objects that represent both elements of the scenario and agent's beliefs turn into information entities materialized by database tables or XML files, for example. Meanwhile, the modeled agents are turned into the actual code of the given system.

More can be understood about the use of ARKnowD in the following chapters. Chapter 4 illustrates the application of the requirements analysis methodology, culminating with the proposal of a recommender system named KARe. This system is further designed in chapters 5 and 6, the former focusing on a platform independent design while the latter provides a specification of this design for the JADE framework.

Chapter 4

Domain and System Analysis

"Try to put well in practice what you already know. In so doing, you will, in good time, discover the hidden things you now inquire about."
Rembrandt

This chapter presents the analysis of a fictitious scenario specifically tailored based on available KM literature, to serve as a case study in this thesis. Here, we present the analysis of the scenario's organization, eliciting the requirements for KM support, especially focusing on the development of a supporting Information System. The main objective of this analysis is to show the usability of Tropos for the early stages of KM Systems development, as proposed by ARKnowD.

The chapter is organized as follows: section 4.1 introduces the chapter; section 4.2 presents the scenario used as our case study; section 4.3 starts the analysis, presenting the first Tropos's diagrams made with basis on the case study; from section 4.4 to section 4.8 develops the remaining of our analysis, presenting in each section, a model made with basis on a relevant aspect of our case study, taking the perspective of the different agents of the scenario; section 4.9 provides some reflections on how the analyzed organizational setting supports Constructivist KM; section 4.10 discusses the requirements for a system to support Constructivist KM in the scenario, and finally, section 4.11 presents the conclusions of this chapter.

4.1 Introduction

Knowledge Management scenarios are highly influenced by their human dimension, involving intricate relationships among different agents, along with their personal motivations and abilities, and guided by norms and behaviors that are part of the organizational culture. However, most of current KM systems are developed following a purely techno-centric view (Pumareja et al., 2003) (Bonifacio and Bouquet, 2002), focusing on the functionality of the system under development rather than on the real needs and wishes of the stakeholders. Such approach is bound to fail, as it does not consider the particularities of the environment in which the system is to be used.

As a solution to the mentioned problem, we claim that more focus should be given to the initial phases of system development, aiming at grasping the requirements of the system to be, by deeply understanding the structure of the targeted organization. This analysis should consider both the overall objectives of the organization and the individual perspective of organization's members. This allows the creation of a rich conceptual framework, allowing the analyst to make a clear connection between the functional and non-functional requirements of the system-to-be, according to relevant stakeholders and their intentions (Giorgini et al., 2005).

Currently, a methodology for KM requirements analysis in the available literature of both agent-oriented software engineering and organizational science does not exist. On the agent-oriented software engineering side, most of available methodologies for system requirements analysis are based on a clear statement of requirements. In other words, they often apply methods to capture requirements, such as use cases, but do not provide a means to elicit or negotiate the requirements among the stakeholders. As argued in section 2.6, in a complex KM scenario, it is necessary to analyze the current situation, trying to grasp which processes should be changed to accommodate new KM practices and systems, besides eliciting the requirements of the system itself. This is especially true in case we aim at supporting Constructivist KM, in which organizational members of different functions and hierarchical levels should be consulted about their own goals and require-

ments. On the other hand, initiatives from the area of organizational science are usually based on interviews and observations of agents of the different organizational units, gathering their different points of views, critics and suggestions. However, such studies are rarely supported by a modeling language to allow reasoning and communication about the problem domain. Consequently, the analysis conclusions are highly subjective and based on the researcher's personal experience.

The methodology exemplified here allows different analysis methods, such as understanding trust, commitments and vulnerability in the relationships between agents, and grasping the 'hows' and 'whys' of a particular choice (Bresciani et al., 2004). For this purpose, the methodology proposes the analysis of the goals of the system's stakeholders and their inter-dependencies as the initial steps towards understanding the requirements of a KM system. The main strengths of this approach can be summarized as anticipating the concerns of all agents involved in a given scenario, focusing on the stakeholders' aims while abstracting from unimportant issues, until the domain is well understood and the analyst is ready to propose a solution (either by changing current organization's structure and processes, or by applying technology). In addition to that, the adopted notation is visually rich and accessible, besides being supported by existing modeling tools (Perini and Susi, 2004).

The models described in this chapter are the result of an iterative process and have been refined after several analysis cycles, as proposed in section 3.3. Different stages of this analysis have been ealier presented in (Guizzardi et al., 2003) (Guizzardi et al., 2004b) (Guizzardi and Perini, 2005).

4.2 Knowledge Management in CoPs: a Fictitious Scenario

In order to demonstrate our proposed methodology, we use here a fictitious scenario. Although not a real case study, this scenario has been carefully tailored to be realistic, taking into consideration the available literature

(Dignum and van Eeden, 2003) (Gongla and Rizzuto, 2001) (Wenger, 1998) (Orlikowski, 1992a) (Pumareja et al., 2003).

Luca starts working in BHI Software Company. He is a programmer with 10 years of experience. As a newcomer at BHI, he needs to adjust to the organization's work practices. This involves integrating to the project on which he is going to work, and adapting to the work style of his working team. Furthermore, it also includes learning about the company's policies and management directives.

Aiming at providing its workers with a rich environment for knowledge sharing, BHI Management fosters the development of Communities of Practice (CoPs) across the organization. These communities are self-organizing groups whose members share interests and goals, or perform similar tasks within the organization. They are not necessarily from the same working team or division, and their members are dispersed across the 10 branches of BHI. The CoPs play an important role in allowing newcomers to get acquainted with their new working environment, naturally learning about products, projects, specific domains, and procedures.

Luca decides to join a CoP named 'OpenS', which focuses on understanding how open source software can be used to support the development of BHI products. Though the communities are self-organizing systems, a special sector within the company was created to support them: the Knowledge Management Division. One of the main objectives of this division is supporting the CoP on pursuing explicit targets related to the organization's goals. This allows the community members to feel important as a group for the organization at the same time that the CoP's value is more concretely measurable from the Management's point of view. In addition to that, this division also provides information about the active CoPs that are open for new membership, in order to facilitate the integration of newcomers in the organization.

4.3 The Domain Stakeholders

According to the *Tropos* methodology (Bresciani et al., 2004), domain analysis starts by identifying the main stakeholders, modeled as agents, with their goals. Figure 4.1 shows an initial model where the BHI company top management is modeled as the agent Management, depicted as a circle. The organization has an initial softgoal relative to having the organization's team working well [1], which expresses how BHI intends to achieve more general objectives such as pursuing high quality of the products and of the production processes (pursuing high quality products/processes goal), as well as innovation (innovating goal) by considering human resources as a main asset.

The BHI's Knowledge Management Division and the communities within the organization play critical roles with respect to BHI strategic goals, according to the scenario, so they are also modeled as specific agents, namely the KM Division agent and the CoP agent. Luca plays the role of a newcomer in the organization (Newcomer agent), with his main goal of adjusting to the working practices of the organization (adjusting to the organization practices goal).

In this initial model, only the main goals of the Management and the Newcomer agents are included, indicating that our analysis concentrates on the perspective of these two agents. Further modeling steps consist in analyzing each agent's goal from the point of view of the agent itself, aiming at identifying the strategic dependencies and delegations between agents, i.e. those which allow for goals achievement. Figure 4.2 shows basic goal delegations between the scenario's agents.

The analysis of the Management's softgoal team working well points out a strategic organizational goal, i.e. CoPs fostering, which is then delegated to the Knowledge Management Division (KM Division agent). In return, the KM Division relies on the organization's Management to be legitimized

[1] The reason for modeling team working well as a softgoal is the fact that the Management is not monitoring and measuring explicitly the team work quality. In the process of refining the goal analysis from the point of view of the organization's Management, the contribution of the team working well softgoal to the other goals of this agent can become more explicit

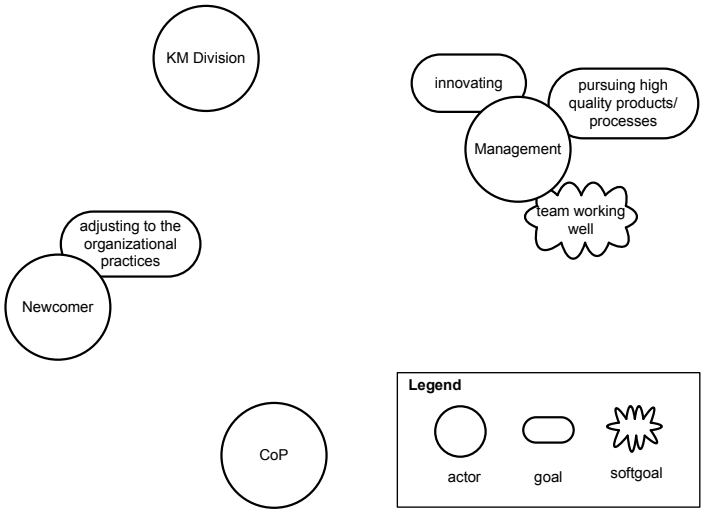


Figure 4.1: Initial domain model of the scenario

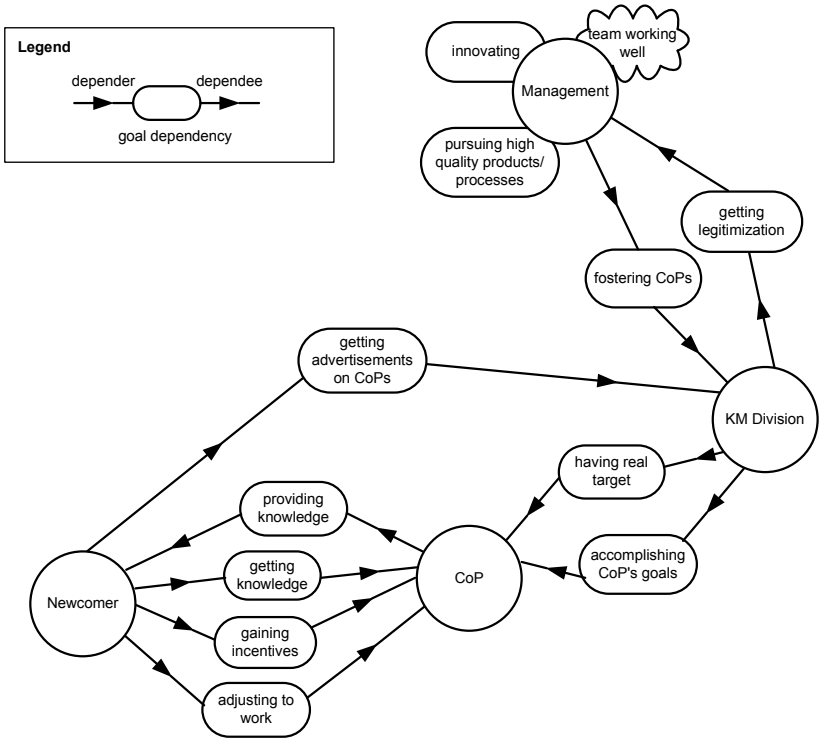


Figure 4.2: Main goal delegations between the agents of the scenario

for playing the specific role of motivating and supporting Knowledge Management practices (legitimization getting goal). The initial Management's softgoal, leading to its main goal of supporting CoPs, generates all other goal delegations between the remaining agents in the scenario.

Taking, for instance, the pair of agents Newcomer and CoP, we note that there are goal delegations in both directions. The Newcomer delegates to the CoP the goals of getting knowledge, gaining incentives, and adjusting to work. On the other hand, the CoP aims at profiting from the Newcomer's own knowledge and experience (getting knowledge goal coming from the CoP to the Newcomer). This mutual delegation characterizes what the i* framework names "sustainable relationship", i.e. a relationship in which two agents delegate to each other one or more of their own goals (this also applies to acquisition and dependency in both directions). Sustainable relationships indicate that there is some kind of balance between the two agents, thus helping them achieve individual goals. On the other hand, if there are dependencies, delegations or acquisitions only from one side, this indicates a vulnerability by this *depender* agent towards the *dependee* (Yu, 1995). Such unbalance should be corrected in order to guarantee that both agents are committed to each other. This is exactly the case between the KM Division and the CoP agents. Note that while the KM Division delegates two goals to the CoP (having real target and accomplishing CoP's goals), the CoP does not seem to depend on the KM Division for achieving any goal. This can result in a lack of motivation on the part of the CoP to target the goals delegated by the KM Division.

This kind of analysis could be crucial in a KM scenario. In our case, for instance, we realize that while demanding the CoPs involvement with organization's objectives, the KM Division does not provide, as a counterpart, any incentive to the CoP. In fact, the KM literature (Orlikowski, 1992a) indicates that an incentive policy is essential to motivate knowledge sharing. As a result of this preliminary analysis, the analyst may propose to the KM Division the adoption of a method for fostering CoPs, such as the *Seduce, Engage, Support (SES)* model (Dignum and van Eeden, 2003), captured in figure 4.3.

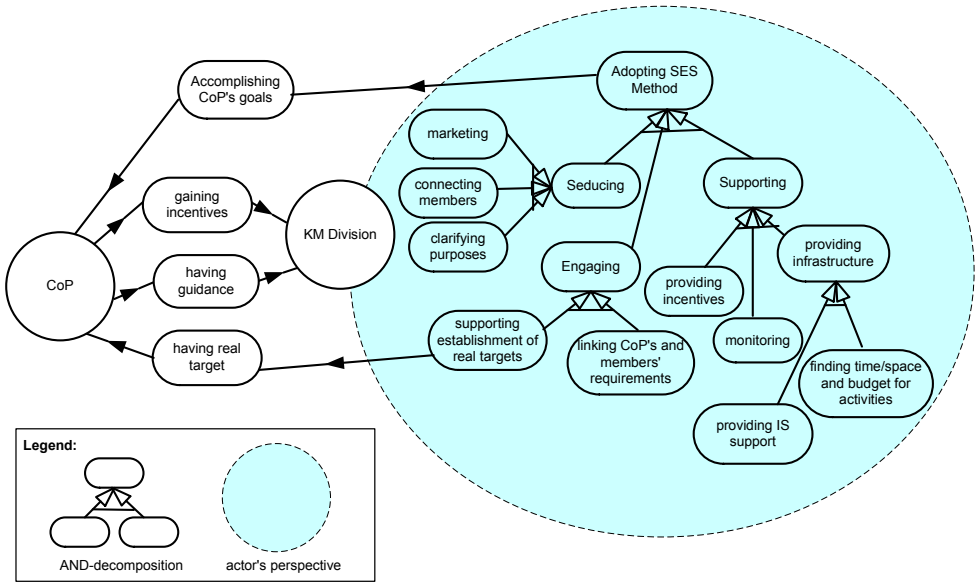


Figure 4.3: Creating a sustainable relationship between KM Division and CoP

The particular perspective of one or more agents can be analyzed using the three basic goal analysis techniques provided by Tropos: *means-end analysis*, *contribution analysis* and *AND/OR decomposition*. This allows the refinement of the domain model by identifying new agent dependencies, delegations and/or acquisitions. Figure 4.3 allows the analysis of the internal perspective of the KM Division, which adopts the SES method (adopting SES method goal). This initial method has been analyzed and decomposed in sub-goals (AND-decomposition), providing us with an overview of the SES method, which comprises three phases: seducing, engagement and support phases (seducing, engaging and supporting goals). During the first phase, *seduction*, the context and aims of a CoP are identified and described (clarifying purposes goal), potential members are made aware of their connections and common interests (connecting members goal), and a "marketing campaign" is launched, showing the added values and benefits of the CoP for the whole organization (marketing goal). In the second phase, *engagement*, both community members as organization are involved in the process of setting up the CoP. The aim is to design a community that is as closely related as possible to the requirements and wishes of the members (linking CoP's and

members' requirements goal) and whose tasks and targets are well embedded in the strategic priorities of the organization (supporting establishment of real targets goal). The aim of the third phase, *support*, is to consolidate the CoP, by developing CoP-specific methods and tools for the organization, management and innovation of CoP activities (providing infrastructure goal), besides verifying its progress (monitoring goal) and granting incentives (providing incentives goal).

Note by the goal delegations between CoP and the KM Division that the latter keeps its old delegations toward the former. But now, the CoP relies on the KM Division for getting incentives to develop its activities (getting incentives goal), and for having guidance throughout its lifetime (having guidance goal). This corrects the previous unbalance between these two agents, creating a sustainable relationship between CoP and KM Division.

4.4 Focusing on the Perspective of the Newcomer

In order to adjust to his new work environment, the newcomer counts on his colleagues and superiors for helping him fit into organizational practices. Suppose that the analyst wants to understand the processes the newcomer uses for integrating into work. For that, he conducts two separate interviews: one with the organization manager, and the other one with the newcomer. Figure 4.4 models the result of these two interviews showing two perspectives: the manager's perspective (A) and the point of view of the newcomer (B).

Analyzing part (A) of Fig. 4.4, we realize that with the goal of having a prepared team, the Organization Manager worries that the Newcomer gets quickly integrated into work (having newcomer integrated into work goal and quickly softgoal[2]). The Organization Manager thinks that the means

[2]Softgoals are commonly used to provide a quality to the goal. In this case, for example, "quickly"is the way the Organization Manager hopes the goal it qualifies ("having newcomer integrated into work") is achieved.

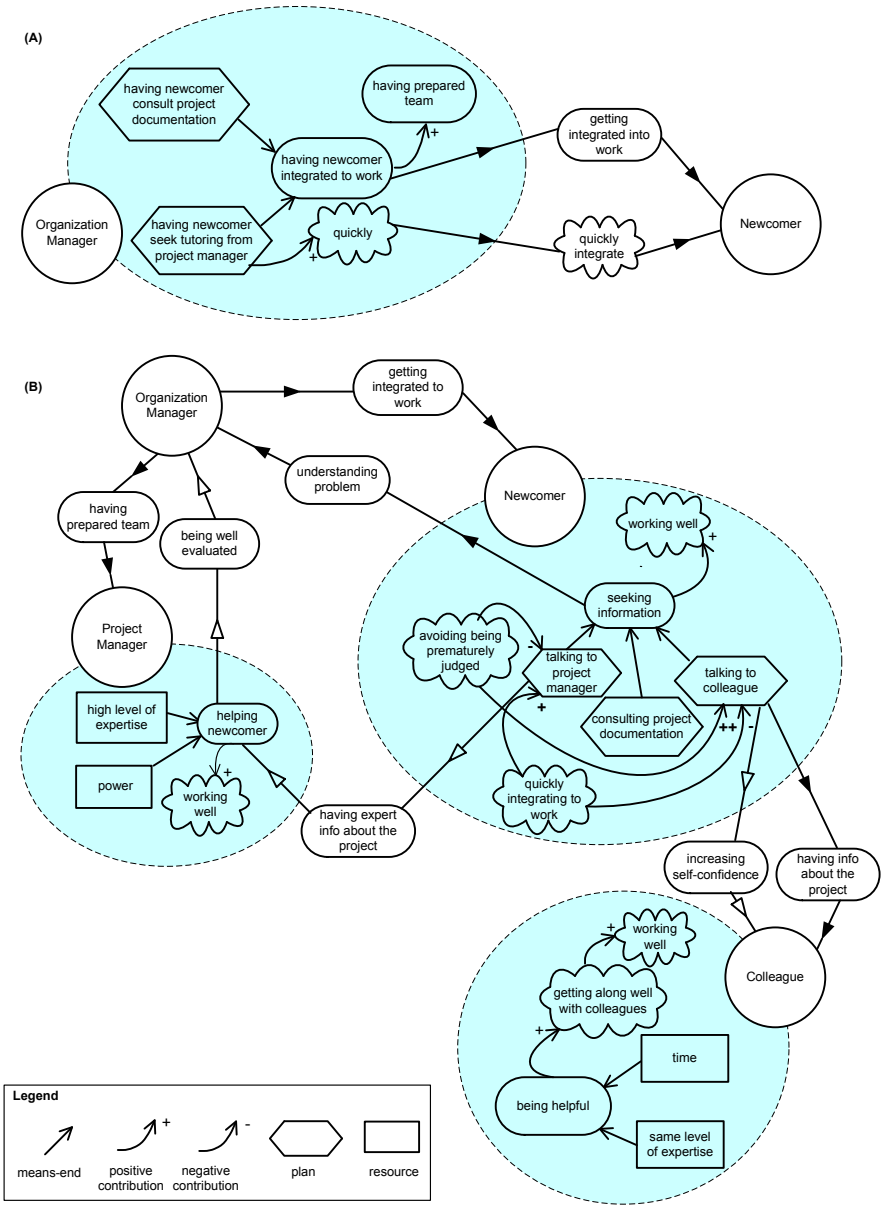


Figure 4.4: Newcomer's integration into work on (A) the organization manager's perspective and on (B) the newcomer's point of view

the Newcomer has to achieve this quick integration is reading the project's documentation and talking to the Project Manager (having newcomer consult project documentation and having newcomer seek tutoring by the project manager plans).

However, in (B), we note that instead of consulting the Project Manager, the Newcomer talks to a Colleague (having info about the project goal delegation). We can understand the reasoning behind his choice by analyzing his perspective. The Newcomer has three plans to seek information about the project: talking to project manager, consulting project documentation and talking to colleague. He believes it will be quicker to integrate into work if he talks to the Project Manager (positive contribution going from quickly integrated into work softgoal to talking to project manager plan), since he is an expert on the project (high level of expertise resource). The Newcomer is thus dependent on the Project Manager for knowing well about the project since being an expert, the Project Manager is most qualified person to provide project information (getting expert info about the project goal dependency). Besides expertise, the Project Manager has the power (power resource) to acquire any means the Newcomer might need to complete his work, such as a software or a book. But on the other hand, these same high level of expertise and power resources make the Newcomer feel pressured, thinking that he might make a mistake and the Project Manager may prematurely judge him as being incompetent (negative contribution going from the being prematurely judged softgoal to the talking to project manager plan). On the other hand, this feeling of pressure no longer exists in the case of talking to a Colleague (double positive contribution going from the being prematurely judged softgoal to the talking to colleague plan), because the Newcomer perceives his Colleague as having the same level of expertise than him (same level of expertise resource). Hence, we can say that the Newcomer depends on his Colleague to gain self-confidence (increasing self-confidence goal dependency) before thinking of interacting to the Project Manager. Analyzing the different contributions to the Newcomer's plans, we realize that this dependency gains strength and leads the newcomer to talk to his Colleague. The assessment of contributions (positive and negative) is a common analysis method

offered by Tropos and here, it allows us to understand the reason why the Newcomer chose one path of action instead of another.

Part B) of this diagram also illustrates the divergence between the constructs of dependency and delegation in ARKnowD. As pointed out by Castelfranchi et al. (1992), understanding the dependencies between different agents within the organizations may provide good means for finding new opportunities for collaborating and working more efficiently. In other words, once noted, such dependencies may lead to future delegations. For example, having observed this specific situation can lead the analyst to make suggestions to the organization manager. The analyst may suggest that in order to guarantee that the newcomer has correct information, the manager should not expect the newcomer to take the initiative to talk to the project manager, but have the project manager contact the newcomer instead. Another possible recommendation is that the organization and project managers assure the newcomer that making mistakes is not seen as a problem in the organization, diminishing his feelings of pressure and uneasiness. Accepting mistakes as learning opportunities rather than flaws is the right kind of attitude toward guaranteeing a conducive environment for Knowledge Management (Lave et al., 1991).

In this diagram, we make a small extension to the semantics of the Tropos notation. In Tropos, only concrete resources are considered in the analysis (e.g. a tool, a document, or a piece of information). However, here we represent *intangible assets*, such as personal characteristics of the project manager and the newcomer's colleague. The reason behind this choice is that in a KM scenarios, it is usually very important to analyze the social behavior and relationships among agents, for which this kind of intangible assets may be very valuable.

Fig. 4.4 shows that the same situation may be modeled in different ways, when seen through the eyes of different agents in the organization. This shows the importance of interacting with agents from different hierarchical positions in the organization, so as to have a more refined view of the working setting. As pointed out by many KM researchers (Brown and Duguid, 2000) (Wenger, 1998), the description of the work provided by managers

or written in manuals often diverge from the actual practices within the organization. Knowledge workers usually find several shortcuts and choose different paths to solve a problem more efficiently or in a way they find more reasonable (Wenger, 1998). ARKnowD can be applied to model these differences, helping the analyst to reason and communicate about inconsistencies between the perspectives of two or more agents.

4.5 Joining a Community of Practice

In order to learn more about the organization, the newcomer takes the initiative of joining the OpenS community of practice. This is modeled in Figure 4.5. Here, the internal goals of the Newcomer are analyzed and the delegations towards the CoP, motivated by these goals, are identified.

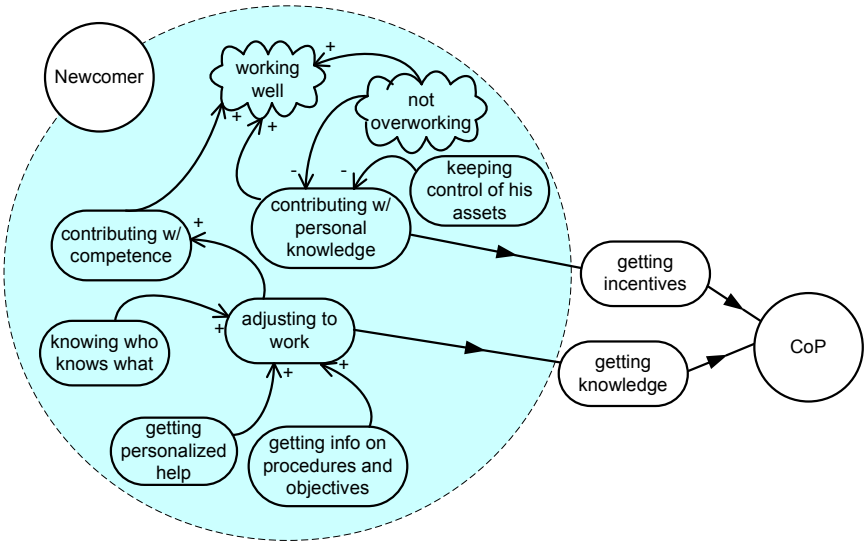


Figure 4.5: Newcomer's perspective when joining a CoP

The Newcomer's most general goal is the working well softgoal, i.e. he aims at doing his work efficiently, while also feeling good about himself and about the organization as a whole. In order to accomplish this, he aims at contributing with his competence and contributing with personal knowledge, gained in previous personal and professional experiences. Going deeper in the analysis of this last goal, we see that two other goals contribute negatively

towards it (not overworking and keeping control of his assets goals). These are common problems already noted by the KM community. Issues of trust (keeping control of his assets goal) and motivation (not overworking goal) often lead to dissatisfaction towards the traditional centralized KM systems (Pumareja et al., 2003) (Orlikowski, 1992a). These goals have profoundly impacted some of the choices concerning the requirements of a supporting Information System (discussed in section 4.7).

Let us now analyze the contributing with competence goal a bit further. In order to fully and most effectively contribute with his acquired competence, the Newcomer needs to adjust to his work environment (adjusting to work goal). Three goals contribute to the Newcomer's adjusting to work, namely: knowing who knows what, getting personalized help, and getting info on procedures and objectives.In order to do adjust to work, the Newcomer needs new knowledge about his work and about the organization as a whole, which leads to the Newcomer's delegation towards the CoP for getting knowledge. As we can note in this example, relying on Tropos, the ARKnowD methodology allows the analysis about the reasons behind certain delegations. In other words, the goal linked to the delegation's outgoing arrow (here, the adjusting to work goal) provides the 'why' of the delegations between two agents (in this case, the delegation of the Newcomer towards the CoP to get new knowledge).

4.6 Adding New Agents: Detailing the CoP Structure

New agents can be added during the analysis in order to model specific roles associated to an agent, or to new particular agents needed for goal dependency and/or delegation. In Figure 4.6, inside the dotted rectangle, we analyze the structure of the CoP agent, which can provide a solution to the goal delegations involving the newcomer, with respect to knowledge providing/getting.

As pointed out in the ontology developed in section 3.5, ARKnowD differ-

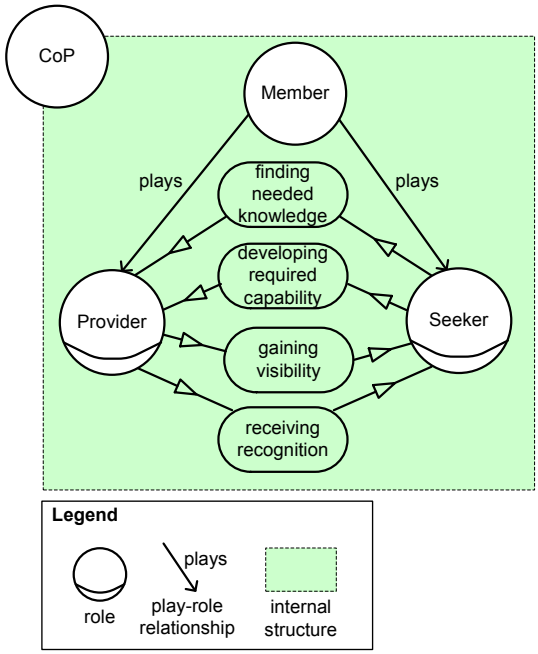


Figure 4.6: The internal structure of the CoP

entiates between *agent* and *role*, inspired by the definitions of i* (Yu, 1995). An agent has a concrete, physical manifestation, such as a human being, an organization, or a software system. A role is an abstract characterization of the behavior of a social agent within some specialized context or domain of endeavor. In this sense, its characteristics can be transferable to other social agents, i.e. roles are used to represent specific behaviors independently of who plays it.

Note that so far, we talked about specific agents, such as the Newcomer, the Organization Manager and the CoP. In the example of Figure 4.6, we generalize the behavior of persons who provide and search for knowledge within the CoP using two roles, namely the roles of Provider and Seeker. And in addition to this, we say that the Member agent (representing a member of the community) can interchangeably play these two roles (note the assignment labeled as "play" going from Member to each the two roles). Roles could have been used to represent organizational roles, like 'manager', for example. But for our purposes, it is more convenient to have 'manager' and other organizational roles represented as agents, assuming that for example,

a manager is always a manager (rigid entity). This gives us the chance to use roles to represent entities that are antirigid in our model, such as those of Provider and Seeker. In other words, a Provider at one moment can be the Seeker in a future opportunity.

Besides making this differentiation, the diagram shows the mutual dependency between the Seeker and Provider roles. The Seeker depends on the Provider for the goals of finding the knowledge he needs for a specific task (finding needed knowledge goal), and of developing new capabilities (developing required capability goal). On the other hand, the Provider depends on the Seeker to gain visibility (gaining visibility goal) and to help the Seeker to develop new capabilities, so receiving recognition (receiving recognition goal). Note that members of the community of practice may play both roles in different situations, generating a community of peers. Another interesting point is that as soon as a Newcomer participates in the CoP, he will play the two Member roles.

Considering the objectives a CoP is designed for will assure the satisfaction of the previously identified goal delegations between the Newcomer and the CoP. But could the Newcomer rely on the CoP to get personalized help when having a problem to solve (getting personalized help goal)? Should CoP goals and structure be modified?

4.7 Identifying the Needs for the KARe System Agent

In order to fulfill some of the goals of the Newcomer, the CoP structure has to be revised and needs to accommodate new agents (roles), having the ability to satisfy them. Figure 4.7 depicts a model showing that the CoP delegates some of the goals of the Newcomer to an agent named KARe.

The CoP delegates to KARe the goals of:

a) letting users keep control of their knowledge assets (directly derived from the Newcomer's keeping control of his assets goal). The KARe

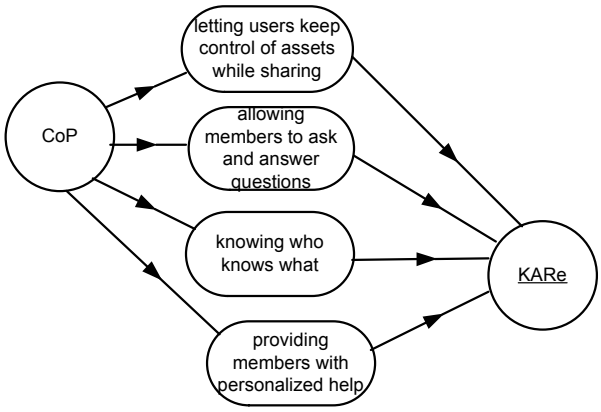


Figure 4.7: Goal delegations from the CoP to the KARe System

system should allow each user to keep their assets in their own PCs, while making them available to other community members;

b) allowing members to ask and answer questions through messages exchange. This feature is important because some of the Newcomer's questions may not be answered by reading artifacts. Sometimes, it could be necessary to communicate with CoP members for building the solution to a specific problem, as in Fig. 4.4, where the Newcomer talks to his Colleague to gather project information. KARe should mediate this interaction, by finding the best person to answer to a specific knowledge request;

c) informing who knows what (directly derived from the Newcomer's goal of knowing who knows what);

d) providing members with personalized help, by considering their personal characteristics when providing knowledge (straightly obtained from the goal of getting personalized help by the Newcomer).

The four goals listed above become KARe's main requirements. In this chapter, we exemplify the elicitation of system requirements with an analysis of a simple case study involving the adjustment of a newcomer into work. However, we acknowledge that, in real cases, this elicitation may be motivated by the analysis of much more complex scenarios.

By analyzing the four goals from the point of view of the system agents, we can identify more detailed requirements, and analyze alternative solutions. For instance, we may consider satisfying KARe's goals by defining new artifacts to be produced along the organization's processes or we may look for what KM enabling technology can be considered to design a better solution. A detailed proposal of the KARe system, along with its design model can be found on chapter 5.

4.8 Adjusting the Evaluation Method

After the Newcomer had been participating for some time in the activities of the CoP, a subsequent analysis showed that his participation had not been very high, and this could be explained by the evaluation method adopted by BHI. Figure 4.8 (A) captures this problem.

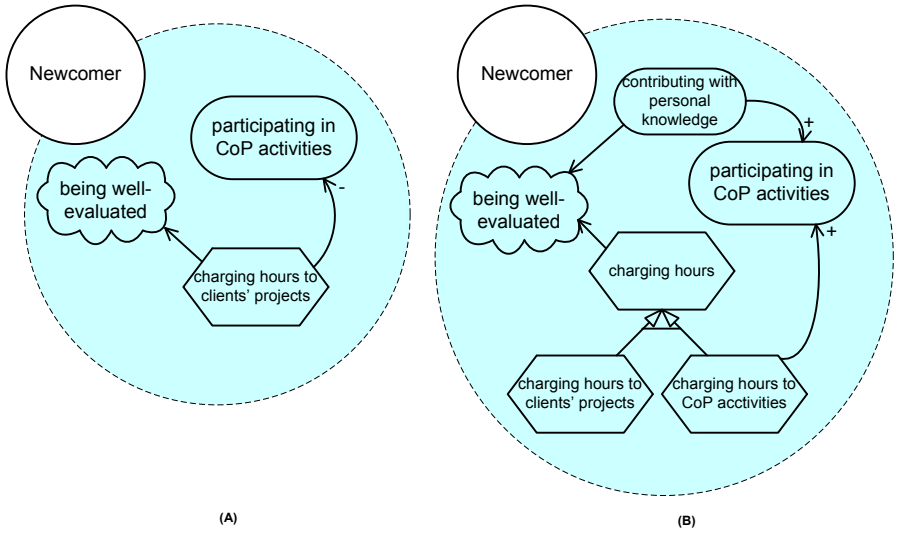


Figure 4.8: Performance evaluation affecting participation in CoPs

Fig. 4.8 shows that the evaluation of the Newcomer's performance was based on the hours he charged on clients' projects in which he was involved. So, in order to be well-evaluated, the Newcomer needed to spent as much time as he could on tasks related to his usual programmer's tasks (charging hours to clients' projects plan), which did not give him much time to get

involved in CoP activities (negative contribution towards the participating in CoP activities goal). This is a common problem noticed in the KM literature: although the organization makes some effort to support KM, its evaluation methods are not accordingly adjusted to accommodate activities related to knowledge creation and sharing (Orlikowski, 1992a). In order to solve this problem, a new evaluation method is proposed, as shows Figure 4.8 (B).

In the diagram of Fig. 4.8 (B), in order to be well evaluated, the Newcomer must charge his hours to clients' projects (charging hours to clients' projects plan), but also to CoP activities (charging hours to CoP activities plan). And besides, he should contribute with his personal knowledge (contributing with personal knowledge goal). These additions contribute positively to his participation in CoP activities (positive contribution to participating in CoP activities goal). So, by adopting this new evaluation method, the organization is motivating the participation on CoP activities to grow.

Note that while the charging of hours follows an established procedure (this can be understood by the semantics of the *plan construct*, used to represent it), the personal knowledge contribution (represented as a *goal*) can be achieved in different ways, to be decided by the Newcomer. For instance, he could provide contributions to his peers by using the KARe system, emailing his colleagues, meeting them in person, etc.

4.9 The Conducted Analysis in Light of Constructivist KM

Looking back at the analysis conducted in this chapter, we realize many points of connection to the previous reflections regarding the principles of Constructivist KM (or building blocks, as we called them in section 2.5. In fact, these building blocks may be used as a checklist to guide the analyst in understanding which level of support the organization currently provides to Constructivist KM. Next to this, these principles may also provide insights about how to improve KM processes within the organization, either by changing organizational structures and processes, or by adopting techno-

logical and non-technological tools.

First of all, the way the analysis has been carried out, based in the point of view of several actors within the organization is compliant to the *autonomy* building block, previously emphasized as essential to bring motivation to employees regarding the organization's KM practices. As in top-down approaches commonly adopted in current KM initiatives, the analysis considers the strategic intents of the organization's top management. However, besides that, it also takes a bottom-up perspective, bringing into evidence the needs and wants of the knowledge holders, i.e. individuals working in the several points of action of the organization.

Other elements of the analysis point out that autonomy is a valued principle in the illustrated organizational setting. Profiting from ARKnowD's analysis techniques, the existence of more or less goal delegations compared to plan delegations among agents leads the analyst to draw conclusions regarding the choices of the members of the organization regarding autonomy. As mentioned before, goal delegation represents situations in which the decision of the strategy behind goal accomplishment is left to the delegatee. Conversely, in cases modeled with plan delegation, the delegator prescribes the way the delegated goal should be achieved. As an illustration, in part A) of Fig. 4.4, the top manager lets the newcomer decide how to become more familiar with the project in which he is going to work (goal delegation), rather than prescribing to him a specific way to proceed.

The autonomy building block is also considered when the organization decides to support the emergence of CoPs. These communities usually grant their members with autonomy to choose 'when', 'how' and 'with whom' to exchange information and expertise. In addition to that, CoPs create a rich *context* for knowledge creation and dissemination. In this context, new knowledge is produced and disseminated not simply to fill in an existing knowledge base, but with the purpose of fulfilling a need of the community. These needs are generally conditioned by the targets defined according to organizational strategies.

Moreover, crossing divisions and even branches of the organization, CoPs

also motivates *non-hierarchical knowledge sharing*. In other words, participation in CoPs' activities independent on organizational roles and positions. Although these positions exist and are important for carrying out specific tasks and responsibilities within the organization, the analysis shows that in the case of knowledge exchange, everyone is considered equal. This uniform treatment is reflected in the representation of two roles (knowledge provider and knowledge seeker) to be assumed by every CoP member, despite their organizational roles and positions.

The adopted technological solution also supports the Constructivist KM building blocks. This is more specifically focused on the subsequent section, which discusses the requirements elicited to the KARe system as a result of the conducted analysis.

4.10 Focusing Closer on the KARe System Requirements

Section 4.7 focuses on the elicitation of the main requirements of the KARe system, based on the needs and wants of the domain stakeholders already captured in previous models. The resulting set of requirements is presented in Table 4.1.

KARe Requirements
R1: Allowing users to keep control of their knowledge assets while sharing knowledge
R2: Supporting members to ask and answer question
R3: Providing information on experts regarding particular knowledge
R4: Providing personalized help to the users

Table 4.1: The requirements elicited for the KARe system

Aiming at providing support to Constructivist KM, KARe adopts a peer-to-peer model. Peer-to-peer networks provide at the same time, a framework for *social interaction* and for *physical and meaningful artifacts* access and exchange. What makes peer-to-peer different than the traditional client-

server approach is the fact that there is no central server, i.e. each node of the network can play the role of either a client or a server (Tiwana, 2003). In this way, by adopting a peer-to-peer model, KARe directly supports **R1** listed above, allowing each member to keep their knowledge assets stored in their computer while making them available to others in the network. This complies with the principle of *autonomy* characterizing Constructivist KM. Figure 4.9 illustrates this model.

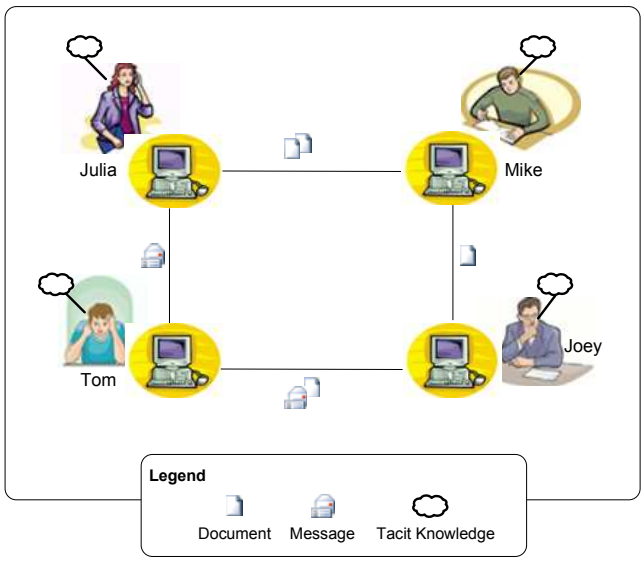


Figure 4.9: Peer-to-peer knowledge sharing

Fig. 4.9 shows four people, each one locally managing their own knowledge assets. In KARe, we classify knowledge artifacts into **documents** and **messages**. A document is considered as a "finished product" regarding a specific subject. Documents can have various formats (text, audio, video, etc.) and be of different types (manual, report, speech, etc.). A message, on the other hand, refers to a communication construct, used to mediate dialog and discussion. The system motivates *social interaction* by supporting the members of the organization to ask and answer questions (requirement **R2** listed above). In this way, the members of KARe's peer-to-peer community exchange the knowledge artifacts maintained in their personal collection. In other words, KARe aims at imitating the social processes commonly applied when one has a particular problem to solve during one's daily work. Instead

of consulting manuals and documentations, the worker is motivated to get involved in a dialog with workmates, which may lead him/her to grasp more than procedures, the values and tacit strategies adopted in the organization. Fig. 4.9 also acknowledges the presence of tacit knowledge, which is hoped to be explicitated throughout social interactions, especially with the exchange of questions and answers (i.e. messages).

The peer-to-peer model reflects the non-existence of a central power or any kind of authority controlling the peers interactions and exchanges. In this way, despite of hierarchical roles or positions, all peers are seen as providers and seekers of knowledge. In other words, this model complies with the Constructivist KM building block of providing a *non-hierarchical knowledge sharing structure*. The distributed nature of peer-to-peer networks reflect the naturally distributed character of knowledge and expertise within an organization. Knowledge and expertise exists sparsely in the (internally understood and shared) members' understanding of each other's knowledge, and in the (hidden) behavioral and cognitive similarities among individual users. In this respect, KARe aims at uncovering and making salient some of these hidden characteristics so that users might become more aware of the existing organizational knowledge and its corresponding CoPs. This directly regard requirements **R3** and **R4** listed above. Both the information about experts in particular subjects, and the personalized help are obtained through the adoption of *user models*. In KARe, user models describe personal characteristics of a user and how he/she views the other peers in the peer-to-peer network.

In an organizational setting, employees can interact as peers of a big peer-to-peer network and naturally group themselves in communities. Besides, such network can even transcend organization boarders, allowing external actors or other organizations to take part in this knowledge sharing space. The dynamic environment created as a result of such knowledge exchange enables the emergence of constructive *perturbations* that lead people to seek self-improvement by asking questions, and reflect about their knowledge when answering to incoming knowledge requests.

4.11 Conclusions

This chapter has started illustrating the application of the ARKnowD methodology, focusing on the initial activities of the development cycle, namely the requirements elicitation and analysis activities.

We used a fictitious scenario especially elaborated based on KM literature for the purposes of this exemplification. With this scenario, we tried to illustrate some of the main problems involving KM settings, such as: lack of trust and motivation to provide knowledge, the non existence of an incentive policy towards knowledge sharing, and an inadequate employee performance evaluation criteria for organizations supporting KM. In addition to that, we made sure all constructs of the Tropos notation, as well as techniques applied in this methodology, have been properly illustrated throughout the chapter. Table 4.2 summarizes the illustration of the Tropos's constructs and analysis techniques throughout the chapter.

Our analysis shows how the Tropos methodology can be effectively used to elicit the requirements of KM support, in terms of *processes* and by adopting a *supporting Information System*. Two examples of process changes suggested by the analysis have been captured in the diagrams of Figs. 4.3 and 4.8. The former models the adoption of a CoP supporting method by the KM Division, in order to balance the delegations between the CoP and the KM Division, in a way that both gain from their collaboration. The latter focuses on the adjustment of the organization's evaluation method in order to motivate the involvement of the organization's members with CoP activities. Moreover, our analysis supports the elicitation of requirements for a KM system, named KARe (depicted in Fig. 4.7). A detailed proposal of the KARe system, along with its design model can be found on chapter 5, while its implementation is discussed on chapter 6.

At the end of this analysis step, a review activity is conducted, aiming at verifying the achievement of the main objectives of domain analysis and requirements elicitation. In other words, we focus on the understanding of critical dependencies and delegations between domain agents for individual goal achievement and the identification of needs for new or alternative

Construct / Analysis Technique	Figure							
	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8
agent	x	x	x	x	x	x	x	x
goal	x	x	x	x	x	x	x	x
dependency				x		x		
delegation		x	x	x	x		x	
softgoal	x	x		x	x			x
plan				x				x
resource				x				
role						x		
decomposition			x					x
means-end analysis				x				x
contribution analysis				x	x			x
agent's perspective analysis			x	x	x			x
agent's internal structure analysis						x		
comparing different perspectives				x				

Table 4.2: Summary of Tropos's constructs and analysis techniques illustrated in this chapter

agents, dependencies and delegations to manage unsatisfied goals. Decisions on what to focus on in the following steps and on which alternative solutions to be refined in an architectural design, are taken at this point.

For instance, at that point of the analysis of the case study we choose to go ahead analyzing a KM solution resting on the adoption of a KM system based on peer-to-peer technology: the KARe system, introduced in the diagram of Fig. 4.7 . This choice complies with the Newcomer's wish to keep control of his knowledge assets, besides resting on considerations about the effectiveness of this technological solution in favoring knowledge sharing through questions and answers. Moreover, the analysis of our scenario pointed out the relevance of managing tacit knowledge, i.e. the knowledge which is confined in people's mind, and to transform it from tacit to explicit and back to tacit, completing the knowledge creating cycle as proposed in (Nonaka and Takeuchi, 1995). Section 4.10 presented a flavor on how the elicited requirements are explored, but a more detailed description and the design of the KARe system are the subject of the next chapter.

Chapter 5

The KARe System

"The important thing is not to stop questioning.
Curiosity has its own reason for existing."
Albert Einstein

In this chapter, we present the proposal and design Knowledgeable Agent for Recommendations (KARe). KARe is a socially aware recommender system that tries to simulate the social behavior of a community of practice (CoP) within a peer-to-peer network. It maintains the characteristics of CoP members in user models and seeks for knowledge on their behalf, according to their needs and interests.

Besides a comprehensive description of the system, this chapter presents its design, applying ARKnowD. First, Tropos is used to model the architecture of the system and later, AORML is applied in the system's detailed design. Maintaining the consistency throughout the design, we make a conversion between the notations of Tropos and AORML, according to the transformation rules presented in chapter 3.

This chapter is organized as follows: section 5.1 introduces this chapter, describing how KARe models the users and support them reactively and proactively; sections 5.2 and 5.3 present KARe's design. The former focuses on the high level architectural design while the latter deals with the detailed design of the system; section 5.4 presents an initiative to integrate KARe with other two related tools, providing a comprehensive toolbox to support

KM; section 5.5 discusses some related work, based on an innovative layered model that facilitates the classification of KM systems and their comparison to KARe; and finally, section 5.6 presents the conclusions of this chapter.

5.1 Introduction

As an organization develops, its knowledge and expertise becomes increasingly distributed. While promoting the growth of specialized knowledge communities, this process also makes discovering relevant knowledge from these communities more difficult. A KM system should support the natural organizational processes that promote knowledge creation and exchange, thus supporting the members of the organization to find relevant knowledge. With this in mind, in the previous chapter, we analyzed the problem of a newcomer who wanted to join a particular community within the organization where he works. As a result of this analysis, we have elicited a few requirements for a system that addresses such problem.

The elicited requirements have been presented and briefly discussed in section 4.10. As pointed out in that section, KARe aims at fulfilling these requirements by following a peer-to-peer model, which directly resembles the distributed nature of knowledge and expertise within the organization.

Instead of requiring the user's to seek a knowledge base through keywords, the system supports the user on sharing knowledge through *questioning and answering*. According to Freire and Fagundez (1992) (refer to section 2.3.4), a question is the first knowledge sparkle, as questioning is a means to explicitate one's personal knowledge, starting with a reflection on what one knows and what one does not know. In addition to that, questioning provides an opportunity for others to express their points of view, many times tacit. Allowing peers to ask and answer questions, KARe creates a rich environment for social interactions, already recognized as a driving force behind knowledge creation and innovation. Throughout time, knowledge is stored by the community members in their local repositories, allowing KARe to retrieve knowledge to the user, both reactively (at user request) or proactivelly

(autonomously identifying user's needs).

Seeking at providing personalized assistant to the users, helping them find the best responders to solve their doubts, KARe relies on describing each user by the means of a *user model*, that captures cognitive and social characteristics of the user.

The following three sub-sections provide more details about the system's user modeling feature, and clarify KARe's reactive and proactive support. They contribute to the clear understanding on how KARe fulfills the previously elicited requirements, leading the way to consistent architectural and detailed system design.

5.1.1 User Modeling in KARe

User models are composed of different user properties that define their characteristics and preferences, which are particular for each system and/or purpose. For instance, in e-learning systems, user models are used to determine learners' progress in a course, and their performance level regarding specific knowledge or skill. In e-commerce, user models indicate preferences and needs of users regarding particular products. For KM, user models add flexibility to the system. They often support the system to find experts to answer to specific knowledge requests, and to deliver knowledge according to particularities of the users or of the situations in which they are involved.

Central to this discussion is the decision regarding which properties should be considered in the user model. When people share their knowledge with others, they are not just sharing information; they are also sharing cultural and social references (Mantovani, 1996). Likewise, when people seek knowledge, they are not just seeking information; they are seeking information grounded in, and carrying different meanings to different social communities. By considering the available literature and carefully examining several KM cases, we have elaborated KARe's user model (also termed *peer model*) based on the characteristics listed in Table 5.1.

The KM literature points out *interest* and *expertise* on particular topics as two of the main characteristics that differentiate organizational members

User Model Properties		
Personal Characteristics	**Interaction Characteristics**	**Physical Context**
interest	trustability	location
expertise	collaborative level	time
role	reliability	
availability		
presentation preferences		

Table 5.1: Relevant user characteristics when searching for knowledge

(Brown and Duguid, 2000) (Hansen et al., 1999) (Quinn et al., 1996). In KARe, both interest and expertise are inferred from the user's knowledge artifacts, organized by the means of personal taxonomies. More details about how such taxonomies support the user on organizing and sharing knowledge may be found in the subsequent section.

Knowing that someone is an expert on particular topic may not be sufficient to indicate him/her as an appropriate knowledge provider. In addition to that, some indication on how well he/she may respond to a query in that topic should also be considered. This we call *reliability* or *expertise level*. In KARe, when receiving an answer to a given help request, the seeker is allowed to evaluate the provider's contribution, giving it a grade that is then stored in his (the seeker's) PM. An average of the available grades is then used to select and rank providers. Note that each seeker has its own evaluation of the providers, meaning that the same person may be evaluated with different reliability levels by different people. This follows our approach to imitate reality, in which people have different and individual views of one another.

Along with reliability, two other factors are described in terms of the peer's interactions with one another (thus listed in column 2 of Table 5.1). *Trust*, also referred as reputation or confidence measure, is claimed in the KM literature as one of the most important factors considered when people share knowledge (Castelfranchi, 2004) (Esfandiari and Chandrasekharan, 2001). In KARe, the peers may indicate those who they trust through a

list of friends. The remaining interaction feature is the *collaborative level*, measured throughout the providers' interactions in the system. If a peer receives questions and does not reply to them, this has a negative impact on his collaborative level, regarding a particular seeker. The opposite happens if a peer promptly replies to incoming requests. By explicitly modeling the peers' collaborative level in KARe, we hope to avoid (or at least diminish) the existence of "free riders", i.e. peers that seek for contributions but never contribute to others (Vassileva, 2002). To guarantee that absent users will not suffer losses in collaborative level, KARe also considers user's *availability*. In other words, KARe allows the peers to indicate how much time they can spend on answering help requests. This way, those peers who have higher availability are more contacted than the others. Besides that, peers may provide information regarding vacation or sick leaves.

Besides the personal characteristics of expertise and interest, KARe relies on the *role* as an important characteristic to drive the choice for particular knowledge artifacts, and/or users to answer to knowledge requests. Peers may play different roles within an organization and community. This refers to organizational roles, such as accountant, secretary, and consultant. But it can also include specific roles played in a CoP, such as CoP leader and CoP web-master. According to research in CoPs, the role(s) a knowledge provider plays within a community may help to determine whether or not the information received is structured at an appropriate level (Lave et al., 1991). As noted in the analysis of the scenario presented in chapter 4 (refer to section 4.4), people tend to feel more comfortable when exchanging knowledge with others with similar roles.

The physical context (Chen and Kotz, 2000) (column 3 of table 5.1) is also taken into account when KARe is searching for knowledge. Here, time and location are explored. KARe considers for instance, if a user is at home or at school, and the time of the day to determine the choice for specific artifacts (e.g. if it is late at night, the user might prefer a shorter or less complex text to read). Section 6.5.2 of next chapter describes a mobile KARe prototype that uses peer locations to select him/her as a responder for pending questions.

5.1.2 Using KARe to Ask and Answer Questions

KARe is a multi-agent system that recommends artifacts to meet the user needs based on questioning and answering, simulating the natural social processes involved in knowledge sharing. When we have a real problem at work, we often rely on asking a question to a colleague with whom we share the office, or to someone who is considered an expert in a subject related to our problem. The process of questioning and answering stimulates the pursuit for innovation. This can prevent the crystallization of knowledge and procedures, impelling people to seek for new and better ways of acting and performing. Moreover, this process forces both questioner and responder to find a common interpretative schema, by putting themselves in each other's position and trying to grasp each of their individual and tacit view on the discussed topic. In order to provide personalized assistant, helping the users to select the best respondents to solve their doubts, KARe relies on user models as described in the previous sub-section, capturing user's cognitive and social characteristics.

Asking and answering to questions is an interactive process. The questioner finds a suitable colleague and poses his doubt. Usually, this choice is based on the questioner's assumption that his colleague knows about the targeted subject, besides feelings of trust and comfort towards the responder (this choice is illustrated in the analysis of the Newcomer's scenario, in the Tropos diagram of Fig. 4.4, chapter 4). The responder, on his turn, is likely to help the questioner, provided that the trust between them is mutual. He will then use his own language and knowledge to provide the answer to the questioner. Besides solving the problem at hand, having the answer gives the questioner the ability to share this new knowledge with other colleagues.

Fig 5.1 illustrates the process of asking and answering questions in three steps. In (A), a questioner named Mike has a specific doubt concerning one of his tasks. He finds a colleague who has the answer to help him and poses his question, receiving the answer subsequently. In (B), Mike applies the received answer in order to solve his problem. Now, keeping the answer to himself, Mike is able to help others in need, as shown in (C) where he

provides the answer to a colleague who had a similar doubt.

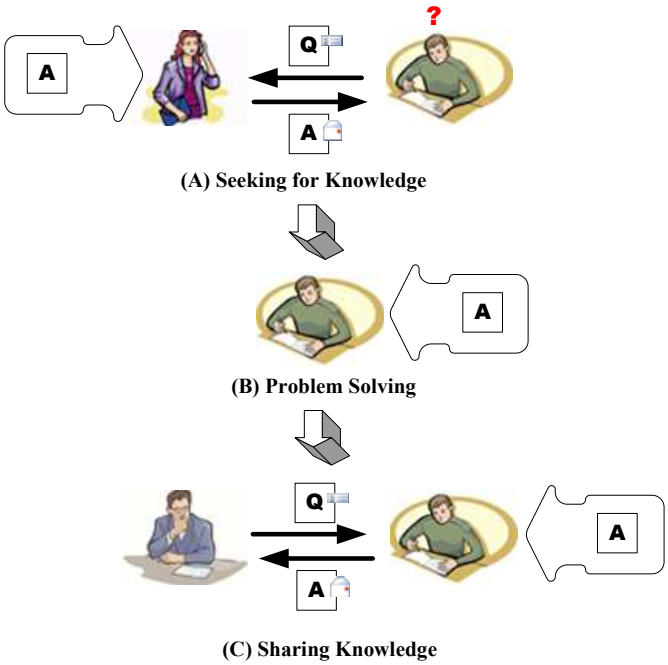


Figure 5.1: Process of (A) acquiring knowledge, (B) using it for solving a specific problem and (C) socializing it with others

In KARe, we simulate this process using a peer-to-peer infrastructure. Each user (a peer) is able to organize his knowledge assets (typically, working documents) according to his own domain conceptualization, using a taxonomy. A taxonomy is a concept hierarchy that describes the user's view of a specific domain, also named context in (Bonifacio et al., 2004). After defining meaningful concepts and their inter-relationships the user distributes the artifacts according to the "matching" concepts in the hierarchy. Figure 5.2 shows the organization of the personal knowledge assets of three users, connected by a peer-to-peer network.

KARe allows the user to pose natural language questions, searching in other peers' collection for answers among their stored artifacts. The answer can be found among documents or messages sent by other peers responding previous similar questions. In case no response is found, the system indicates a suitable peer (based on peer's user models) to provide the answer to that specific question. Having received a suitable answer from the indicated peer,

the questioner now has this answer classified in his own taxonomy and stored in his system, so that he can be consulted by others regarding the same subject. These processes are illustrated in Figures 5.2 and 5.3.

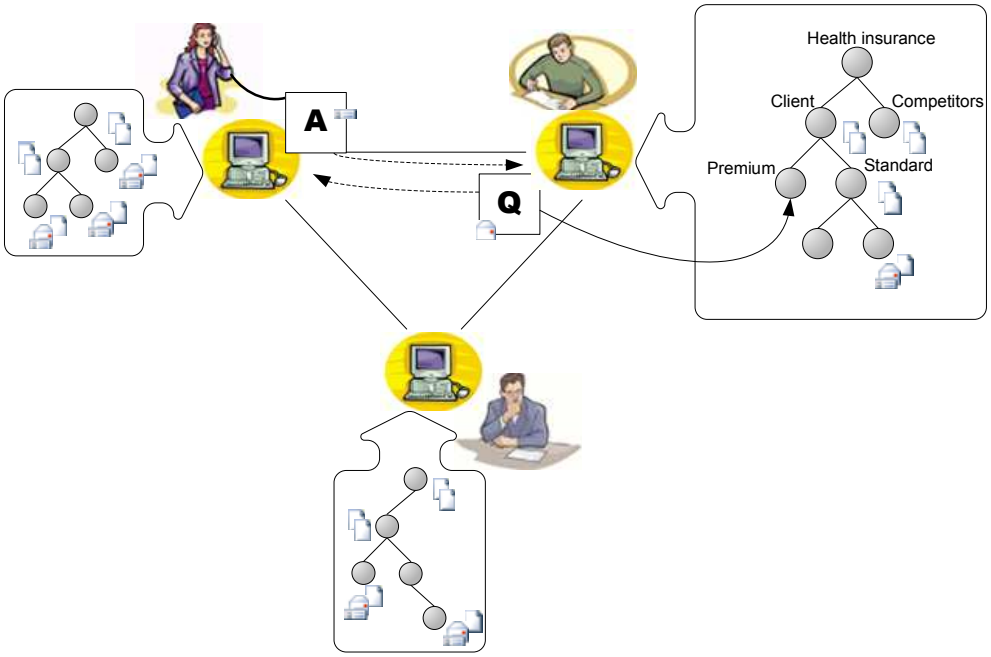


Figure 5.2: A human peer responds to a question when no answer is found by the system

Fig. 5.2 shows how KARe deals with the situation previously depicted in Fig. 5.1 (A), i.e. a user submits a question (now contextualized by a concept in his own taxonomy). The system submits the question to a peer whose user model seems to describe a suitable responder. The user answers to the question submitting it through the system to the questioner. Note that the contextualization of the question may help the responder to understand more about the questioner's doubt. For instance, suppose that this is an insurance company and that Mike's question is "What measures should we take when a client is late with his payment for the acquired services?". Some information is not expressed in Mike's question, for instance: what kind of service is he talking about? However, this information is explicitated by the contextualization of the question, since it is classified under "Health insurance-Clients-Premium" in Mike's taxonomy. Besides clarifying the type

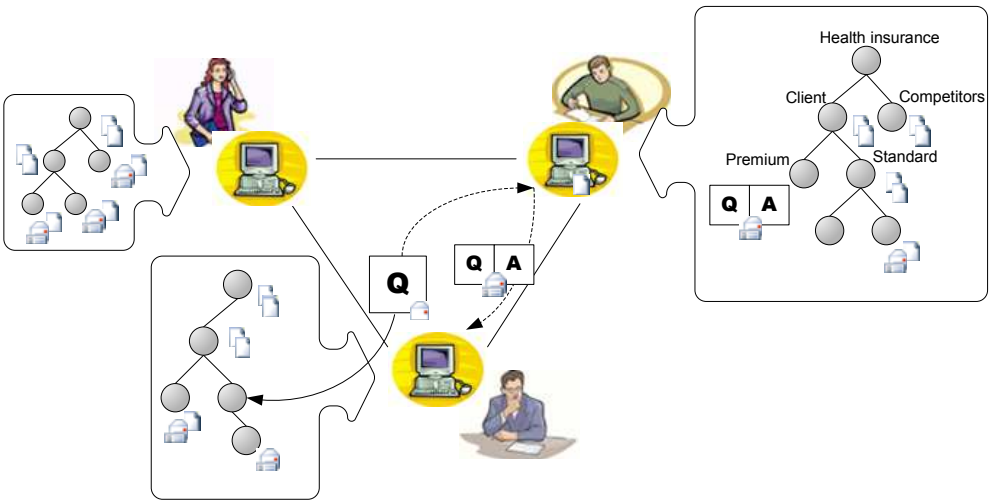


Figure 5.3: The system retrieves an answer previously stored by another peer

of service ("health insurance"), this contextualization also indicates the type of client Mike is reffering to (in this case, "premium"), which may have some impact in the responder's answer.

In Fig. 5.3, Mike has already received the answer to his question, which was then classified under the concept he had previously indicated. In this way, Mike now stores the answer for his own future reference and for sharing it with peers in need. The figure shows the case in which Joey requests similar information, by posing a question similar to Mike's. In this case, Mike does not need to personally answer to the question, as the system has already found it in his computer, subsequently sending it to Joey. This illustrates KARe's handling of the situation early depicted in Fig. 5.1 (C).

On the other hand, if the questioner is not satisfied with the answer, the system provides him with a list of possible responders. Responders are chosen based on their user model, which comprehends the following characteristics (refer to previous sub-section): expertise, reliability, trust, role, collaborative level, and availability.

By storing the same question/answer pair in different peers, we chose to replicate to increase the possibility that this knowledge will remain in the organization even when some members are not available anymore. Considering

that these peers will be involved in continuous interactions, the knowledge considered "useful" to the community (i.e. information and documents they need for their daily work) is likely to remain in the community, even if the members that originally owned them leave. This complies to the principle of *redundancy*, presented by Nonaka and Takeuchi (1995) as one of the conditions for knowledge creation and disseminatio within organizations (for more on this topic, refer to section 2.2.4).

5.1.3 Proactive Knowledge Delivery

Besides this reactive type of search, KARe also searches for knowledge artifacts in a proactive fashion, in three distinct ways: a) periodic search; b) solving pending requests; and c) suggesting interaction with similar users.

In the periodic search, KARe recurrently searches the distributed peers for new knowledge artifacts of interest for his/her associated peer. The period of search is previously configured by the peer, and it can be also changed throughout time. When searching for knowledge, KARe takes into account the following information from the user model, described in the previous section: seeker's interest, provider's expertise, seeker's trust on certain providers, and seeker's role. In addition to that, the system is aware of location and time, besides user's preference, in order to present the content in the most appealing format.

In solving pending requests, the system tries to solve previous knowledge requests that have not yet been satisfied. There may be situations in which a request results in no satisfactory response, either because no knowledge has been found among the other peers, or because the founded documents or incoming answers have not satisfactorily solved the questioner's doubt. In this case, this request becomes pending, and the system keeps track of new incoming knowledge artifacts that may suffice that particular knowledge need, delivering them to the questioner in a later date.

Finally, when noticing similarity regarding interest, expertise and the remaining factors considered in the user model, the system may provide a peer with a list with similar users to which he/she may interact.

5.2 Architectural Design

At this point, we would like to go back to the modeling activities we have started in chapter 4. The next activity is the system's architectural design. During this activity, new agents emerge as sub-agents of the KARe system. These new agents are the choice of the system designer to fulfill the requirements captured in the requirements analysis activity. The designer usually bases her/his choice of architecture on previous experience or on available architectural patterns, previously used for similar purposes.

According to ARKnowD, besides supporting requirements analysis, Tropos is also applied for the architectural design, providing a smooth transition from the problem level analysis to the system level analysis. It enables us to easily trace back the functionalities of the system to the goals of the domain agents. In more details, the main advantages this approach gives us are: a) allowing us to analyze which of the system's general goals are adopted by each of the internal agents; and b) supporting us on capturing the goal delegations[1] between the system's internal agents. For detailing the design of the system (refer to section 5.3), we finally make the transformation proposed in chapter 3, proceeding the design with the use of AORML (Wagner, 2003), which can support information modeling, besides capturing agents' behavior and interaction.

The analysis of the requirements of the KARe system leads to the identification of a possible structure of the system agent in terms of sub-agents, i.e. the system global architecture is identified through delegation of main system goals to internal sub-agents. In this way, KARe's architecture is composed of three main sub-agents:

- Artifact Manager (AM): maintains the personal knowledge repository of the peer. This means maintaining the taxonomies created by the user, besides allowing him to include or exclude items from the repository.

[1] If during analysis, it is possible to capture dependencies between agents that are uncommunicated and even unknown, in the design of system agents, dependencies are rare. Instead, relations between agents at the level of architectural design are typically captured by the use of delegations.

- Peer Assistant (PA): each peer has an associated Peer Assistant (PA) that represents him/her in the network of peers. This agent is in charge of searching for knowledge on peer's behalf, reactively and proactively, and in a personalized way.

- Broker: matches peers as adequate knowledge sources for specific requests, and supports proactive search by matching similar peers. Note that while the other two agents compose the peer-to-peer infrastructure, the Broker is a centralized entity.

The emerging structure is that of an agent organization (or more generally of a peer-to-peer system), whose high level architecture may be modeled in terms of goal delegations, according to Tropos, as in the example depicted in Figure 5.4.

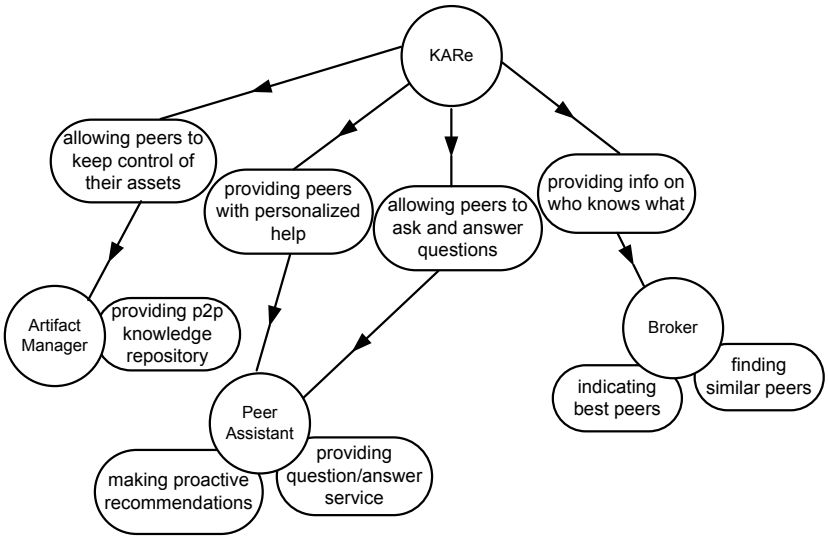


Figure 5.4: Tropos diagram showing the high level architecture of the KARe system

KARe is depicted on the top of Fig. 5.4, delegating the four goals previously adopted by the system on behalf of the CoP (refer to chapter 4, Fig. 4.7) to the three agents described above: the AM the PA, and the Broker.

KARe delegates to the AM, the goal of allowing users to keep control of their assets. In order to fulfill this goal, the AM maintains a peer-to-

peer knowledge repository (providing p2p knowledge repository goal). In this way, the users are allowed to organize their knowledge assets locally, while exchanging knowledge with other peers

The PA is responsible for the goals of: providing peers with personalized help and allowing peers to ask and answer questions. These goals materialize, by the use of technological solutions, two important goals delegated by the KARe system and are further refined, providing us with more details regarding the proposed architecture. To achieve these two main goals, the PA provides the user with proactive recommendations (making proactive recommendations goal) and supports a question and answer service (providing question & answer service goal).

The last goal of KARe, namely the providing info on who knows what goal, is finally delegated to the Broker. Hence, the Broker is responsible for indicating who are the best users to answer to a certain knowledge request (indicating best peers goal). Or, in case of a proactive search, the Broker indicates who are the peers similar in regards to a specific user (finding similar peers goal).

The existence of a central entity, namely the Broker, indicates that KARe does not adopt a pure peer-to-peer model (like Gnutella, for example), but is rather designed following the hybrid model, also adopted by Napster (Oram, 2001). The idea behind this choice is allowing more flexible support to the users of the system, facilitating the access to the user models containing personal and social features of the system peers. Figure 5.5 shows a possible peer-to-peer network topology where all six network nodes contain an AM and a PA, but only two nodes contain Brokers. Limitations of safety and performance of the system (in case a network node containing the Broker is down or overloaded) may be overcome by replicating the Broker in other nodes of the network.

Fig. 5.4 presents a first architectural model, which may be subsequently refined, allowing us to understand better by which means the goals are achieved, besides detailing the delegation relations between the architecture's agents. Figure 5.6 presents such refinements.

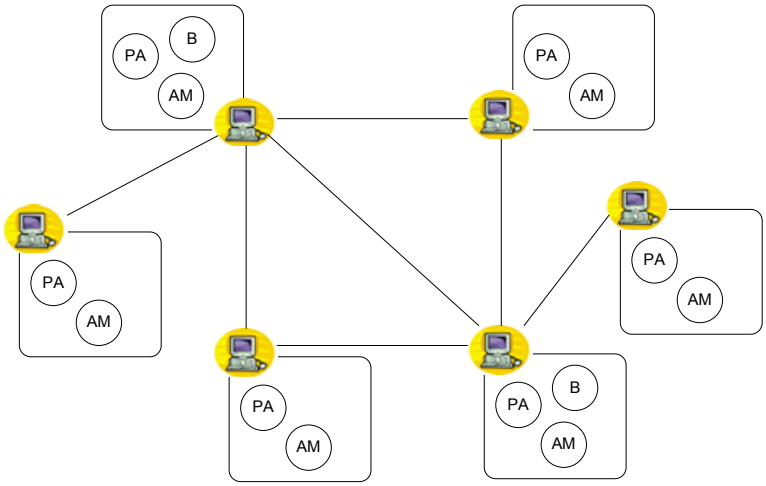


Figure 5.5: The distribution of agents within the nodes of KARe's peer-to-peer network

Note that here, the agent's internal goals have been transformed into plans, indicating that there is a specific strategy (or more precisely, an algorithm) to achieve these goals. At this point, such strategies are only stated, being completely clarified only during detailed design. For example, the AM's goal of providing p2p knowledge repository has been refined into two different plans: maintaining taxonomies and organizing knowledge artifacts. In other words, this agent allows the user to create and update their taxonomies, besides organizing their knowledge artifacts, classifying them on taxonomic concepts. In addition to plans, resources have also been included in the model. For instance, in order to maintain the taxonomies, the AM needs access to the taxonomies themselves, included here as a resource, which is further indicated as a means for the execution of the maintaining taxonomies plan.

Besides including plans and resources, the model of Fig. 5.6 shows the delegations between the system's agents, and between the user (CoPMember) and the system's agents. Three of the PA's plans require the use of knowledge artifacts, namely the request for explanation, question response and providing proactive recommendation plans. In order to acquire such resource, this agent relies on the AM. In addition to that, the PA's question response

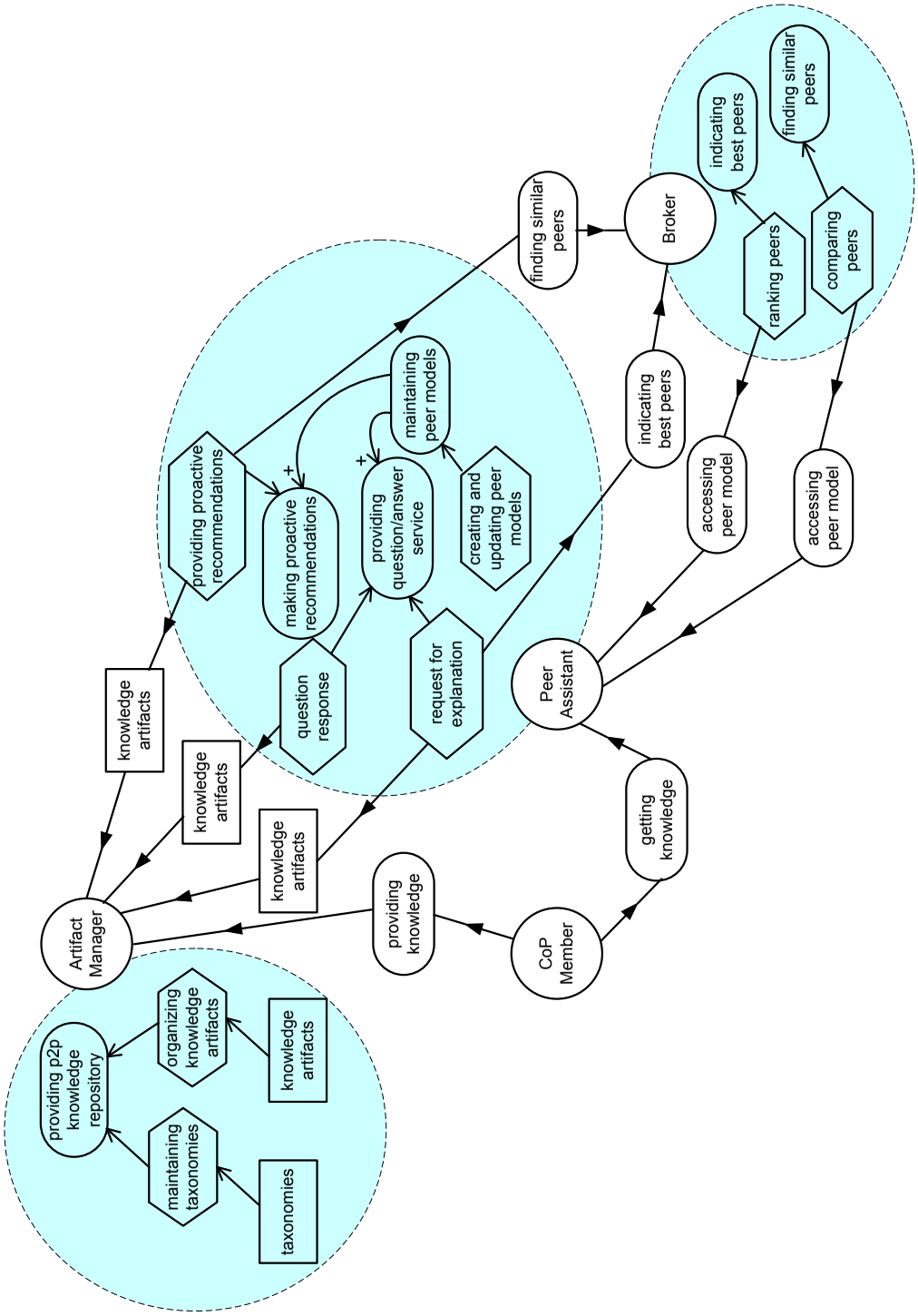


Figure 5.6: Tropos diagram showing KARe's high level architecture in more details

and providing proactive recommendation require that the Broker fulfills the goals of respective finding similar peers and indicating best peers. On the other hand, the Broker depends on the PA for accessing peer models, which contain important characteristics to enable the Broker to carry out the plans associated to the PA's delegated goals.

5.3 Detailed Design

At this point, we begin detailing our system design, starting with an understanding of the internal structure of our system (i.e. information modeling). At this point, we should then convert the Tropos diagram that details the system's architecture (Fig. 5.6) into an AORML model, following AR-KnowD's transformation rules presented in table 3.2 on chapter 3. This results in a draft AOR Agent Diagram (AD), as shown in Figure 5.7

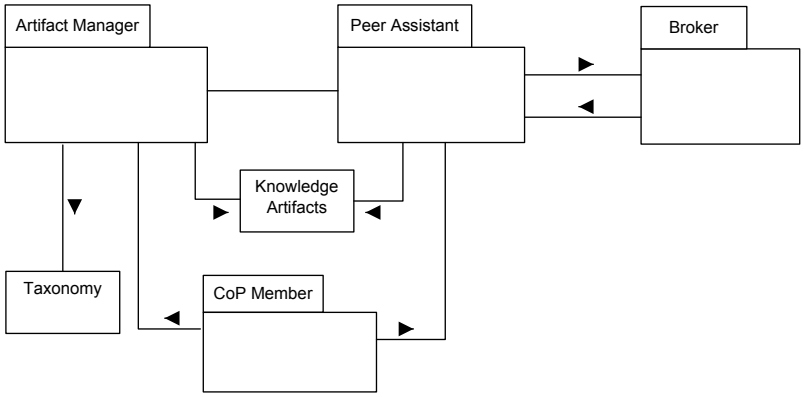


Figure 5.7: Draft Agent Diagram

The AD of Fig. 5.7 shows us the following transformations:

- Tropos *agents* become AOR *agents*;
- Tropos *resources* become AOR *objects*;
- Tropos *delegations* become AOR *association relations*.

The directions of the association relations between agents are inferred from the direction of the delegations between agents. And in case of re-

lations between agents and objects, it is assumed that the relation comes from the agent to the object (reflecting agent's activeness and object's passiveness). Note that such diagram can be obtained automatically by simply transforming the notation elements from one language to the other. Having this draft diagram at hand, the system designer is now able to refine it, which results in two distinct diagrams: a conceptual diagram and a design diagram, respectively exhibited in Figures 5.8 and 5.9.

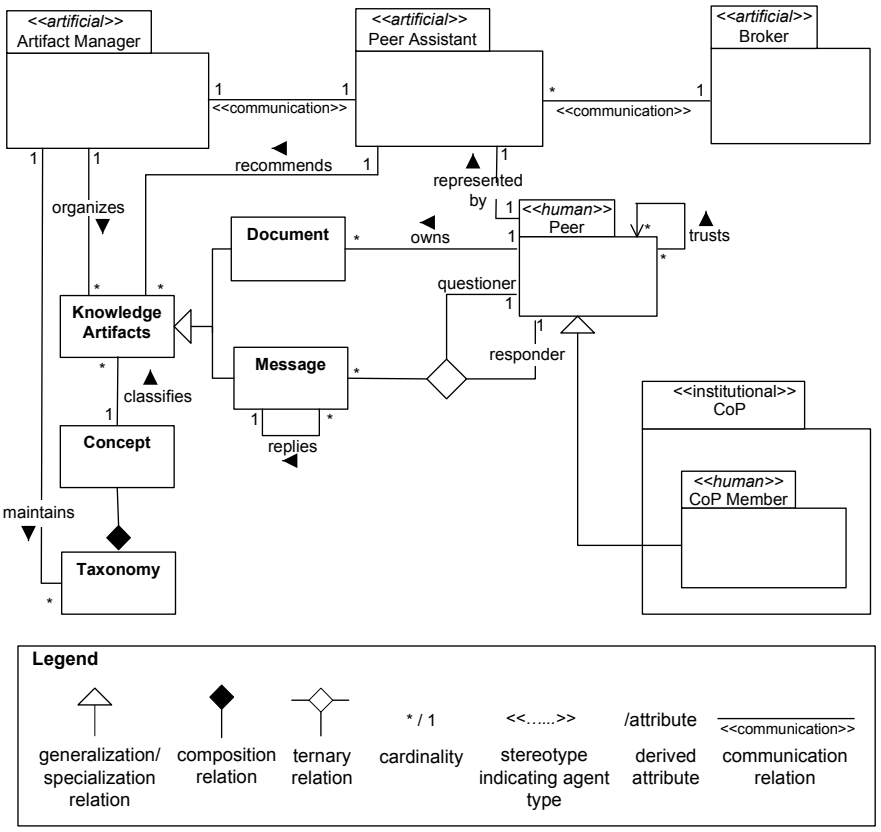


Figure 5.8: Conceptual Agent Diagram

A conceptual AOR Agent Diagram (AD) enables the understanding of the conceptual relations between the agents, being them system (artificial) or domain (human and institutional) agents. In this diagram, the type of agent is depicted using UML[2] stereotypes. In this way, we may differentiate the

[2]since AORML extends UML, it allows the designer to make use of the UML elements every time AORML does not offer an alternative solution.

artificial (ArtifactManager, PeerAssistant and Broker), human (CoPMember, Peer) and institutional (CoP) agents of the scenario. The Peer agent has been added, providing more generality to the system. Now, a Peer can be a community member or not, allowing all organizational members to participate in system's interactions, regardless of their affiliation to specific communities. Besides Peer, the CoP institutional agent has been added, to show that a CoPMember is part of a CoP [3]. The relations between agents (and between agents and objects), previously inferred from the delegations between agents in Tropos, are now named, and cardinalities have been added, giving a more clear reading of how the agents and objects relate. Agents are connected through a special kind of relation: the *communication relation*. As already mentioned in section 3.8, if such type of relation exists between two agents, it determines that these two agents communicate to achieve their goals. The details of their communication is made clear in the interaction diagrams that we present in section 5.3.1.

Some objects and relations have also been added to refine the draft AD. For example, a specialization relation has been added to show that knowledge artifact can be of one of the two types of artifacts previously described in section 4.10: document and message. The relation of these artifacts to the Peer have been added: Peer *owns* documents while *sends* or *receives* messages (ternary relation). The introduction of a concept object indicates that the taxonomy is composed of a set of concepts, and that each concept classifies a number of knowledge artifacts.

Although the conceptual model enables a clear understanding of the system to be and their relations with domain agents, the designer needs a more refined version of the conceptual AD, focusing solely on the internal structure of the system, and abstracting away from details of the domain. This is finally achieved with the diagram of Fig. 5.9.

The design model only contains elements that compose the system, while all domain agents are left out. Note that, in this diagram, a Peer has been

[3] in AORML, as in UML, composition can be shown either by the use of the composition relation (a diamond-ended relation), or by internalization. Here we chose the latter, as we think it is more clear.

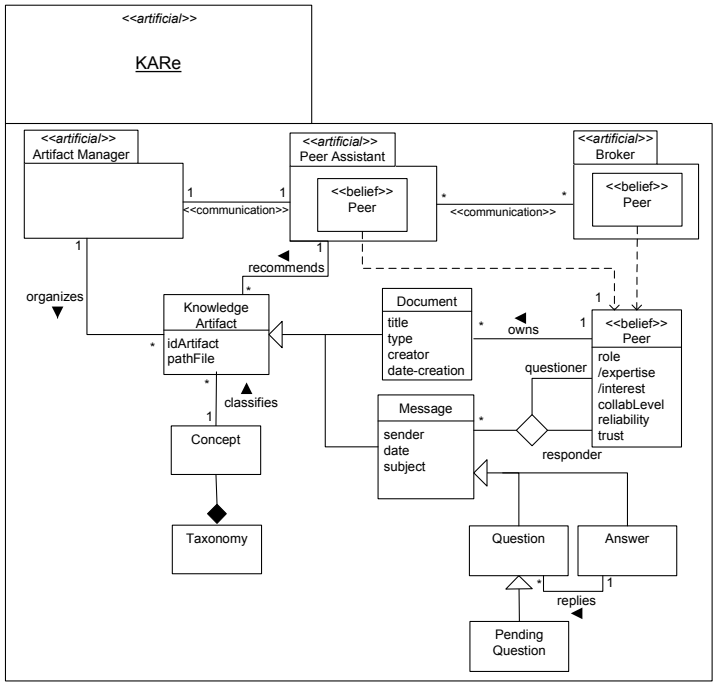


Figure 5.9: Design Agent Diagram

objectified. This is due to the fact that the Peer agent is not part of the system, but the information about him/her that is relevant for the system (i.e. the user model) should be. The Peer object models the belief that the artificial agents have about the Peer agents. In addition to this, we have specialized the Message into Question and Answer, refining it into the two types of messages we may have in the system. Besides, the Question is further refined into Pending Question. This helps us model the proactive functionality of the system to retrieve answers to questions that have not been satisfactorily answered before.

5.3.1 Behavior and Interaction Modeling

Having understood how the information is structured, the design proceeds with the elaboration of AORML interaction diagrams to model agents' internal behavior and interaction with other agents. In other words, these diagrams allow the designer to understand better the specific functioning of

the agents in response to incoming messages and/or events.

Interaction modeling generally starts with the elaboration of a few AOR Interaction Sequence Diagrams (ISDs), detailing agent's interactions to perform each of the system's functionalities. Instead of modeling general situations, ISDs depict prototypical situations, in which different possibilities regarding the same functionality may be explored. This has shown to be very useful in enabling a clear understanding of the system. Usually, the designer reasons about how functionalities are accomplished while elaborating this kind of diagram. It is common to refine them several times before a final version can be achieved. Moreover, they can also lead to modifications in the information structure, modeled with the use of ADs. After understanding how the agents interact with each other, AOR Interaction Pattern Diagrams (IPDs) allow the designer to detail the internal behavior of some agents. And finally, AOR Interaction Frame Diagrams (IFDs) abstract from modeling specific situations, showing solely the types of messages exchanged between two agents. This is helpful for clarifying the interface between two agents.

At this point, we turn back to our Tropos architectural model of Fig. 5.6, which informs us which situations we should target on interaction modeling. These situations are given by the plans adopted by each of the system agents. Plans in Tropos "represent, at an abstract level, a way of doing something" (Bresciani et al., 2004, pag. 207). In other words, a plan is an algorithm or a set of steps that lead an agent to accomplish certain goals. Here, plans function as UML use cases. AOR ISDs are able to model the set of steps that lead to a plan execution, showing how agents' interact and eventually perform actions and respond to events of the environment. According to Fig. 5.6, we should model the following plans: maintaining taxonomies, organizing knowledge artifacts, creating and updating peer models, request for explanation, question response, providing proactive recommendations, ranking peers and comparing peers.

Maintaining Taxonomies

Figure 5.10 shows the Peer interaction with the AM on the creation of part of the taxonomy of Fig. 5.2 of section 5.1.2.

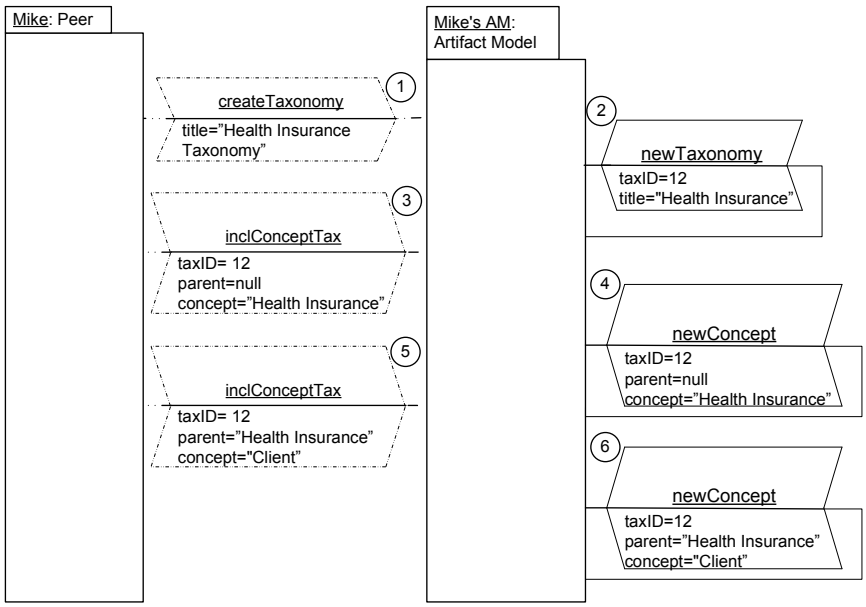


Figure 5.10: The Peer creates a personal taxonomy

The ISD of Fig. 5.10 depicts Mike creating a new taxonomy named "Health Insurance" (createTaxonomy message) and including two concepts on this new taxonomy (inclConceptTax): "Health Insurance" as the top concept, and "Client" as its child. Similar messages may be sent in order to include other concepts. On the right side of the diagram, a few actions of the AM in response to the incoming messages are depicted. The AM creates a new taxonomy (newTaxonomy action event), providing it with an associated id (taxID), which identifies the taxonomy in the system. Then, following's Mike's messages requesting the inclusion of concepts in the given taxonomy, the AM performs newConcept actions, creating the concept.

Similar ISDs can be sketched for *updating* or *deleting* concepts in a taxonomy. However, as these diagrams would be redundant with relation to the already presented diagram in Fig. 5.10, we choose not to include them here. As aforementioned, the ISDs depict prototypical situations, instead

of generalizing the interactions[4]. This supports the designer on reasoning about different possibilities before generalizations can be made (usually in the IPDs and IFDs).

Organizing Knowledge Artifacts

The AM supports the system user on organizing his/her knowledge assets in his/her local knowledge repository. An example of how a new document can be included in the user's knowledge base is presented in Figure 5.11. Interesting in this diagram is the presence of the metadata that qualifies the given document. To avoid too much repetition, we refrain ourselves from providing the ISDs for *updating metadata* or *deleting document*, as they would be very similar to the one here presented.

Mike submits a new document to be included in his local knowledge base (includeDocument message). He indicates in which taxonomy (taxID parameter) and under which concept (concept parameter) the document should be classified. Besides this, other document metadata are included as parameters: title, type, creator, and date-of-creation. And finally, the document file is also submitted. The AM includes the new document in Mike's knowledge repository by performing the newDocument action. Besides the submitted metadata, the AM creates an identifier for the submitted document (idDoc parameter).

Creating and Updating Peer Models

The PA is responsible for building the peer model. For this, it needs the user to provide some personal data and to customize some system options. Figure 5.12 shows the creation of the peer model, through interaction between the user and his PA, when he starts using KARe.

To start using KARe, the user needs to provide a few initial data to his/her PA. Fig. 5.12 shows that Mike first submits his personal data (personalData message), including name, role, room number etc. Next to this,

[4]underlying agent's names and message labels signify that these are instances instead of classes.

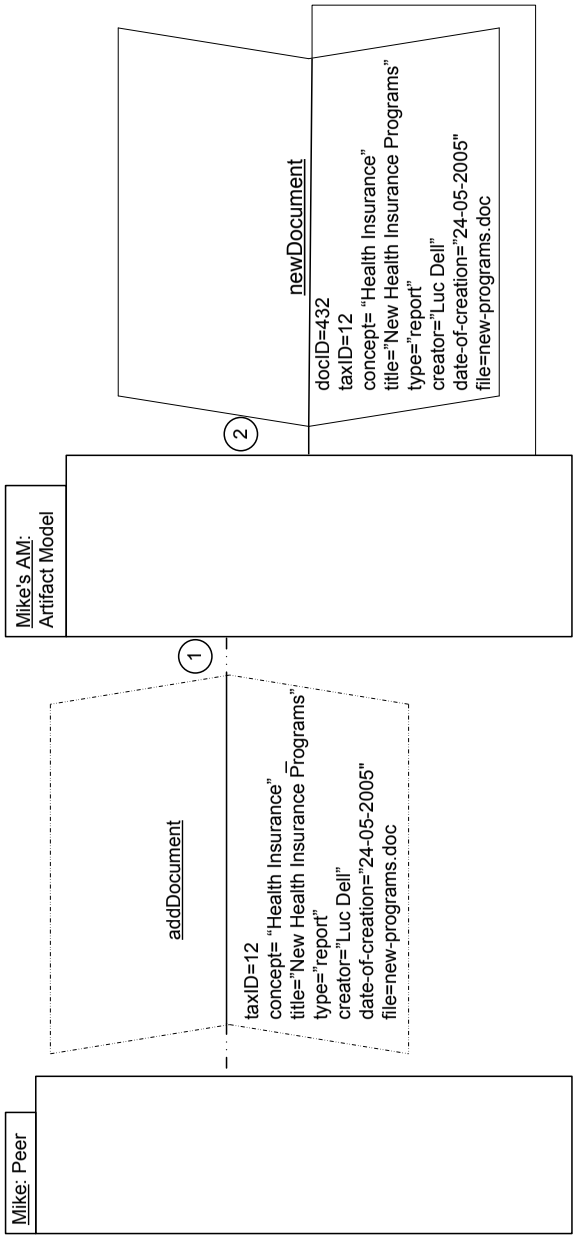


Figure 5.11: The Peer includes a new document in his personal knowledge base

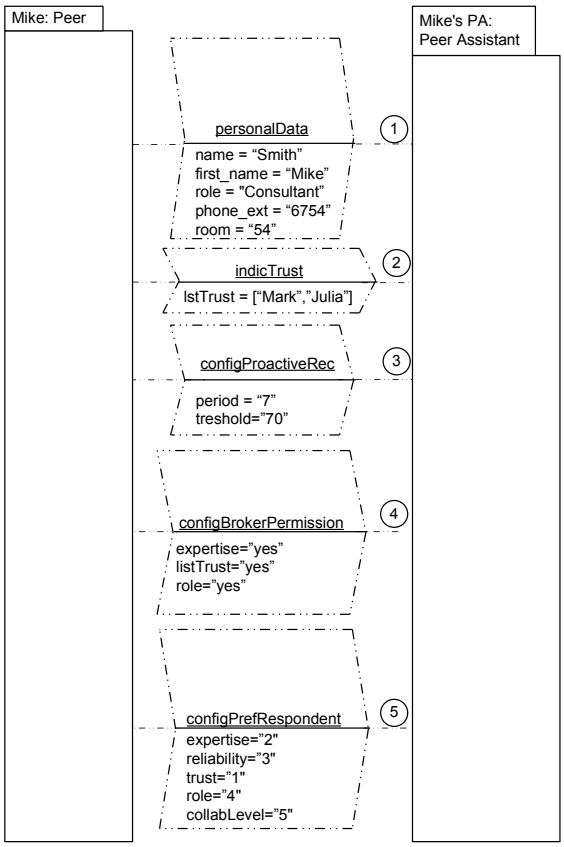


Figure 5.12: The Peer configures his personal information

Mike indicates who are his trusted colleagues (indicTrust message), in this case: Mark and Julia. Then, Mike configures some system options, namely: a) the period he would like his PA to provide him with proactive recommendations (period parameter of the configProactiveRec message), given in number of days (e.g. Mike wants to receive proactive recommendations every 7 days); b) the threshold that should be used by the PA to find similar artifacts during the proactive periodic searches (treshold parameter of the configProactiveRec message), given in percentage (e.g. here, 70%); c) permission to the Broker to access his personal information (configBrokerPermission message). In this case, Mike allows the Broker to view all his personal information, i.e. his expertise, his list of trust and his role; and d) the order of characteristics based in which the Broker should choose peers to respond to his requests (configPrefRespondent message). Here, Mike indicates he wants this choice to be based in the following order: trust, expertise, reliability, role and collaborative level.

After its initial creation, the peer model is then constantly updated, throughout Mike's interaction with other peers. This especially regards some of the *interaction features* (refer to Fig. 5.9). From these, only trust information is directly gathered from the user. The information on how reliable the user finds the other peers, and how collaborative they are in respect to the user can only be gathered throughout peer interaction (see Fig. 5.16 for examples of how reliability and collaborative level are updated in the Peer Model).

Request for Explanation

For reasons of space and clarity, the interaction modeling regarding this plan has been divided in four diagrams, depicted from Figure 5.13 to Figure 5.16. These diagrams depict the situation previously referred to in Figure 5.2 of section 5.1.2, i.e. Mike asks a question that is responded by an appropriate peer selected by the system.

The interaction starts when Mike submits a question to his associated PA (requestExplanation message), properly classificating the message in a con-

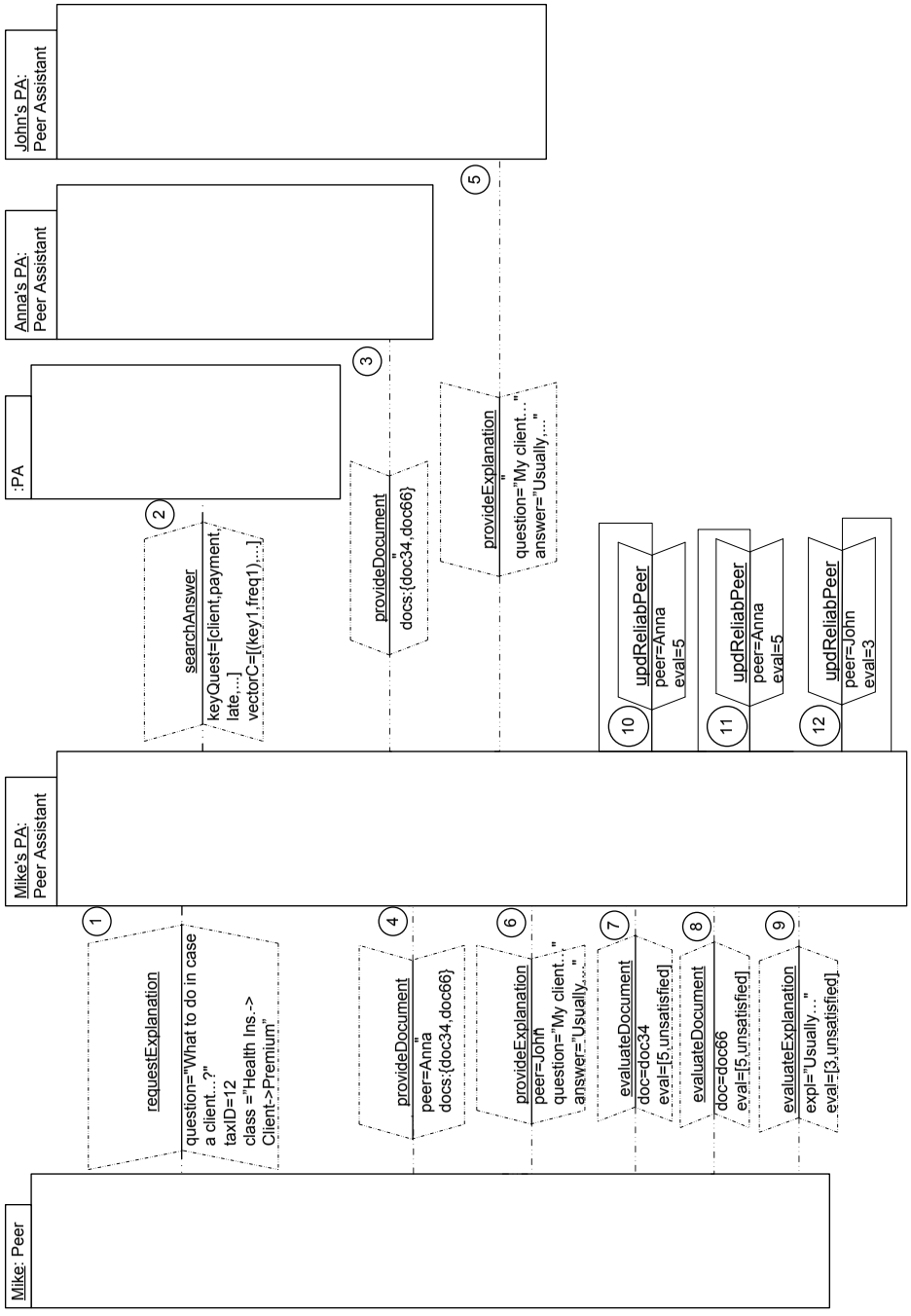


Figure 5.13: Mike submits a question to his associated PA

cept of one of his taxonomies (note the taxID and class parameters in the requestExplanation message). In response to Mike's message, Mike's PA broadcasts a message to the other PAs in the network (searchAnswer message), searching for a response to Mike's question. Following, two PAs respond to Mike's PA search request. At this point, the Peers have not been directly contacted. Instead, only their personal repositories have been searched for an appropriate answer by their associated PAs. Maria's PA submits two documents that seem related to Mike's question (provideDocument message from Mary's PA), while John's PA sends a previously question that is similar to Mike's doubt (provideExplanation message from John's PA). Having received the incoming artifacts, Mike's PA imediately forwards them to Mike, expecting to fulfill his knowledge needs (provideDocument and provideExplanation messages from Mike's PA). However, Mike is not satisfied. He submits poor ratings to all received artifacts, also indicating that he is unsatisfied with the response (note the eval parameter in the evaluateDocument and evaluateExplanation messages). These poor evaluations will trigger the interactions depicted in Figure 5.14. Before going forward, Mike's PA updates in Mike's peer model the reliability of the two Peers who sent knowledge artifacts (updReliabPeer action events). Unfortunatelly, in this case, their reliability will diminish due to Mike's unsatisfaction regarding the received artifacts.

The PA must now identify a specific peer for whom to submit Mike's question. For that, the PA relies on the Broker, for which it submits a findBestResponder message. Note that in this message, Mike's PA provides some useful information to the Broker, namely the experts list already inferred from the previous step (i.e. the experts in the question's related subject are those whose PAs sent automatic responses), authorized information from Mike's peer model, and the order of importance of the responder characteristics according to Mike (the given numbers indicates that Mike would like the responder to be chosen based on the order: trust, expertise, reliability, role and collaborative level. This information has been configured in the initial information exchange between Mike and his PA, as depicted in the diagram of Fig. 5.12). The Broker acknowledges that it has received the request from Mike's PA.

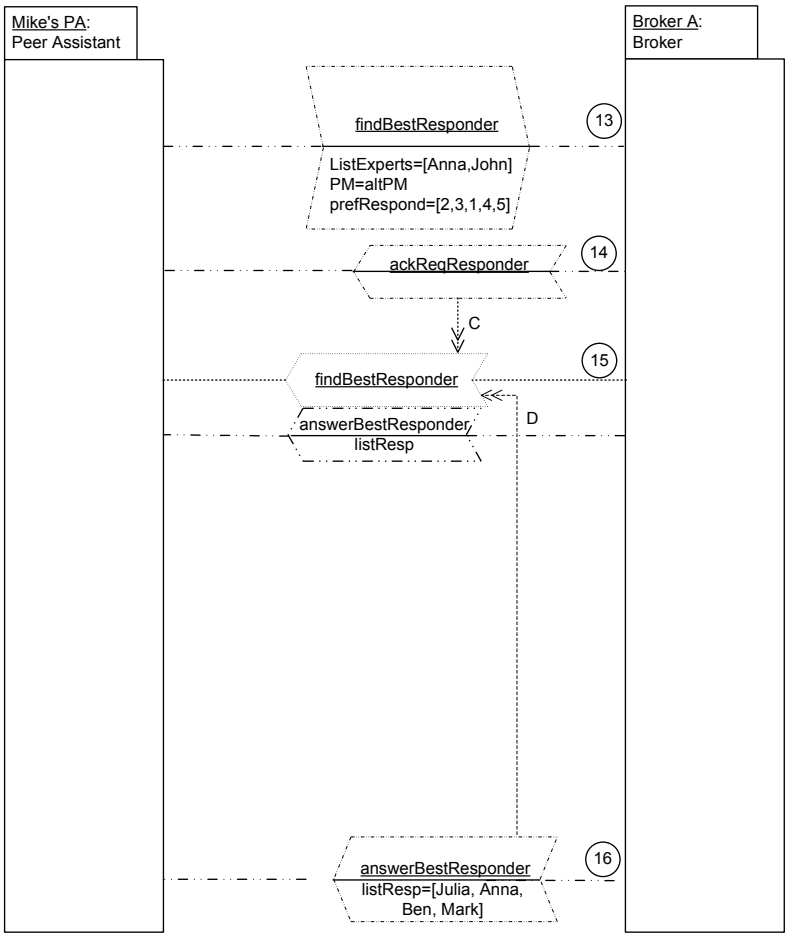


Figure 5.14: Mike's PA searches for a Peer to directly respond to Mike's doubt

At this point, a commitment is established between these two agents. Note that such commitment can be inferred from the indicating best peers delegation between the PA agent and the Broker agent in the Tropos architectural model of Fig. 5.6. As seen in section 3.5, delegation in ARKnowD indicates at the same time, dependency from the delegator (the PA) towards the delegatee (the Broker) and a commitment from the delegatee towards the delegator. The counterpart of a commitment is a claim. Thus, we can also say that the PA has a claim towards the Broker. Commitments are important deontic construct, which establish a kind of contract between two agents. As can be noted in Fig. 5.14, a commitment has one or more arguments as possible outcomes of a given commitment. In this case, the Broker is forced to submit an answerBestResponder message, containing a list of appropriate peers to answer to Mike's request. Any other response (or lack of response) would mean that the commitment was not fulfilled, which should lead to some sanction regarding the committed agent. For the designer, this indicates an important trigger for exception handling in the system. An alternative for the non-fulfillment of the commitment should be then provided in the system's code. However, in the case at hand, the Broker fulfills the given commitment, by sending the list of best peers to Mike's PA.

The diagram in Fig. 5.14 accomplishes one of the Broker's plan, namelly the ranking peers plan. The relation between the PA's request explanation plan and this plan is also clear in the Tropos diagram of Fig. 5.6. First, the delegation coming from the request explanation plan to the Broker agent generates the indicating best peers goal, which is then refined to the ranking peers plan internally to the Broker. In comparison with UML use cases, the relation between these two plans (request explanation and ranking peers) is similar to the relation between two use cases A and B, use case A 'using' use case B.

Figure 5.15 proceeds with a sequence of interactions in which Mike's PA tries to find a response to Mike's doubt by contacting one of the peers in the best peer lists submitted by the Broker.

First, Mike's PA shows him the list of peers sorted by the Broker, so that Mike can choose one or more to send his question (possibleResponders mes-

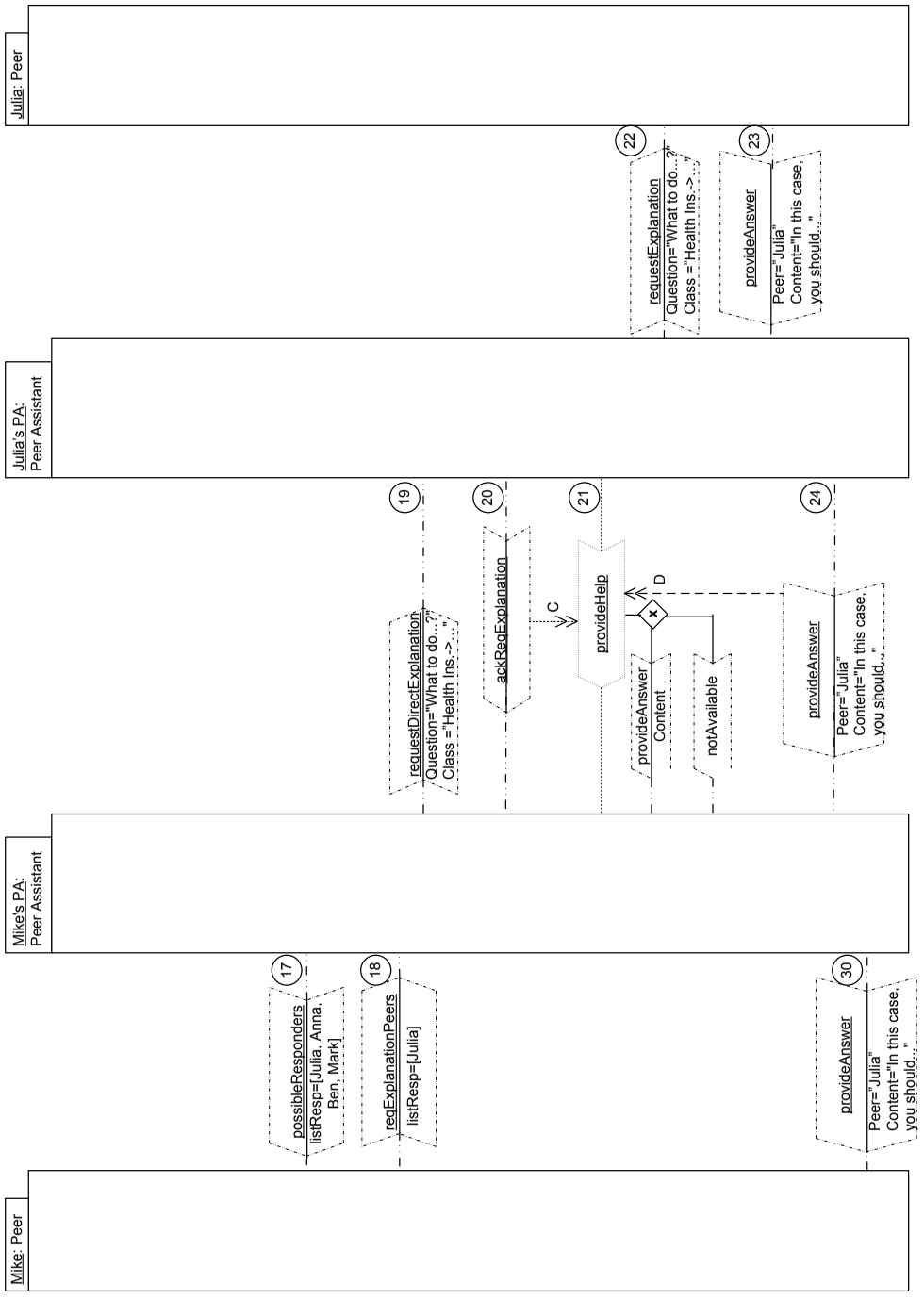


Figure 5.15: A Peer responds to Mike's question

sage). Mike selects 'Julia' from the list (requestExplanationPeers message). Then, Mike's PA submits the question to Julia's PA, so that Julia can be directly contacted for the answer (requestDirectExplanation message). At the moment an acknowledgement is sent from Julia's PA to Mike's PA, a commitment from the former towards the latter is created. This commitment is important to guarantee a positive outcome from their interaction despite its asynchrony. Julia's PA submits the question to Julia (requestExplanation message). However, as stated in this commitment, even if Julia does not respond to the question, Mike's PA receives a message (in this case, the notAvailable message). Nevertheless, in the case at hand, Julia did submit an answer (provideAnswer message from Julia), which is then forwarded by Julia's PA to Mike's PA. Finally, Mike receives the expected answer from his associated PA. Figure 5.16 presents the final message exchange to end this plan's execution.

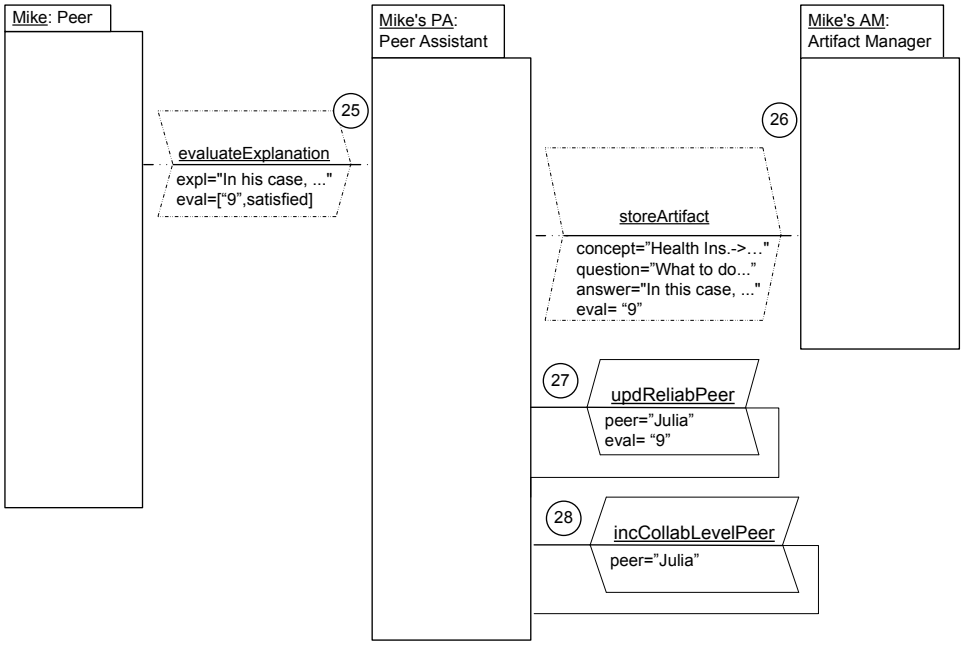


Figure 5.16: The question and answer are stored in Mike's knowledge repository

Having received an answer to his question, Mike analysis if it is satisfactory and provides his PA with an evaluation (evaluateExplanation mes-

sage). By verifying that Mike is satisfied with the provided explanation, Mike's PA submits it to Mike's AM (storeArtifact message), so that question and answer can be stored in Mike's personal repository, being from now on available for consultation by other peers with similar doubts. Note by the concept parameter in the storeArtifact message that the question and answer pair will be stored in the concept Mike selected to submit his question in the first place. Besides, the PA adjusts the Julia's reliability (updReliabPeer message), and increases her collaborative level (incCollabLevelPeer message) concerning Mike.

The reliability of a peer p_j according to the opinion of a peer p_i is calculated as the means of all evaluations provided by p_i about documents or messages submitted by p_j. Let n be the number of times p_i has evaluated an artifact from p_j, and gk the grade given by p_i to an artifact received from p_j. Then, p_j's reliability is given by a simple means calculation, as shown in Formula 5.1.

$$reliability(p_j/p_i) = \frac{\sum_{k=1}^{n} g_k}{n} \tag{5.1}$$

The collaborative level is the measure of how many direct responses a peer has given to another. Being n the number of times p_j responds to a direct contact from p_i, the formula for increasing the collaborative level of p_j in respect to p_i is given by Equation 5.2. Being m the number of times p_j does not respond to a direct contact from p_i, the formula for decreasing the collaborative level of p_j in respect to p_i is given by Equation 5.3. In other words, the initial collaborative level of all peers is 0 (zero). Throughout time, this number is updated, becoming positive or negative, according to the peer's respose. Thus if the collaborative level of p_j is increased regarding p_i (due to an incoming response from p_j to p_i), this means that 1 is added to the previous collaborative level of p_j in p_i's peer model. Contrarily, if when directly contacted, p_j does not provide any answer to p_i, p_j's collaborative level is decreased in p_i's peer model, meaning that 1 is subtracted from p_j's previous collaborative level.

$$incCollabLevel(p_j/p_i) = \sum_{k=1}^{n} 1 \tag{5.2}$$

$$decCollabLevel(p_j/p_i) = \sum_{k=1}^{m} -1 \tag{5.3}$$

Figure 5.17 exhibits the situation in which a second user (Joey) submits to his PA, a question similar to Mike's. This case has been illustrated in Fig. 5.3 of section 5.1.2. No detailed description should be necessary for the Understanding the diagram of Fig. 5.17 as it follows the same logics of the previous diagrams described for this plan.

Question Response

In order to illustrate the question response plan, we choose to use an AOR Interaction Pattern Diagram (IPD), as the ISD for this situation is too simple and the IPD provides us with more information to support the system design. The IPD presents, besides the agent interaction, the internal behavior of one of the agents. Note that this diagram is built for the general case, not presenting particularities of singular situations as the ISD, but instead covering all possible outcomes of the given interaction. Figure 5.18 presents the IPD for the question response plan.

The diagram of Fig. 5.18 depicts the PA submitting a question to the AM (searchAnswer message)[5]. The AM's reactive response to the incoming message is represented by the R1 rule, detailed in Table 5.2.

When receiving the message, the AM looks among the user's Artifact Models for a response that satisfies the incoming question. If the artifact is found, the AM forwards it to the PA (in case it is a previous question, the question and answer pair is forwarded in a provideExplanation message; contrarily, documents are submitted in provideDocument messages). In case no artifact is found, a noAvailableArtifact message is issued.

[5]the parameters of this message, namely keyQuest and vectorC refer to the approach used by KARe for artifact retrieval. More details on this approach are given in chapter 6

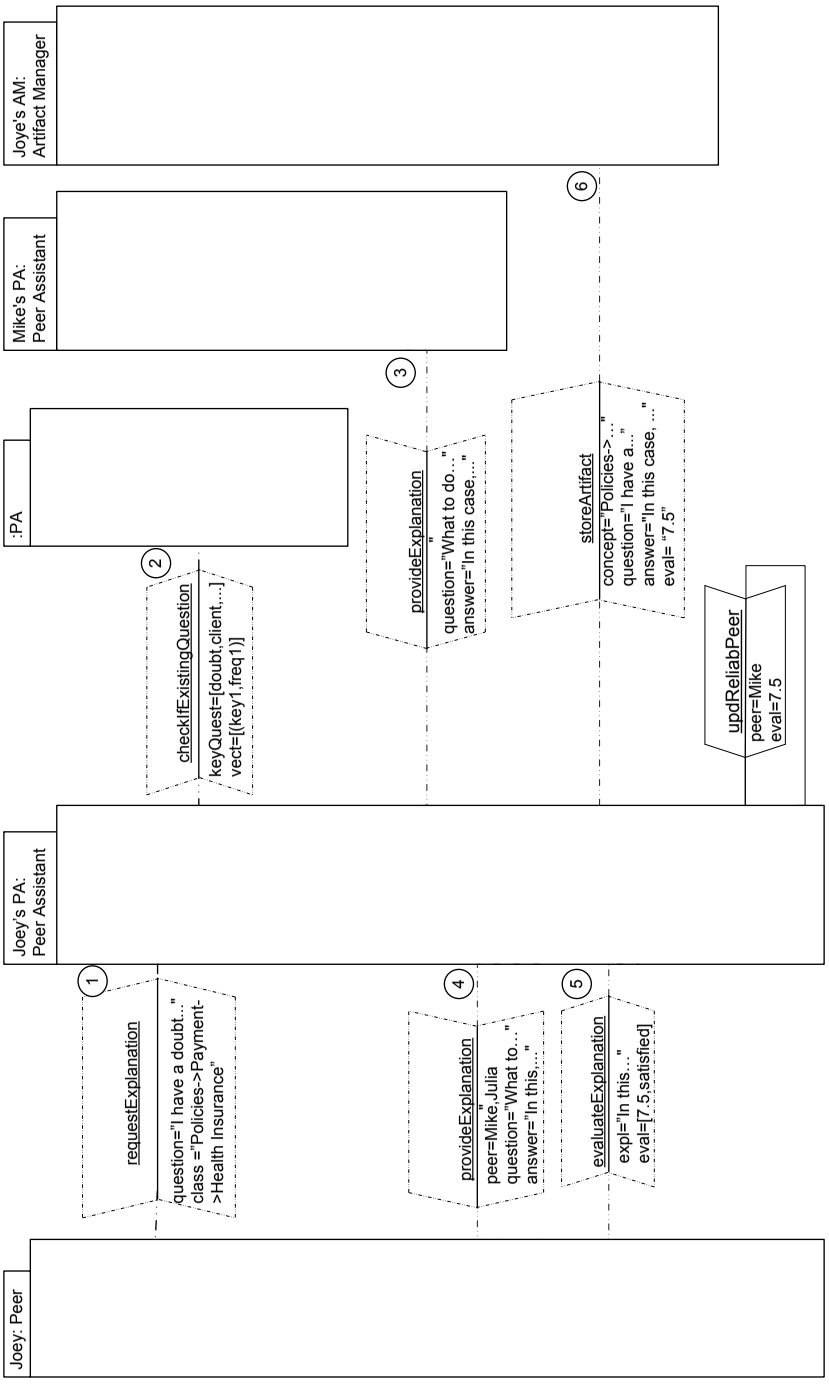


Figure 5.17: Joey's question finds a quick answer

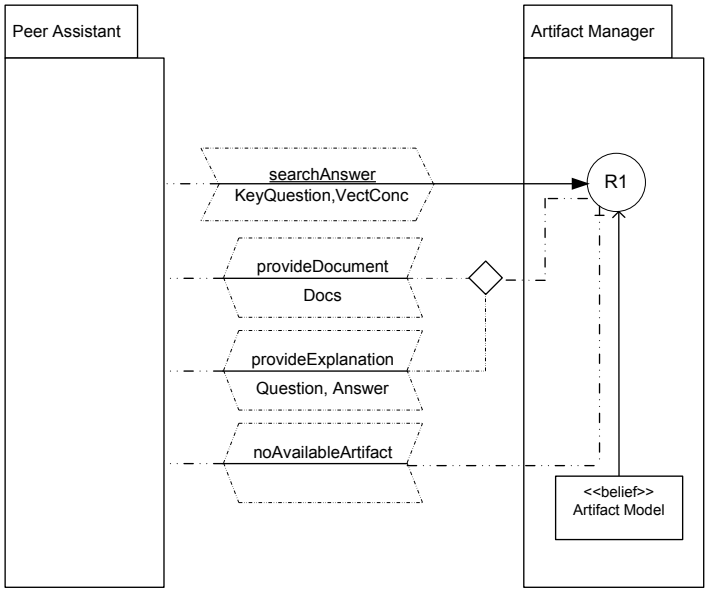


Figure 5.18: The AM's internal behavior when the answer to a question is requested by the PA

ON	Event	RECEIVE searchAnswer (?keyQuest, ?vectorC) FROM ?PeerAssistant
IF	Condition	SimilarArtifact(?keyQuest,?vectorC, ArtifactModel(?Document))
THEN	Action	SEND provideDocument(?Document) TO ?PeerAssistant
ELSE IF	Condition	Similar(?keyQuest,?vectorC, ArtifactModel(?Question))
THEN	Action	SEND provideExplanation(?Question?Answer) TO ?PeerAssistant
ELSE	Action	SEND noAvailableArtifact TO ?PeerAssistant

Table 5.2: Textual description of the rule R1 of the AM

Providing Proactive Recommendations

The Providing Proactive Recommendations Plan can actually be refined into three sub-plans, each covering one of the proactive knowledge delivery functionalities described in section 5.1.3, i.e. a) periodic search; b) solving pending requests; and c) suggesting interaction with similar users. In this section, we present one AOR Activity Diagram for each of these functionalities. This type of diagram combines the AOR IPD and the UML activity diagram, and was first introduced in (Taveter and Wagner, 2005).

The internal behavior of the PA when handling a periodic search is depicted in Figure 5.19.

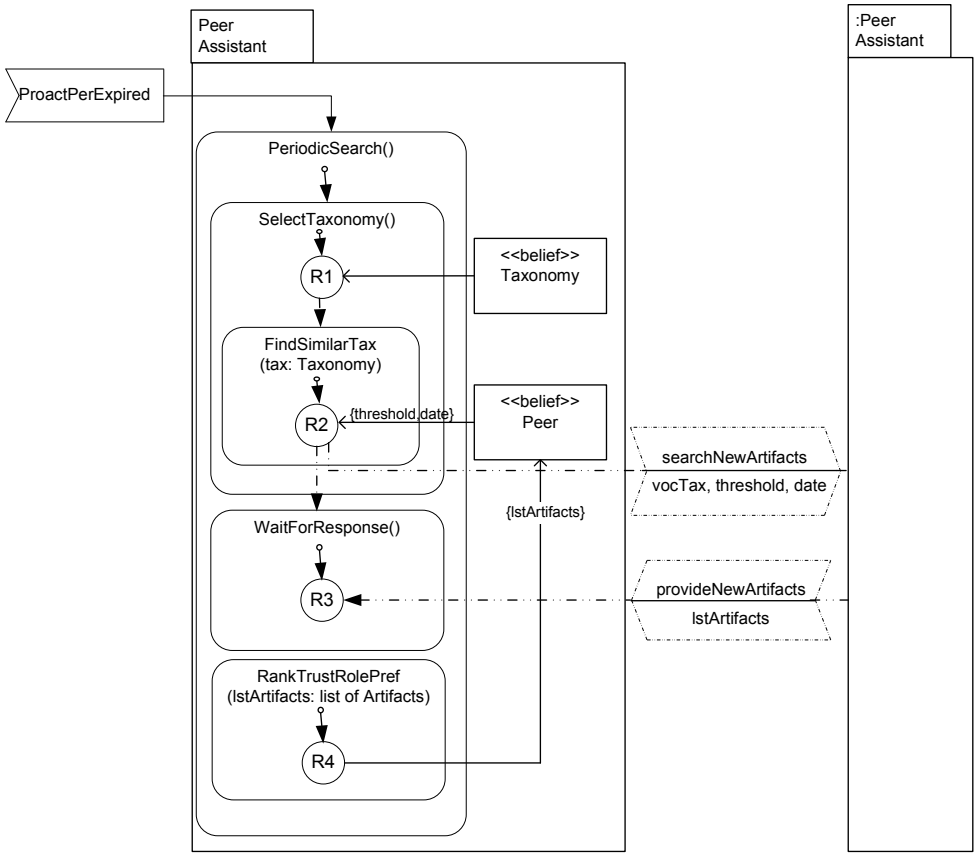


Figure 5.19: The PA periodically asks other PAs for new artifacts of interests for its peer

The PA's action is triggered by an environmental event, i.e. the period

for proactive recommendations set by the user (refer to the diagram of Fig. 5.12) has expired. The task of the PA then consists in finding new artifacts of interest for its associated peer. As mentioned in section 5.1.1, the user's interest is perceived by the taxonomies he/she maintains. Hence, for each of the user's taxonomies, the PA broadcasts to the other PAs a request for the artifacts recently created or updated, classified under similar taxonomies maintained by the remaining peers in the network (see the searchNewArtifacts message). The taxonomy is represented by a general set of keywords referred to as vocabulary (see the vocTax in the searchNewArtifacts message). The similarity between two taxonomies can be then calculated by comparing their vocabularies. The keywords composing a taxonomy's vocabulary are chosen with basis on the concepts of the taxonomy and on the artifacts (documents and messages) classified under them. Both the vocabulary creation and the similarity measure become more clear in chapter 6, where KARe's recommendation mechanism is fully described.

For controlling a satisfactory level of similarity between the taxonomies, the PA uses a percentage threshold previously configured by the peer (refer to the diagram of Fig. 5.12). This means that only taxonomies whose similarity measure is equal or superior to the selected threshold should be considered. Besides this, the PAs receiving the request only return artifacts that have been created and updated after a specified date. This date is maintained by the PA, indicating when was the last time it has conducted a periodic search. As indicated in the diagram, both date and threshold are stored in the user model (represented by the Peer belief).

After receiving the list of artifacts, the PA ranks this list according to the trust of its associated peer in other users, similar roles and preferences regarding specific artifact type or format. In other words, items coming from trusted users, users with the same role of the peer, and complying with the peer's presentation preferences score higher and are thus placed on the top of the list. The list is delivered to the peer (although not represented here for simplicity) and is available for him/her until the next periodic search.

Figure 5.20 depicts the PA internal behavior when solving a pending request on behalf of the peer it represents.

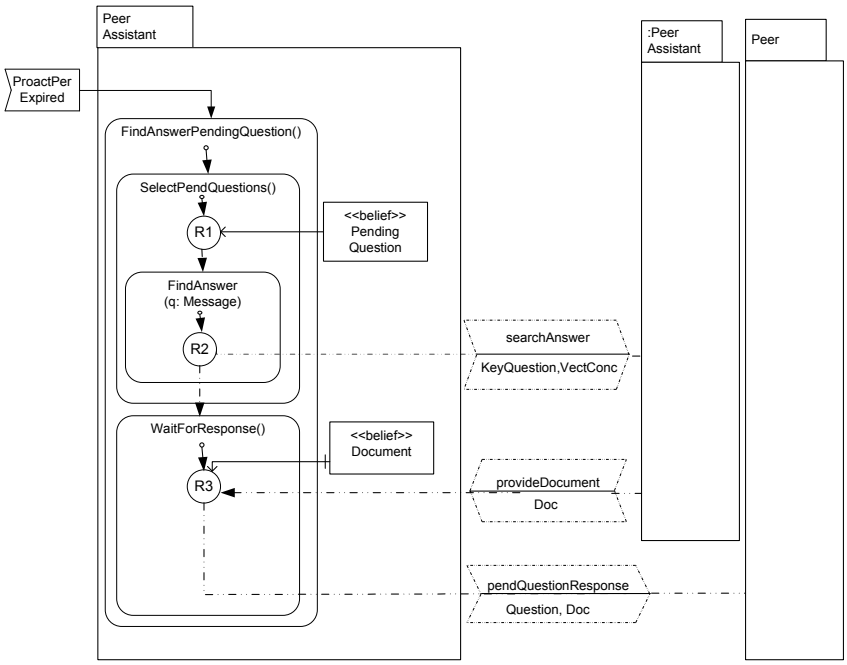


Figure 5.20: The PA looks for the answer for a pending question on behalf of its peer

Again, the PA's initiates his action after perceiving that the proactive recommendation period has expired. First, the PA looks for pending questions in the Message repository. For each pending message, the PA broadcasts a message to other PAs, searching for an appropriate answer. When receiving the message, the other PAs look for answers among the artifacts of their peers (following the logics of the diagram of Fig. 5.18), and submit the answers they find to the requesting PA. When the PA receives the incoming documents, it checks if they have already been delivered to the peer. Verifying that this is not the case (note the crossed arrow coming from the Document belief object), the PA delivers the received knowledge artifact to the Peer. Here, we just treat the case where the PA receives documents as answers, but this is a simplification, as pairs of questions/answers could also be sent. When receiving questions and answers as responses, the PA's treatment of these artifacts is analogous to the documents' treatment. Thus, we decided to supress this possibility for clarity purposes.

For suggesting similar peers to interact with its associated peer, the PA

relies on the Broker. In this case, it is more interesting to design the internal behavior of the latter than the former, as shown in Figure 5.21.

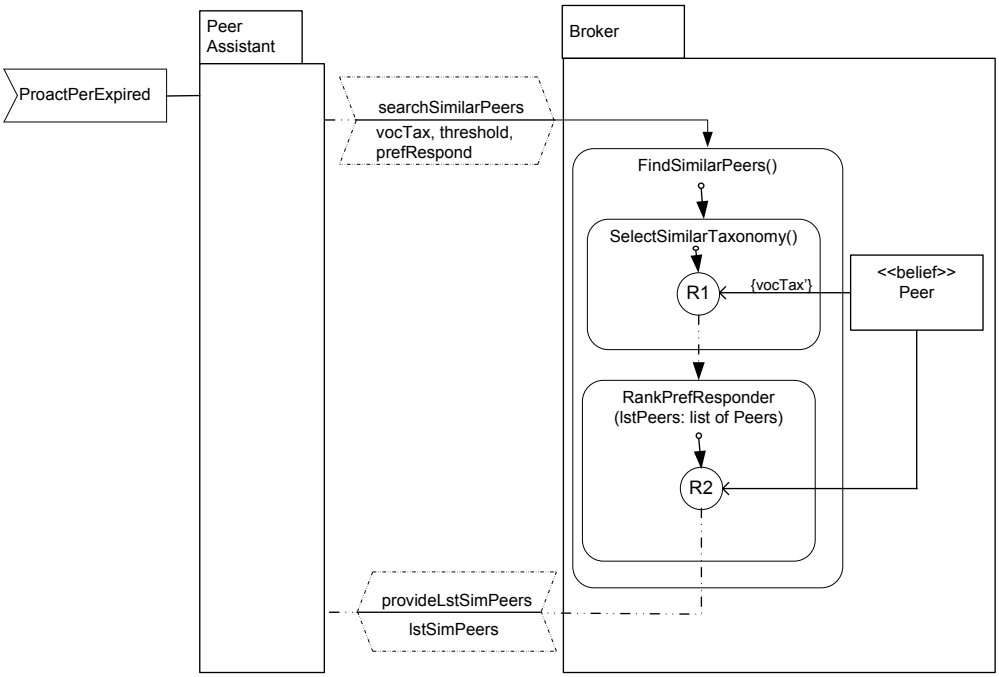


Figure 5.21: The PA searches for similar users on behalf of its associated peer

The expiration of the proactive recommendation period (ProactPerExpired event) once more triggers the PA to seek for proactive assistance on behalf of its associated peer. Then, the PA submits to the Broker a request for a recommendation regarding similar peers (see searchSimilarPeers message). As in the periodic search case, the PA does this for each taxonomy maintained by the peer. In other words, peers similarity is mainly given by the comparison of their taxonomies (representing both their interest and expertise). Later, the other user model factors are used for ranking purposes.

When receiving the PA's request, the Broker first compares the incoming taxonomy vocabulary (refer to vocTax parameter of the searchSimilarPeers message) with the vocabulary of other peers' taxonomies, whose information the Broker has been collecting in the system's idle periods (note vocTax' coming from the Peer belief, which represents the user model maintained by

the Broker). Again, the threshold earlier set by the peer is taken into account to guarantee that an adequate level of similarity is achieved. After building a list of similar peers, the Broker ranks this list according to the cognitive and social aspects present in the user model, i.e. expertise (still given by the similarity of the taxonomies), reliability, trust, role and collaborative level. The ranking is done according to the order of preferences previously established by the peer (refer to the diagram of Fig. 5.12). This is the same order used by the Broker to select responders to specific knowledge requests, exemplified in the diagram of Fig. 5.14. The ranked list of peers is finally sent to the PA, which presents it to its associated peer. Furthermore, the list is kept for user's consultation until the next search for similar peers.

5.4 Integration with Other Systems

An initiative has been launched to integrate KARe to other complementary tools, as part of the SCALE (Supporting Community Awareness, Learning, and Evolvement) research project. The main aim of SCALE is to promote the learning, development, and growth of communities of practice across small and medium enterprises through the development and integration of intelligent adaptive technology. In particular, SCALE proposes to develop a toolbox of solutions, integrating an existing system (KEEx) and two other tools under development: IVisTo and KARe. This toolbox enables mediation of knowledge exchange between organizational members by a) helping users become aware of each other and their communities, b) promoting interaction, knowledge sharing, and organizational learning, and c) facilitating the evolution of community practices.

KEEx (Bonifacio et al., 2004) allows users to share knowledge in a peer-to-peer fashion. Similarly to KARe, the users of KEEx contextualize their documents using taxonomies called contexts. Then, they are allowed to search for other documents by keyword or by context similarity. While profiting from this already available functionalities, KARe can complement KEEx by granting users with the possibility of asking and answering questions, imi-

tating the natural processes community members use to share knowledge. Moreover, KARe adds proactivity to the system, by suggesting knowledge items to users, besides identifying similar peers with whom the user may be interested in interacting with (refer to sections 5.1.2 and 5.1.3 for a detailed description of these functionalities).

IVisTo comes to enhance the visualization methods provided both by KEEx and KARe. Both in KEEx and in KARe, the result set is presented as an ordered set of knowledge artifacts, ranked by similarity concerning the user query. IVisTo provides more a sophisticated way of visualizing this result, by raising users' awareness of the types of knowledge communities that exist in the distributed network. More specifically, this tool allows the visualization of KARe's user model features (see section 5.1.1), such as trustability, organizational role, reliability, collaborative level and availability are visualized in dynamic social networks. The user is able to manipulate such networks, choosing how to view and access knowledge artifacts. Figure 5.22 illustrates how KARe, KEEx and IVisTo may be integrated, showing a snapshot of the recommendation screen.

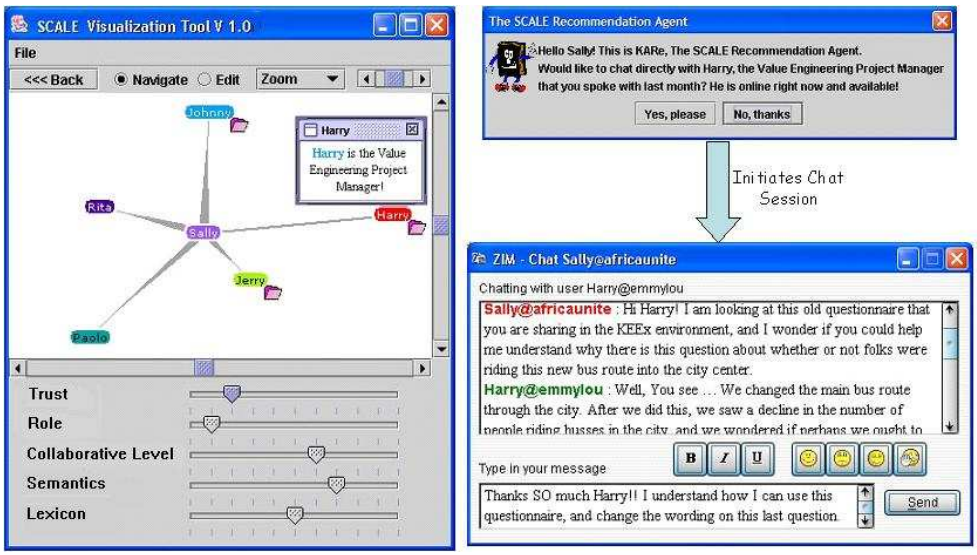


Figure 5.22: SCALE toolbox

The left side of Figure 5.22 shows a prototype of the IVisTo tool. The tool takes into consideration both the social community-oriented informa-

tion, and the more traditional and accessible lexical and semantic similarity information provided by KEEx. IVisTo displays a weighted combination of social networks, where each social network addresses a different user model variable, and the weights are given by the user's preferences. The bottom half of IVisTo window contains a set of slider bars representing the social variables in KARe's user model and the lexical and semantic attributes given by the lexical and context matching algorithms of the KEEx knowledge management platform. Using these slider bars, the user can indicate the importance, or weight, of each variable. Behind the scenes, the system generates a social network for each of the user model variables, and then computes one single network by calculating a weighted sum of the individual networks. For example, Sally's visualizations show her in the center of the screen, and the peers that returned knowledge artifacts as the result of a query in the periphery. Finally, the length of the links between her and her peers suggest the degree of similarity between her and her peers according to each user model variable. In the case of "Role", the length of the links suggests the degree to which Sally holds a similar role as each of her peers. In this way, IVisTo can provide each user access to a personalized view of the knowledge society, weighted according to his knowledge and interests. A KARe window is shown on the right side of Fig. 5.22, illustrating how KARe may proactively recommend a user to contact another peer with which to share knowledge.

Figure 5.23 presents the SCALE integration model, which defines the interactions between the SCALE Visualization Tool, the KARe multi-agent component and the KEEx knowledge management environment. A new component named the **User Model Engine** is added to facilitate this integration. This design decision is based on the fact that all three tools may work separately, although benefits are achieved by using them as one integrated system.

Although KEEx has its own interface, the SCALE system provides a new and intelligent interface, given by IVisTo. Fig. 5.23 shows that the recommendations provided by KARe are also available through this interface (*User Recommendation* arrow). KEEx output (lexical and semantic information)

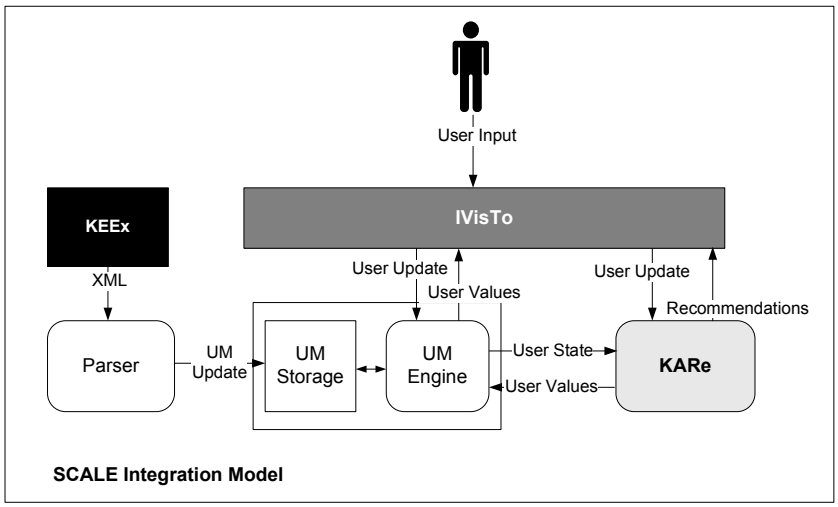


Figure 5.23: SCALE integration model

are presented in the form of an XML file, which is parsed to the User Model Engine (XML and UM update arrows). The UM Engine plays a central role in the integration model. It is responsible for guaranteeing the consistency of the user model information both to VisTo and to KARe, which updates the user model based on the ongoing interactions among the other peers.

5.5 Related Work

In order to analyze the work related to KARe, we elaborated a model that summarizes the main elements of KM systems. A KM system must provide knowledge to the right person at the right time. Although each system presents different architecture and functionalities, an abstract structure can be created to enable the analysis of their common elements. In general, KM systems integrate knowledge artifacts, disseminating them through the knowledge community. In order to accomplish that, they must: a) have access to the semantic of the content of the available knowledge artifacts; and b) identify the user's needs and preferences. The four layers illustrated in Figure 5.24 provide an abstraction for reasoning about these two general requirements.

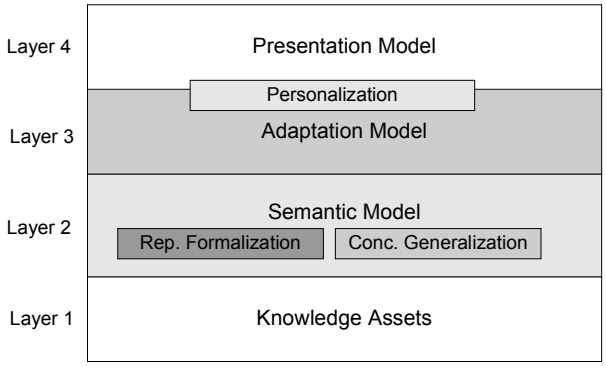


Figure 5.24: A layered view of KARe

The layers in Fig. 5.24 should be understood as different set of services that need to be provided by a KM system. These services are disposed in layers to indicate that the superior layers rely on functionalities provided by the inferior ones. Layer 1 represents the knowledge assets exchanged by the knowledge community. As earlier described in section 4.10, knowledge artifacts may be classified into **documents** and **messages**. The difference between documents and messages is important because they have different purposes within the community. While a document is used by the community members to learn about a particular procedure or topic, messages are typically used for communication purposes (for instance, to inform something, to clarify doubts and to debate particular issues). Messages can be regarded as important resources for the disambiguation of tacit knowledge. As already pointed out, much of one's knowledge is not registered in any kind of physical artifact, but rather confined in one's mind. Nevertheless, Nonaka and Takeuchi (1995) have informed us that intuitions, feelings and tacit ideas can be socialized between community members, through direct communication (refer to chapter 2 for a discussion on socialization).

The Semantic Model of layer 2 is responsible for the reasoning of the system about the content of knowledge artifacts, addressing the problem early mentioned in a). Thus, the Semantic Model is responsible for the two tasks described by Fischer and Ostwald (2001) as related to knowledge integration in KMSs: *representational formalization*, i.e. putting information in an appropriate computational syntax so that the system can access and interpret

it; and *conceptual generalization*, i.e. providing domain specific semantics to each knowledge artifact.

The problem early presented in b) is addressed by Layers 3 and 4: the Adaptation Model and the Presentation Model, respectively. The Adaptation Model is specifically concerned with 'what' is going to be presented to the users, while the Presentation Model determines 'how' such knowledge will be presented. Both layers deal with Personalization, i.e. consider user's needs and preferences, through their user models to support knowledge delivery.

5.5.1 Materializing the Semantic Model

Representation Formalization can be generally achieved by providing some metadata about the stored knowledge artifacts. Metadata (such as title, owner, author, date, etc.) enable a systematic organization of knowledge artifacts in a way that important information about them may be easily captured by the system. Important here is how to decide which metadata should be considered. There are several ongoing initiatives related to the definition of metadata specific for certain knowledge fields. One of these initiatives is the EDUTELLA project Nejdl et al. (2002), which aims at providing a peer-to-peer networking infrastructure to support the exchange of educational material. In order to accomplish this, peers can make their documents available in the network, specifying metadata information as a set of RDF statements.

Several researches adopt the Dublin Core Initiative [6] as the initial or complete set of metadata. However, as mentioned in (Davies et al., 2003a), some metadata might be specific for a given community. This way, it is advisable to keep this option customizable, allowing the community members to negotiate and create new relevant metadata for their particular purposes. A common way to organize this information is the use of a database to centralize and efficiently recover metadata about knowledge artifacts. Alternatively, metadata languages may be applied, such as XML or RDF, the

[6] http://dublincore.org/

latter presenting more reasoning power the the former (Davies et al., 2003b).

In order to accomplish concept generalization, artifacts may be classified according to a domain conceptualization. Although taxonomies and ontologies have been used in the past, the Semantic Web has recently increased the interest on using such conceptual models to explicitate the semantics regarding knowledge artifacts (Davies et al., 2003b). A taxonomy is somewhat more restrict than an ontology as it mainly focuses on hierarchical relations between concepts, usually depicted in a tree structure. Figure 5.25 illustrates the difference between ontologies and taxonomies. Regarding concepts these two methods of conceptualization are equivalent. However, as shown in the figure, ontologies admit more complex types of relations than taxonomies.

Several applications in the Semantic Web rely on ontologies to support KM. Reimer et al. (2003) for example, builds an ontology of skills to support the development of a catalog of workers. This catalog can support users on finding experts on specific subjects, tasks and skills. Another system is OntoShare (Davies et al., 2003a), which creates a content management system used by organization's members to classify and share knowledge artifacts based on a shared ontology. OntoShare users annotate documents using RDF and classify them according to an ontology. Besides delivering knowledge reactively, by matching the incoming documents with the user model, the system is able to proactively retrieve knowledge to the user. Differently than KARe, however, this system adopts a client-server approach, and presupposes the existence of a shared conceptualization (ontology) among the members of the community.

KEEx (Bonifacio et al., 2004), on the other hand, apply taxonomies (referred to as contexts) as conceptual frameworks to mediate the exchange of knowledge between peers. Another adept of taxonomies is Hyperwave[7], which classifies the resources according to taxonomies that can be then consulted by the system users. KARe shares similar vision with KEEx, since it is also based on the Distributed Knowledge Management approach (DKM) (Bonifacio and Bouquet, 2002). KEEx allows each individual or community

[7]Knowledge Management with the Hyperwave eKnowledge Infrastructure, whitepaper available at http://www.hyperwave.com/

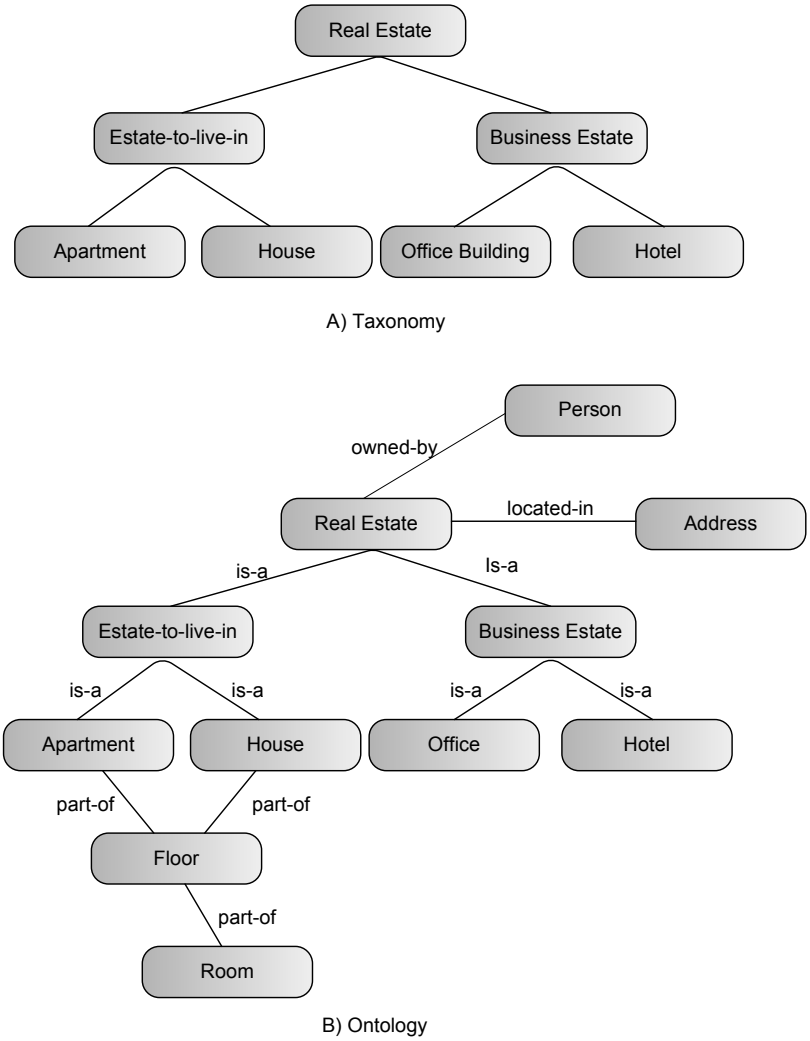


Figure 5.25: Exerpts of a real estate A) taxonomy and B) ontology

of users to build their own knowledge space within a network of autonomous peers. Each peer can make documents locally available, along with their context. When searching documents from other peers, a set of protocols of meaning negotiation (Bouquet et al., 2003) are used to achieve semantic coordination between the different representations (contexts) of each peer. KEEx is specifically concerned with the exchange of documents and does not address peer collaboration through the exchange of messages, which is one of the targets of KARe. In this sense, as described in section 5.4, KARe adds functionality over KEEx, supporting the natural social processes of asking and answering questions. As a result, KARe adjusts better into current organizational practices, providing the opportunity for organizational members to solve problems and doubts in collaboration with their workmates.

The choice of using taxonomies instead of ontologies is motivated by the DKM philosophy, which defends that rather than sharing an unique conceptualization, each organizational member has his own view of his/her work domain. Thus, both in KEEx and in KARe, each user builds his own conceptual model. As ontologies are generally considered too complex and time consuming to be built, we consider taxonomies as a more realistic model for the common user to create. In a sense, many workers already create directory classifications of this kind, both for physical or digital file systems. SWAP (Fensel et al., 2003) also adopts the DKM strategy, relying in what they term "lightweight" ontologies. The authors have not so far clarified, however, the nature of these lightweight ontologies, and how they relate both to the more general concept of ontology and to taxonomies.

We do acknowledge that several organizations are today investing on the construction of ontologies to describe their activities and domains of expertise (Reimer et al., 2003) (Gangemi et al., 2003) (Gruninger et al., 2000). Seeking to profit from these efforts, an organization ontology can be used as an initial conceptual model for KARe's users. The users may them choose parts of this bigger conceptualization to classify their artifacts, based on their own interests and targeted areas.

5.5.2 Supporting the Adaptation and Presentation Layers

The Adaptation and Presentation Models are responsible for providing the system with the means to retrieve knowledge according to the user's particular needs, interests and preferences. In order to accomplish that, these layers are commonly built around a user model, as the one described in section 5.1.1. In this section, we describe how other systems model their users in comparison with KARe[8].

The **Adaptation Model** is concerned with 'what' piece of knowledge is needed in particular situations. Different characteristics may be explored to combine knowledge artifact's content and user's need. For example, two of the most common features used to classify different users is *interest* and *expertise*. Interest is commonly used as a basis for the system to provide new knowledge artifacts to the user, while expertise is the main source for referrals to specific users as knowledge providers. Similarly to KARe, IVisTo (Soller et al., 2004) and Ontoshare (Davies et al., 2003a) classify the user's interest and expertise based on the concepts they choose to classify their belonging artifacts. In MARS (Yu and Singh, 2002), such information comes as a vector of keywords, created and updated with basis on questions and answers exchanged by the users.

I-Help (Bull et al., 2001) gathers information of interest and expertise from topics provided by the users, but also using data mining in records gathered throughout user's interaction with system tools. I-Help has recently evolved towards the peer-to-peer model (Vassileva, 2002).A student needing help can request it through his/her agent, which finds other students who are currently online and have expertise in the area related to the question. As in KARe, there is a centralized matchmaker service, which maintains models of the users competences and matches them to the help-requests. On the other hand, this system does not support management and sharing

[8]besides systems specifically tailored for KM, we also considered three e-learning systems (AdELE, Elena, and I-Help) in our classification. This is justified by the fact that they are representative of systems aimed at flexibly delivering knowledge to the users (learners, in these cases), possessing elements of all the layers of our model

of documents, restricting itself to messages exchange.

Regarding *reliability*, i.e. the level of expertise of a provider regarding a particular topic, MARS adopts a similar approach in comparison to KARe. This system also locally updates the reliability of the providers (i.e. on the seeker's user model), based on the seeker's feedback regarding the incoming response to a particular query.

Concerning *trust*, three other tools support an approach similar to the one adopted in KARe, i.e. through a "list of friends": IVisTo and I-Help. Besides positive trust indication, I-Help provides the additional possibility of indication of a non-trust list.

Once more, IVisTo shares with KARe the *collaborative level* approach, calculating it based on direct feedback given by a knowledge provider to a knowledge seeker. In MARS, this feature, referred to as sociability, is calculated by the ability of an agent to refer to others with valuable information. MARS calculates and updates this value based on user feedback.

FRODO (van Elst et al., 2001) and IVisTo (Soller et al., 2004) also consider *roles* as one of the main determinants of a specific knowledge need. As in KARe, in these two other systems, the user's roles are indicated by the users themselves.

The physical context of the user may also be an important subsidy for the Adaptation Model. For example, specific knowledge artifacts may be of help when users are executing a given task. Both KnowMore (Abecker et al., 2000) and FRODO (van Elst et al., 2001) monitor the user's *current task*, by integrating an workflow management system as part of their KM solution. FRODO provides an interesting approach of classifying tasks according to a task ontology, and then indicating the information needs required by specific tasks. In this way, the system is able to retrieve supporting knowledge artifacts from the available ones in the organizational memory. AdELE (Garcia-Barrios et al., 2004) monitors the user's reading task, tracking his or her eye-movement to produce fine-grained data about user reading behavior. The collected information is used to adjust the user model regarding interest and expertise of the user on particular topics, thus supporting the system

on suggesting new content.

Besides the Adaptation Model, the **Presentation Model** is also supported by the user model. Rather than focusing on 'what' knowledge to present, this model is concerned with the way knowledge should be presented. Hence, the Presentation Model is focused on format rather than on content. As described in section 5.1.1, KARe provides knowledge to the users, based on their *presentation preferences* and on their *physical context* (time and location). Similarly, AdELE (Garcia-Barrios et al., 2004) also bases its choice on presentation format using contextual information gathered with eye tracking technology (e.g. delivering more images/tables for a user that has problems with large and complicated texts).

5.5.3 Using the Layered Model to Classify KM Systems

The discussion regarding KARe's related work is summarized in Table 5.3. This table illustrates how to use the provided layered model to classify different KM systems. This allows us to discuss their different functionalities and solutions in light of the semantic, adaptation and presentation models described before. We can note from the comparison presented in the table, which systems provide more or less support for each layer of the model. For instance, Hyperwave, KRAFT and Ontoshare provide no support to the adaptation and presentation layers, concentrating only on the semantic model. MARS, on the other hand, places greater strength on the adaptation model, presenting limited support to the semantic layer, and no support to the presentation layer. Understanding these differentiations may assist the determination of which supporting system to use in a particular case. The table shows that KARe covers well all three models, thus providing a comprehensive range of functionality for its users.

5.6 Conclusions

This chapter has proposed and designed the KARe system. KARe aims at supporting Constructivist KM, by flexibly imitating the social processes that lead to knowledge sharing through question and answering. The system fulfills the requirements elicited in chapter 4, providing an appropriate means for newcomers and old-timers of Communities of Practice to share knowledge. For that, KARe adopts a peer-to-peer infrastructure, granting users with more control over their knowledge items, providing information about experts in specific topics, and creating a dynamic environment for collaboration and knowledge exchange. In addition to that, the system reflects a layered model, which considers both the need for the syntactic and semantic representation of knowledge artifacts, and the benefits of adapting these artifacts to user's personal and cognitive characteristic, and presentation preferences.

For the design of KARe, we applied ARKnowD, successfully verifying the usefulness of our methodology to support this development activity. ARKnowD supports the analyst and designer to go all the way from domain analysis to system architectural and detailed design. The designer starts with a Tropos model of the system's architecture and slowly moves towards the detailed design, where AORML has been applied. The transformation method previously defined in section 3.7.3 was applied in this chapter, converting from Tropos to AORML notation. This transformation facilitates the understanding of the system model, and help the designer to trace back the system's functionalities to the requirements elicited during the previous analysis activity. For instance, the AM's organizing knowledge artifacts and maintaining taxonomies plans, respectively modeled by the AOR ISDs of Figs. 5.11 and 5.10 accomplish the AM's providing p2p knowledge repository goal. This goal has been generated by the allowing peers to keep control of their assets, delegated by KARe to the AM. And finally, tracing it back to the domain analysis presented in chapter 4, this goal has been previously delegated by the CoP to the KARe system (refer to the diagram on Fig. 4.7).

Systems	Layer 1: KA		Layer 2: SM		Layer 3: AM							Layer 4: PM	
	Doc	Msg	Metadt	Ont/Tax	Inter.	Expert.	Reliab.	Trust	Role	Col.Lev.	Context	Pref	Context
AdELE	x		x	x		x	x					x	x
Elena	x		x	x	x	x	x						
FRODO	x		x	x	x	x	x		x		x		
Hyperwave	x	x	x	x									
I-Help	x	x	x	x	x	x	x	x		x			
IVisTo	x		x	x	x			x	x	x			
KARe	x	x	x	x	x	x	x	x	x	x	x	x	x
KEEx	x		x	x									
KnowMore	x	x	x								x		
KRAFT	x		x	x									
MARS		x			x	x	x			x			
OntoShare	x		x	x									

Table 5.3: Classifying KM systems classified according to the layered model

Throughout KARe's design, we exemplified all major constructs and modeling techniques of the AOR external model (Wagner, 2003). From Figs. 5.7 to 5.9 exemplify AOR's information modeling diagram, namely the Agent Diagram (AD). These figures show all AD's constructs, such as the different types of AOR agents (human, institutional and artificial), and several types of relations (e.g. specialization, composition, association, ternary relation and communication), most of them borrowed from UML. These diagrams have shown different views from the system: while Fig. 5.8 focus on a conceptual view of the system, Fig. 5.9 presents a design view, depicting only the elements of the actual system and abstracting away from entities of the domain. AOR interaction modeling diagrams are illustrated from Fig. 5.10 to Fig. 5.20. All diagrams present message passing between agents, and some (as for example, Fig. 5.10 and Fig. 5.13) also illustrate agent's non-communicative actions. The use of commitment (and claim) is exemplified in Figs. 5.14 and 5.15, showing the strength of this construct to enable exception handling, and especially the control of asynchronous communication between agents when humans are involved (5.15). Finally, Figs. 5.18 to 5.21 show how AORML deals with behavior modeling, with the use of reaction rules.

The design of KARe at this point is still platform-independent, thus consisting in the MDA PIM described in section 3.7.1. The next chapter adjusts this PIM for a specific platform, namely the Java Agent Development Framework (JADE). Other platforms could be chosen, for example the system could have been implemented in pure Java, or using the Jxta framework[9], specifically targeting the development of peer-to-peer systems. In this thesis, however, we opted for keeping the agent-oriented nature of the system from analysis to implementation. For implementation of agent-oriented systems, JADE offers a good solution, in which agents adopt particular behaviors, communicate through message exchange, and are supported by a common communication ontology.

[9]http://www.jxta.org/

Chapter 6

Recommendation Algorithm and Implementations

"Nothing becomes real until
it is experienced."
John Keats

In this chapter, we present the core algorithm underlying KARe, i.e. the algorithm that handles the questioning-answering process. Such algorithm is based on an Information Retrieval technique that models knowledge artifacts using their most important keywords. Besides, such algorithm considers information from the taxonomic structures used to classify these artifacts to find a suitable answer to an incoming knowledge request. Section 6.2 presents an introductory overview on Information Retrieval research, describing some techniques that are useful for contextualizing our work. Following, section 6.3 describes KARe's recommendation algorithm in detail, including its evaluation.

A second objective of this chapter is to present the remaining of the detailed design of KARe, enabling its implementation with the JADE framework. By doing this, we hope to demonstrate that the ARKnowD methodology enables the analyst and designer to consistently go from the domain analysis (described in chapter 4) to system design, initiated in chapter 5 and completed here. This final design stage (presented in section 6.4) leads us

through KARe's implementation using the JADE framework.

Further in this chapter, section 6.5 describes the developed KARe prototypes, providing their general idea and implementation details. Finally, related work can be found in section 6.6, and section 6.7 concludes this chapter.

6.1 Introduction

The principle underlying KARe is to support KM by imitating the natural processes of social interaction, allowing organizational members to ask and answer questions. As a consequence of this conceptual choice, the core of the system consists in mediating the question and answering process. An important part of handling this process comprises the automatic recommendation of existing answers to users' questions. This answer may be present in the form of documents or responses to previously similar questions, both classified under the taxonomies maintained by each system peer. This chapter describes an algorithm developed to support KARe's recommendations.

The presently described algorithm is based on *Information Retrieval (IR)* techniques. Information Retrieval is an established but constantly evolving area of research that deals with all processes related to accessing relevant information in large collections of documents (Baeza-Yates and Ribeiro-Neto, 1999) (Salton and McGill, 1983). Thus, research in this field targets some of the most pressing challenges of the information society, aiding people to effectively handle information overload. The relevance of information retrieval for KM support becomes obvious, since KM systems are highly based on the storage of explicit knowledge to be later retrieved according to the situation at hand.

According to Baeza-Yates and Ribeiro-Neto (1999, pg. 3), *"the effective retrieval of relevant information is directly affected both by the user task and by the logical view of the documents adopted by the retrieval system"*. Logical view of the documents refers to how information items are represented by the system. A popular technique is representing documents as a set of keywords

automatically extracted from it, or assigned to it by a specialist. KARe's peer-to-peer nature conditions the retrieval algorithm both regarding the user task and the information item's logical view. The knowledge items stored by a peer are not viewed as a flat collection of documents. Instead, the set of documents are structured by a taxonomy which classifies each of these items under a concept of the tree.

As described in chapter 5, KARe users' begin by classifying knowledge artifacts (such as work documents and the like) using their own taxonomies, much in line with the use of a structured file system. In other words, each artifact is classified in a node of the taxonomy, analogous to storing files in folders of structured file system. The choice of using taxonomies to classify knowledge artifacts provide the system peers with a contextualized view of knowledge artifacts, as already described in section 5.1.2. However, this is just part of the reason why this choice has been made. Another strong claim we make is that such information may be helpful in aiding our recommender agent to automatically find knowledge on behalf of the system users.

In a traditional IR system, items are equally distributed in the document collection, which should be completely searched when a retrieval request is issued by the user. In KARe, however, taxonomies are used to classify documents. Consequently, the system is able to search for the answer only considering particular nodes of the taxonomy where the answer is probably located. Besides diminishing computational complexity, this approach allows the system to profit from user knowledge, previously encoded in personal taxonomies, to retrieve knowledge more precisely.

The search process is triggered when a user asks a question, which he/she first assigns to a concept (node) in his/her taxonomy. Hence, a question (or knowledge request) is logically represented not only by the keywords it contains but also by the keywords representing the concept which classifies it. For finding an appropriate answer, KARe must first match two distinguishing taxonomies, analogously to (Avesani et al., 2005) and (Bouquet et al., 2003). More precisely, when receiving a knowledge request from the *questioner*, the system must find in the *responders'* taxonomies which concepts are more likely to contain artifacts that satisfy this request, subse-

quently retrieving it. The concept representation is obtained by considering its position in the taxonomy, and by the documents classified under it.

Although our assumption about the gains of applying taxonomies seems reasonable, it can only be proved by testing our algorithm using real datasets. Thus, besides describing the applied techniques, this chapter also presents empirical data to validate them.

6.2 Information Retrieval

Information Retrieval relates to the representation, storage, organization and access to information items (Salton and McGill, 1983). Research in this area has initially been motivated by the growth of traditional libraries. However, the use of information technology in such libraries, along with the emergence of digital libraries have given new strength to this research field.

Information is usually embedded in a *document*, which is defined as: *"a single unit of information, typically text in digital form, but it can also include other media. It can be a complete logical unit, like a research article, a book or a manual. It can also be part of a larger text, such as a paragraph or a sequence of paragraphs. A document can be any physical unit, for example a file, an email, or a web page."* (Baeza-Yates and Ribeiro-Neto, 1999, pg. 142). In this sense, document as defined here comprehends both the concept of **document** and **message** previously defined in section 4.10. From now on, we use the term 'document' having this more general meaning.

In traditional IR systems, we can distinguish three phases that compose the information retrieval process:

- *Analysis and Indexing*: each new document is analyzed and appropriately described by a set of index terms. After being suitably classified, the document is incorporated in the existing information collection.

- *Querying*: a request is formulated according to established proceedings, aiming at satisfying the user's information needs.

- *Retrieving and Presentation*: a retrieval mechanism is used to find and present the available documents that may be of interest to the user.

Besides these three main phases, some pre-processing operations are often executed to prepare the document to be indexed and searched. Thus, IR systems (IRSs) usually follow a general structure, illustrated in figure 6.1. The main components of an IRS are: 1) the user interface, where the user is allowed to enter his *query*; 2) the text pre-processor and 3) the indexer, which together accomplishes the *analysis and indexing* phase; and finally, 4) the searching mechanism, responsible for handling the query and performing the last stage of *retrieving* and orderly *presenting* items. The following sub-sections discuss three of the main components of an IRS: the text pre-processor (section 6.2.1), the indexer (section 6.2.2) and the searcher (section 6.2.3). Further, we summarize the classic IR modeling approaches (section 6.2.4) and the main evaluation mechanisms for IRSs (section 6.2.5).

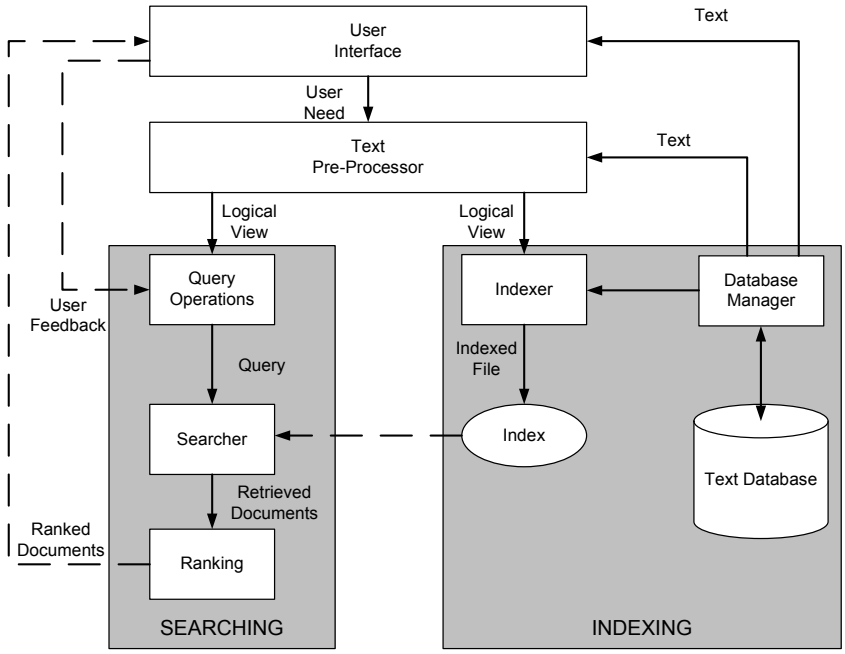


Figure 6.1: Our view on the information retrieval systems' general architecture

6.2.1 Text Pre-processing

In order to create the vocabulary of a collection (i.e. the most representative terms of the collection), we must select all the index-terms that are relevant to represent our collection and include them in the vocabulary index. This implies some text operations to select the index terms. Mostly noun terms are selected because these usually carry the semantics in a sentence. Text pre-processing is thus a necessary step to improve the performance of any IRS. The most common operations performed over text are the *lexical analysis*, *elimination of stopwords* and *stemming*.

Lexical Analysis

The objective of the lexical analysis is to treat digits, hyphens, punctuation marks and the case of letters. This leads to the identification of individual words in the text, so that with the words in hands, one can disregard the ones that are not good index terms for searching (such as numbers). This procedure often includes the removal of punctuation and hyphenation, and sometimes even the case of the letters.

Elimination of Stopwords

Stopwords are terms such as articles, conjunctions, adverbs and prepositions, which are generally too common in all documents. They are usually filtered out because they are useless for information retrieval purposes, since they do not differentiate documents. In other words, these terms frequently appear in all documents, independently of the document's content. Thus, they are not considered good index terms. An advantage of the elimination of stopwords is that it reduces the size of the index, so the system can be faster when performing the search.

Stemming

Stemming is the process of reducing a word to a common root, i.e. it reduces word variants to a common concept. This could be achieved by removing

prefixes and suffixes to a "stem". So, if we have only stems instead of words (and their variants) in our index, the search can be more precise and faster. For example, the words 'swimming', 'swimmer' and 'swim' should all be represented by the same stem 'swim'. There are several types of stemming such as table lookup, successor variety, and affix removals.

6.2.2 Indexing

The indexing activity consists in attributing a set of terms that identify the content of a new document to be included in the collection. This activity is considered by many researcher as the most difficult of all, due to the complexity of finding the ideal set of terms to represent each document (Salton and McGill, 1983).

Indexing is redundant if the collection is small, or if it allows search in the entire text of each document each time a query is made. In practice, this solution was very costly, however, today, this is a common approach used by Web-based search engine. However, still in our days, instead of using a full text search, many IRSs use data structures representing the collection index. This is an appropriate choice when searching in static or semi-static collections. Only if the IRS is used for a dynamic collection, one could think of combining the two approaches, i.e. a small database for searching the new documents (while they have not yet been indexed) and a large index for searching the old ones.

Most IRSs use the inverted file index data structure. Figure 6.2 illustrates this method. The index terms consist of a set of words selected during the text pre-processing. Next to each index term, there is a list of tuples in the format (*Document;Occurrences*), which respectively identify a document where a given index term appears, and how many times it occurs in such a document. In this way, for each word on the database, we can directly retrieve the documents where it is located, knowing also how many times it has appeared in each document.

To select which words should be part of the index (an operation commonly known as *feature selection*), the most well-known approach is based on the

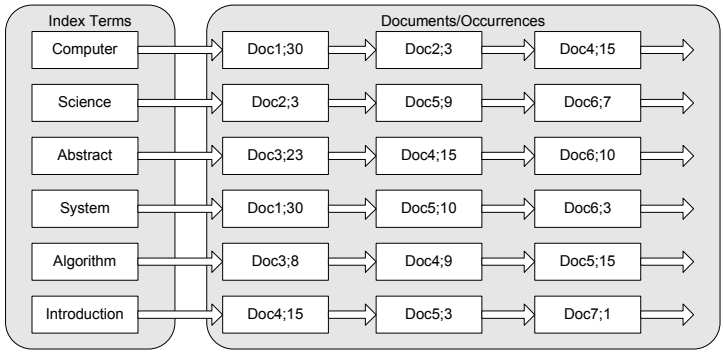


Figure 6.2: Illustrating the inverted documents index

frequency of the words in the text of each document. This technique is based on the calculation of the frequency of all words in a document, organizing them in descending order according to the frequency. Then, a superior and an inferior threshold are determined, and the words of middle frequency are chosen as the index terms. The most frequent words are eliminated because they usually are stopwords, while the less frequent terms are left out for not being representative of the content of that particular document.

6.2.3 Searching

As can be noted in Fig. 6.1, searching occurs in three steps: 1) querying; 2) the search itself and 3) ranking of the results.

Querying refers to the stage when the user expresses his/her information need, through a system request. The most common querying approaches are the use of keywords, sometimes combined with boolean operators: AND, OR, and NOT, which respectively indicate conjunction, disjunction and exclusion of query terms (Baeza-Yates and Ribeiro-Neto, 1999). For providing a more intuitive interface for the common user, some search engines substitute AND and OR operators for statements such as "including all words" and "including one or more words" respectively (see for instance, the advanced search of Google [1] and Altavista [2])

[1] http://www.google.com/advaced_search?hl=pt-BR
[2] http://www.altavista.com/web/adv

The *searching* and *ranking* mechanisms depend on the kind of information retrieval model the system was built upon. For example, the algorithms that apply boolean search do not use any ranking mechanism, since this type of search may only indicate if the document satisfies or not a given query. Besides the boolean model, another common IR model is the vector model. In this model, both the indexing terms and the query terms receive weights. These weights are then used to compute the similarity between the documents and the user query. In this way, the system also considers documents that partially satisfy a query, providing them a lower grade than to those that completely satisfy the query. Section 6.2.4 describes IR modeling in detail.

It is important to note that the retrieval may result in an empty result set. This is due to the difficulty in generating a perfect combination between index and query terms, since the choice for both is uncertain. The indexers usually apply a rule of specificity to choose index terms, while query users tend to use more general terms. This is one of the biggest limitations of IRSs, minimized by the adoption of one of proposed query refinement techniques (Chen and Dhar, 1989). In general, these techniques strongly rely on interaction with the user, who refines the query based on his experience and on the ranking mechanism.

A way of reducing the effort of the user on manipulating the result set, also helping him on formulating new queries, is to contextualize the documents of the result set according to the user query. A popular technique is the use of *metadata* beside the title of the retrieved documents (e.g. date, source, file size and abstract), providing extra information for the user concerning the retrieved documents. Other techniques include *highlighting* in the text of the retrieved documents the query terms, or providing a *KWIC (keyword-in-context)*, which is a summary of the document, extracting from it a few sentences in which the query terms occur.

6.2.4 Modeling

As mentioned in chapter 3, a model is an abstract representation of a portion of the real world. With such representation one is able to simulate problems and solve them by analysis. Thus, regarding IR Models, we understand that on one hand, there is information representation and on the other hand, there are the different retrieving mechanism.

More specifically, "information representation" refers to the logical view of a collection of documents and the queries that form the input of a system. The approach to retrieve the information would be a framework that models the documents and queries into its logical forms, and a ranking function that orders the document set according to queries. Thus, as Baeza-Yates and Ribeiro-Neto (1999) mention, the IR models may be represented as a quadruple D, Q, F, R(q_i, d_j) where:

- D is the document set. The documents are represented as the elements of the set. D : d_1, d_2, . . . , d_n;
- Q is the user query;
- F is the framework to model documents and queries into a logical form;
- R(q_i, d_j) is a function to rank documents according to a particular query.

There are different ways to approach the modeling "framework" and the "ranking function". This results in different IR models, such as: the boolean model, the probabilistic model and the vector model. In this work, we apply the vector model, thus describing it in detail. The other two approaches are only briefly described.

Boolean Model

The boolean model is the most intuitive one among the three. The queries are represented by boolean expressions (composed of keywords that are combined by the boolean operators: AND, OR, and NOT). Documents are then

classified as being relevant or not, based on this query. Thus, this approach does not consider documents that only partially match a query, since boolean expressions are not flexible. In systems that apply this kind of model, the creation of queries is usually simple, but generally leads to large response document sets with lots of irrelevant documents. In order to find more relevant documents, the user must know more about the information to be extracted. In this sense, the user first makes a broader query and then, refines this query by examining the result set. This is usually an iterative process and its success depends on the user's experience on the topic.

Probabilistic Model

This model attempts to capture the IR problem within a probabilistic framework. It assumes that an "ideal answer set" exists for a given query, containing exactly all relevant documents to answer this query. The problem then consists in determining which properties (characterized by index terms) describe this ideal answer set. As these properties are initially unknown, the functioning of this model directly depends on user interaction. A probabilistic system first generates guesses on the probability of the documents' relevance given a particular query. Then, the user is requested to evaluate the retrieved document set, informing the system about his/her evaluation regarding the relevance of each document. The system subsequently uses this information to refine the query. By repeating this process several times, it is expected that the initial description will evolve to a closer description of the ideal answer set, thus providing better results.

Vector Model

In the IR vector model, documents and queries are treated as real algebraic vectors where the dimension of the vectors is determined by the dimension of the vocabulary (i.e. the vectors size is given by the size of the vocabulary index) (Baeza-Yates and Ribeiro-Neto, 1999). Therefore, once the vocabulary has been determined (i.e. the text is pre-processed, determining the index-terms), all documents are represented by vectors. Each dimension of

the vectors is calculated based on the frequency of each index term in each document itself. Having this vectorial representation, it is possible to calculate the similarity between couples of documents or between a document and a query.

In figure 6.3, the depicted vectors are the abstraction of a query (Q) and any particular document (d from a set of documents D), and the angle θ indicates how close these vectors are, thus indicating the similarity between the document and the query. There are several measures for vectors similarity, and one of the most well-known of them is the cosine of the angle θ formed by the vectors. Due to the popularity of this approach, in this work, the cosine measure is also applied. As a consequence of being a cosine, the result of the similarity function varies from 0 to 1 in an ascending order of vectors similarity. Equation 6.1 describes a *cosine similarity* between vectors.

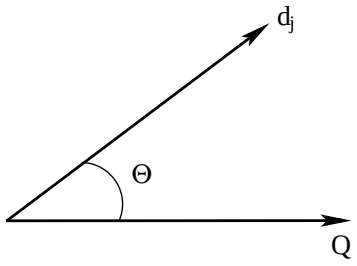


Figure 6.3: The cosine function is used to compute the similarity between a query Q and a document dj

$$similarity(\vec{d_j}, \vec{Q}) = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,Q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,Q}^2}} \tag{6.1}$$

Here, $w_{i,j}$ is the weight of the index terms of document $\vec{dj}$ and $w_{i,Q}$ is the weight of the index terms of the query $\vec{Q}$.

Each dimension value of the vectors is computed based on the frequency of the index term in question on the document collection. There are several ways to compute the dimension's weight and this process is called index term weighting. For this work, we chose the $TF * IDF$ method (see Equation 6.2). This method computes the weight in two steps. First, we calculate the

term frequency (TF, represented by fk) of a particular index term in each document. Secondly, we calculate the amount of documents that contain the term wk, i.e. inverse document frequency (IDF, represented by $\log \frac{N}{n_k}$). In this equation, N represents the total amount of documents in the collection, while nk characterizes the number of documents containing index term wk. The aim of using both TF and IDF in the weight calculation is on one hand, to increase the weight if a term is very popular in a document and on the other hand, to penalize the weight (i.e. to decrease its value) if the term is present amongst many documents.

$$w_k = f_k * \log \frac{N}{n_k} \tag{6.2}$$

6.2.5 Evaluation

There is no absolute measure that evaluates how good an IR system or algorithm is. Usually, algorithms are measured according to their *precision* and *recall* and compared with each other. To find a unique number to be compared, the F1 measure is often used. This measure is an average of the precision and recall results. It is important to remember that to compare two systems, one must use the same set of documents and queries.

Precision

Precision is the relevance measure to the searcher of the items that are retrieved, i.e. if a search returns ten documents of which nine are very relevant, that search has high precision. This measure is more important when the users of the system prefer an approach that retrieves a specific document very fast.

$$precision = \frac{NumberRetrievedRelevantDocuments}{NumberRetrievedDocuments} \tag{6.3}$$

The precision, as can be noted by equation 6.3, measures the ability of the system to refuse the non-relevant documents. In other words, it is a measure of the *correctness* of the system's retrieval approach.

Recall

Recall is the proportion of relevant information that is retrieved by the search, i.e. if a search only retrieves one hundred relevant documents out of three thousand that are available (and relevant), that search has low recall. On the other hand, if it retrieves all the available documents on the topic of the search, it has high recall. This measure is, thus, dependent on the documents of the collection. This measure is more important when the users of the system prefer an approach that retrieves the highest number of documents around a specific topic, but are not concerned on finding one specific document quickly.

$$recall = \frac{NumberRetrievedRelevantDocuments}{TotalNumberRelevantDocuments} \tag{6.4}$$

As we can see on equation 6.4, the recall measures the ability of the system to retrieve relevant documents. Thus, recall refers to the *completeness* of the result set according to all relevant documents contained in the collection.

F1 measure

There are several measures that try to combine the value of precision and recall, referred as the "F measures", as they are known as F1, F2, and so on. The most well known of them is F1, which is calculated based on equation 6.5

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{6.5}$$

6.3 Recommendation Algorithm

In this section, we present KARe's recommendation algorithm. As explained in section 5.1.2, a user contextualizes his question according to his/her taxonomy, while the answer is searched in the taxonomies of the remaining peers in the network. Section 6.3.1 describes how the algorithm accomplishes this,

and section 6.3.2 presents the result of an experiment carried out in order to evaluate the algorithm's performance.

6.3.1 Description

For finding relevant documents in a collection, the standard IR vector model approach computes the similarity between the user query and all the documents in the given collection, selecting the most similar vectors as the winner documents (Baeza-Yates and Ribeiro-Neto, 1999). However, this approach disregards any knowledge that users may have about the structure and concepts of the artifacts being searched. This can lead to increased noise on the search results. For instance, trying to search for the word "agents" in Google [3] results in documents about several different kinds of agents (e.g. chemical agents, software agents, real state agents and travel agents).

In KARe, the user classifies his/her documents according to a personal taxonomy. In this way, similar documents are grouped by the user under the same concept in the taxonomy tree. In addition to that, before submitting the question, the user contextualizes the query, assigning it to a specific concept in the taxonomy. By doing this, the user gives to the system an extra hint on the query's content. Aiming at reducing the noise of the search, our algorithm exploits the taxonomic information supplied by the user to determine the region of the search space where the required information is more likely to be found. Besides providing more accurate results, this approach also reduces the computational complexity of the algorithm in comparison with the standard approach. This happens due to the fact that the standard approach needs to search the whole documents collection (complete search space) for an answer. Conversely, following our algorithm, KARe only searchers particular regions of the search space.

To illustrate our approach, we go back to the example earlier presented in section 5.1.2. Consider two users Mike and Joey, whose taxonomies are depicted in Figure 6.4. The taxonomies classify the user's personal documents and also serve to contextualize the user's question. Suppose now that Joey

[3] http://www.google.com

makes the following question: "How should we deal with clients' late payment?", contextualizing it in the 'Policy' concept of his taxonomy. Referring back to the example presented in section 5.1.2, we know that Mike had a similar doubt in the past (i.e. he previously asked "What measures should we take when a client is late with his payment for the acquired services?") whose answer is now classified under the concept 'Premium'. But how can KARe know about that?

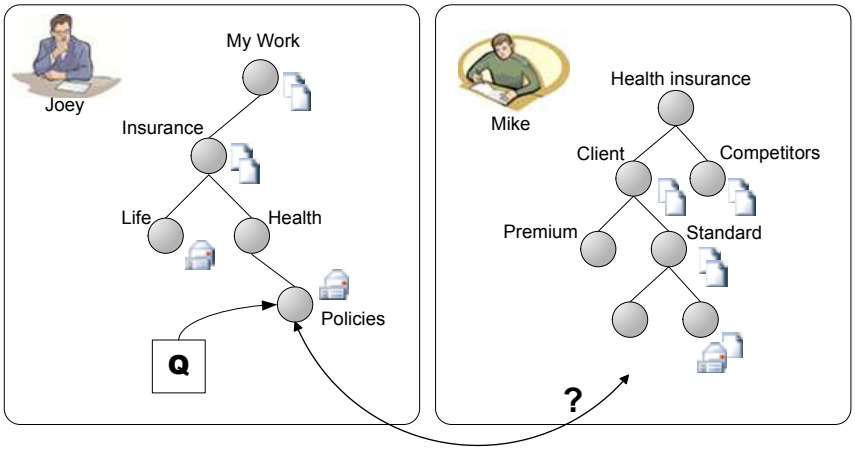


Figure 6.4: Taxonomies of Mike and Joey contextualizing documents and questions

Our algorithm should be able to identify which concept in Mike's taxonomy is more similar to the 'Policy' concept, where Joey's question is contextualized. Then, the answer can be searched within this concept. Essentially, each of the concepts of the user taxonomy has a vectorial representation. Each concept's vector is a mirror of the peer's collection index vector (also referred as vocabulary), containing the weight of the keywords that appear in the documents classified under the concept. Figure 6.5 illustrates a short vocabulary index and a vector representing a given concept C, which contains the index terms 'client', 'insurance' 'pay', and 'health', but does not contain the terms 'life' and 'customer'. In this figure, the used weights are boolean (i.e. '1' indicates the presence of an index term, while '0' indicates absence). Conversely, in our approach, the weights are given by a fraction that measures the suitability of the index term to represent a certain

concept. For finding the most similar concept to a given concept C, the algorithm calculates the similarity between the vector representing C and the vector of each of the concepts in the responder taxonomy.

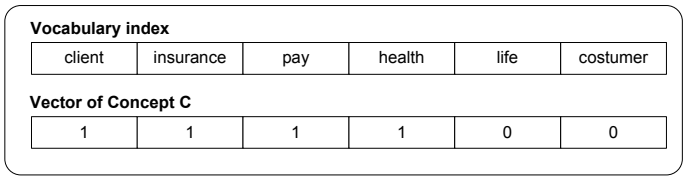


Figure 6.5: A short vocabulary index and a vector representing a given concept C in the user taxonomy

The vector of a concept is calculated with basis on the vectors representing the documents classified under that concept. Besides the documents' keywords, the concept label is also considered in the vector calculation. In fact, not only the label of the concept itself, but in addition also the labels of the ancestors of the given concept are taken into account, as has been earlier proposed in (Adami et al., 2003). More precisely, this is achieved by including the labels of all concepts of the taxonomy in the collection's vocabulary. Consequently, the label of the concept along with the label of the concept's ancestors are considered in the concept's vector calculation. The determination of the concept reference vectors follows the equation 6.6.

$$w(term_i, concept_j) = \frac{\sum_{k=1}^{n} w_{i,k}}{n} \tag{6.6}$$

Here, $w(term_i, concept_j)$ stands for the weight of the term "i" on the concept "j". Such an approach was based on the $TF * IDF$ measure already described in section 6.2.4. Equation 6.6 is basically an average formula, which calculates concept vector C based on an average of the weight of the keywords pertaining to all documents classified under concept C (thus, variable $w_{i,k}$ represents the weight of the term "i" on the document "k").

We call the process of finding the best matching concept in the responder's taxonomy *query scope reduction*. This is the main novelty of our approach. In summary, the query scope reduction can be seen as a reduction in the *search space* before we retrieve information from it, based on the fact that

the required information is more likely to be found in a specific region of this space (in the example above, within Mike's 'Premium' concept that is more similar to the 'Policy' concept selected by Joey to contextualize his query). Adding this process prior to the execution of the query, the quality of our search increases, resulting in a less noisy result set, thus recommending mostly pertinent documents to the users. In addition to that, it considerably reduces the computational complexity of the algorithm since it diminishes the set of documents to be searched.

It is important to note that each user has a different vocabulary index, i.e. the vectors of the concepts in each taxonomy are created based on different sets of keywords (index terms). Consequently, the first step on the query scope reduction is to project the concept vector coming from the questioner in the new space of the responder. This is made by calculating the intersection between the index vector of the questioner and the index vector of the responder. In this way, the concept vector coming from the questioner may be projected into the vocabulary of the responder. This projection is another novelty of our algorithm, specifically targeted at the problem of coping with different semantic representations of a domain.

After the query scope reduction step, the answer to the user's question is searched within the documents classified under the best matching concept. For that, all keywords of the user's query are taken into account to select the artifacts of the given concept. In addition to the query's keywords, the labels of the concept classifying the query and its ancestors are attached to the query (as extra keywords), this way embedding the query with enriched contextualized information. The documents are then ranked in a descending order according to the similarity with the query, and the result set is finally sent to the questioner. Our recommendation algorithm is summarized in the pseudo-code shown in Listing 6.1. The similarity function is left general to emphasize that although the algorithm now applies the cosine to determine vectors' similarity, this function can be substituted for other measures in the future.

Listing 6.1: An excerpt of KARe's recommendation algorithm

```
procedure answer(concVectA, peerQuest, questioner)
{

  //step 1: search the best matching concept for
  //the scope reduction
  projConceptVectorA := intersect(concVectA,
                                          indexB)
  for each (concept on the user B context) {
    s := similarity(currentConceptVectorB,
                              projConceptVectorA)
    if (s > maxSimilarity) {
      bestConcept := currentConceptB
      maxSimilarity := s
    }
  }

  //step 2: search among the documents in the
  //bestConcept
  queryVector := createQueryVector(peerQuest,
                                         indexB)
  for each (document in bestConcept)
    documentList.add(document,
        similarity(queryVector,documentVector))
  documentList.sortBySimilarity()

  //step 3: send the answer back to
  //the questioner
  sendAnswer(documentList, questioner)
}
```

6.3.2 Evaluation

The theory behind our algorithm seems to be consistent, however in order to prove that it actually brings any gains in the efficiency and accuracy of our search, it is advisable to evaluate it through an experiment using real data.

The ideal situation would be to experiment our algorithm against two taxonomies classifying real questions and answers. However such dataset is not available at the moment. Thus, we decided to simulate this dataset using two taxonomies that classify scientific papers. The question is simulated by the title of the paper and the answer is given by the paper's body. This seems reasonable because a question is usually short, providing us with a few keywords for the search. The answer, on the other hand, tends to be a longer piece of text. For performing the experiment, we have used two existing taxonomies: the questioner's taxonomy has been created by a PhD student to collect papers of her interest, while the one of the responder is taken from the ACM Computing Classification System [4]. Table 6.1 presents some statistics regarding these two taxonomies.

	Questioner's Taxonomy	Responder's Taxonomy
Number of Documents	250	315
Number of Concepts	28	15
Average Docs/Concepts	9	21

Table 6.1: Some statistics regarding the experiment taxonomies

The experiment may be divided into two main phases: 1) preparation of the taxonomies; and 2) execution of the evaluation experiment. In the first phase, the following activities were performed:

- the papers that were to be used as queries were selected. These papers should be classified by both taxonomies so that we know which is the contextualizing concept in the questioner's taxonomy and the concept the algorithm should find in the responder's taxonomy;

[4] http://www.acm.org/class/

- the selected papers were subtracted from the questioner's taxonomy to avoid bias (i.e. the keywords of the selected papers should not be used to compute the concept vectors in the questioner's taxonomy);
- the titles of all papers were subtracted from the papers classified by both taxonomies to avoid bias (i.e. the title keywords should not be used to compute the concept and document vectors in both taxonomies), as they were to be used as queries.

Figure 6.6 illustrates the process carried out to execute the evaluation experiment. The first step is to manually contextualize the query, by assigning it to a concept in the questioner's taxonomy. In fact, the information regarding which concept should contextualize the query was already known, since the queries were extracted from papers classified under both taxonomies (refer to items 1 and 2 above). Next, the query (i.e. a paper title) is preprocessed and submitted to the algorithm (the keywords of the title, along with the contextualizing concept's vector are sent to the responder's taxonomy). The algorithm then searches for a concept in the responder's taxonomy (query scope reduction). After the targeted concept is found, the answer to the query is retrieved from the documents within this concept. Finally, we compare the result set with the query, verifying if the algorithm is able to: 1) find in the responder's taxonomy, the **concept that classifies the paper** whose title is the query; and 2) retrieve from the responder's taxonomy, the **specific paper** corresponding to the title used as query.

We have compared the results of our algorithm with the standard approach based on the vector model (i.e. without the query scope reduction step). Concerning our approach, we have considered two options: a) to have on the result set only the best matching concept with the questioning concept; and b) to have a small subset of concepts that best matched the questioning concept. We evaluated these three approaches in terms of *recall* (i.e. the fraction of relevant documents retrieved) and *precision* (i.e. the fraction of retrieved documents that are relevant), calculated using equations 6.4 and 6.3 respectively. Then, the harmonic mean "F1" of recall and

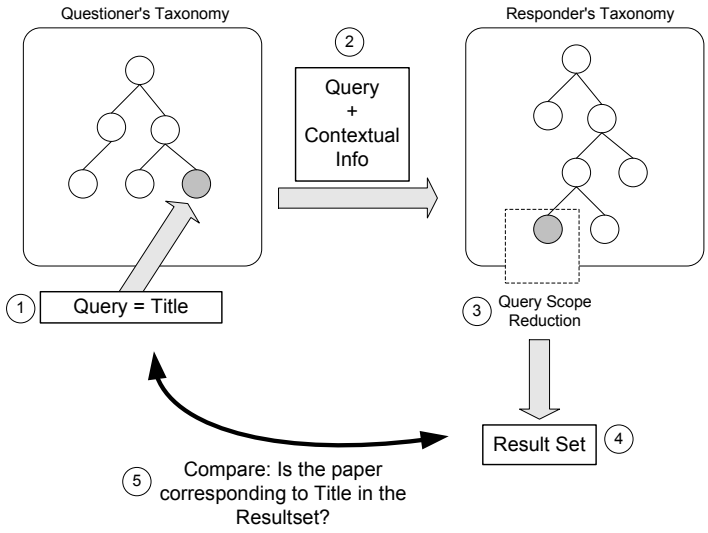


Figure 6.6: The evaluation experiment

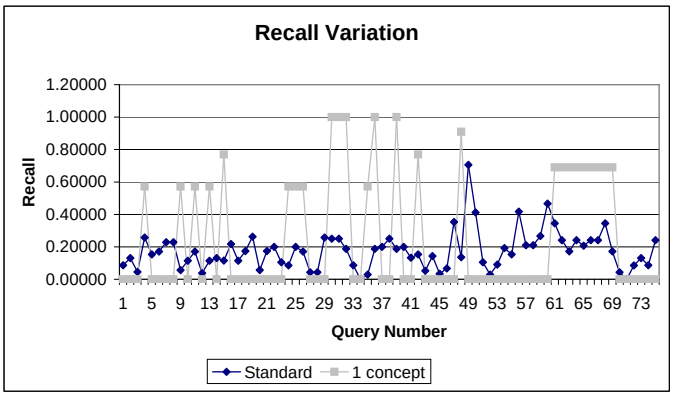
precision was calculated based on equation 6.5 and used to compare our results.

We performed 75 queries over our taxonomies, and the results are shown on table 6.2. The first column of the table shows the results of the standard approach. The second, third and fourth columns show the results of our approach when returning documents from one, two and three concepts respectively.

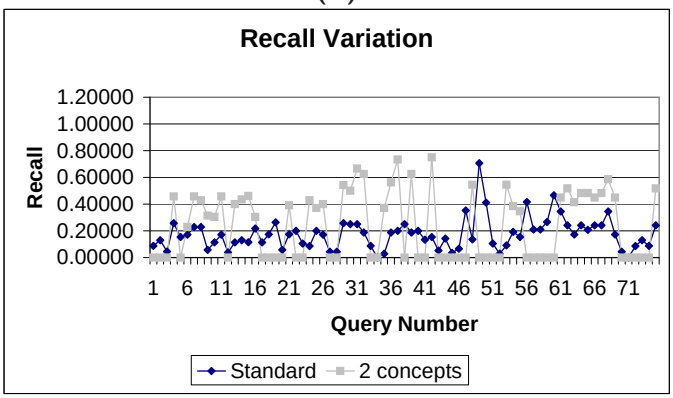
	Standard Approach	1 concept	2 concepts	3 concepts
Number of Queries	75			
Documents Found (DF)	69	25	37	43
F1 (DF)	0.920	0.333	0.493	0.573
F1 (RDF)	0.175	0.243	0.238	0.206
Comp. Complexity	158K	21K	33K	43K

Table 6.2: Experiment results

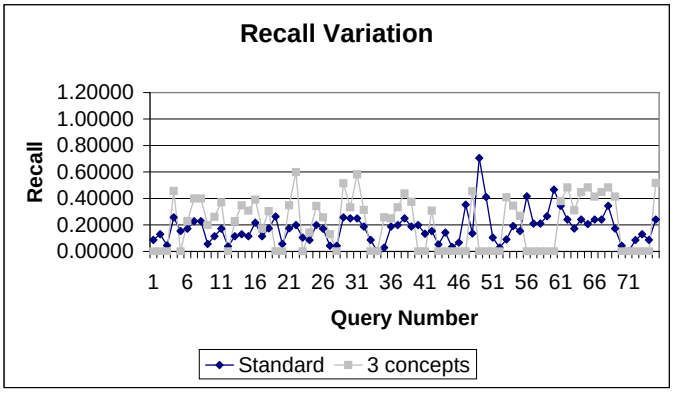
As previously illustrated in Fig. 6.6, our evaluation considers two important results. Hence, F1 was correspondingly calculated based on two measures: 1) the number of times the algorithm finds the specific document whose title is being searched (which we call DF) and 2) the retrieved number of related documents to the one being searched, given by the number



(A)



(B)



(C)

Figure 6.7: Comparison of recall measure taking the standard approach and our proposed approach using (A) 1 concept, (B) 2 concepts and (C) 3 concepts

of documents that are classified under the concept being searched (which we call RDF). For the F1(DF) value, the standard technique has a better measure. However, this approach has performed poorly for the F1(RDF) value, besides having high computational complexity (refer to the measure of 158K, which corresponds to the number of comparisons performed by the algorithm). Conversely, in general our approach returned more related documents than the standard approach, thus having better F1(RDF). In addition to that, our approach considerably reduces the number of comparisons needed to reach a result (see that for 1 concept, it needs 21K; for 2 concepts, 33K; and for 3 concepts, 43K comparisons).

Our first attempt considering only one responding concept resulted in a low F1(DF) value compared to the standard approach. This led us to consider adding flexibility to our algorithm, allowing it to select a small set of concepts in the responder's taxonomy. Taking this approach, we increased the number of specific documents found. Besides, this enhanced approach has two advantages over the standard one, as it retrieves more related documents (better F1(RDF) value), while having a low computational complexity. In general, when we increase the number or searched concepts, we increase the chance of finding the specific paper we look for, but we also end up having a result set with more noise. This can be explained by the fact that when finding the right concept, the approach searching only one concept returns only relevant documents (i.e. documents pertaining to the same concept as the specific paper being search). However, such approach has the disadvantage of not finding the right concept many times. This becomes apparent by the comparison of the graphics exhibited in Figure 6.7. These graphics exhibit comparisons between the standard approach and each variation of our algorithm.

In summary, the results show that the best solution requires achieving the right balance between the probability of finding very specific information and the ability of retrieving related information (i.e. less noisy result sets). It is also important to note that increasing the probability of finding specific information also increases the number of comparisons the algorithm should perform, thus turning the algorithm more computational complex. The

results also appoint in which direction our work should proceed, which is enhancing the ability of the algorithm of finding the right concept. Once this is done, it will not be necessary to increase the number of searched concepts anymore. This solution will then combine both finding specific and related information at once, while also keeping the computational complexity very low.

6.4 Concluding KARe's Detailed Design

In this section, we detail the design of the KARe system, started in chapter 5. We begin by refining the AOR Agent Diagram (AD) presented in section 5.3, which results in the AD depicted in Figure 6.8. This diagram is specifically tailored for implementation in JADE, corresponding in MDA terms to the Platform Specific Model (PSM) (a brief discussion on MDA is available in section 3.7).

In Fig. 6.8, we find most of the agent and classes previously depicted in the design AD presented in Fig. 5.9 (refer to section 5.3). These are the Peer Assistant, the Artifact Manager, Concept, Taxonomy, and Knowledge Artifact, further specialized into Document and Message. These are the agents and objects relevant for the implementation of the part of the system corresponding to the recommendation algorithm.

Besides these already known classes, seven other object classes have been added in the diagram. Six of these objects are related to the recommendation algorithm described in section 6.3. Each knowledge artifact is parsed by a Text Pre-Processor. During this stage, a number of operations are performed on the artifact, such as stemming and stopwords removal (see section 6.2.1). After the pre-processing step, the content of the knowledge artifact is represented as a set of Index Terms. In other words, an Index Term is a keyword from a document or a message after being pre-processed. The whole set of Index Terms consists the peers collection Vocabulary, and this is the basis for the creation of the *Inverted File Index* (refer to section 6.2.2). The Vector class concerns the vectors that represent each concept and each

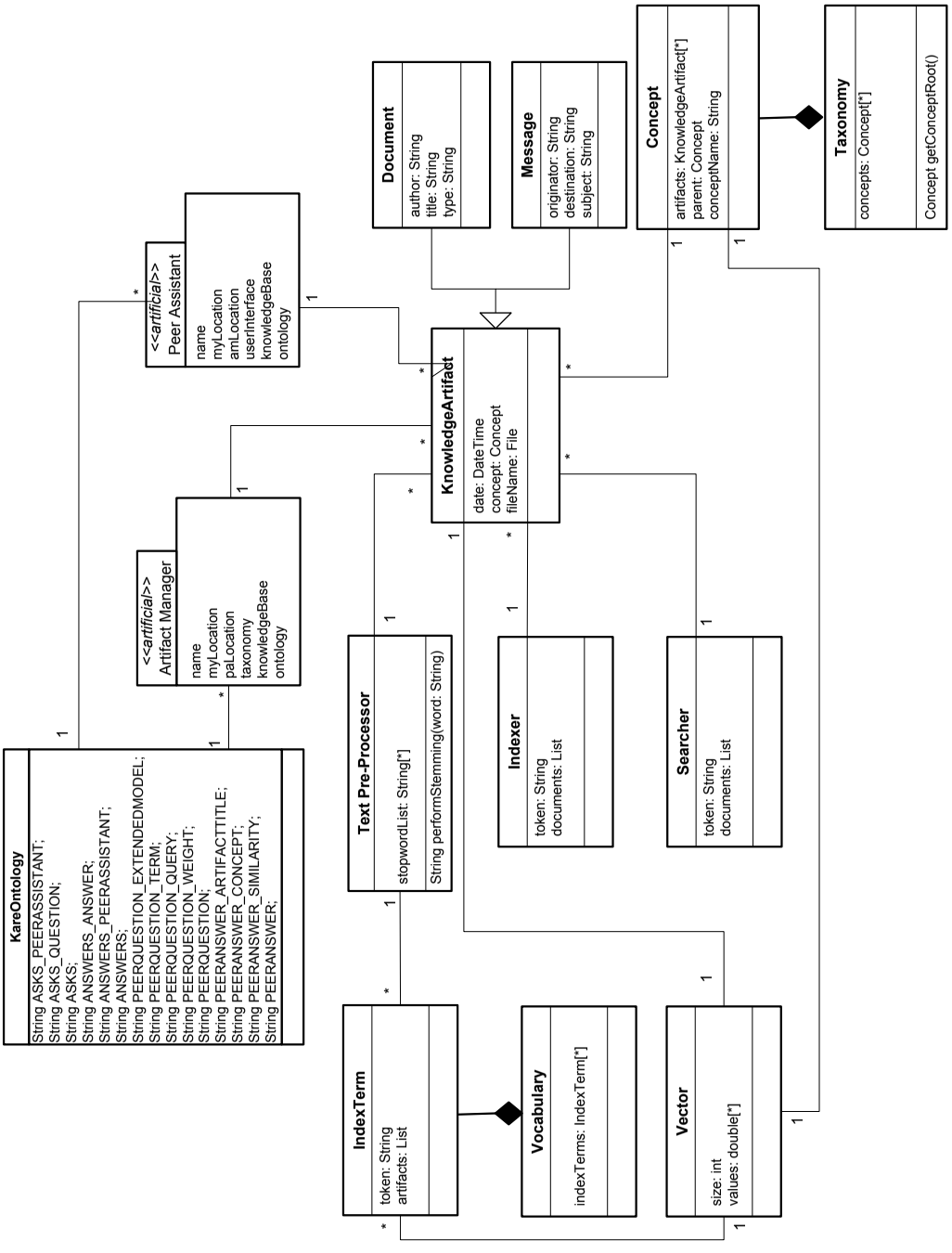


Figure 6.8: Refined AOR Agent Diagram

knowledge artifact, enabling KARe to provide a recommendation. The Indexer class is responsible for maintaining the Inverted File Index, i.e. for adding and removing artifacts to and from the index. Finally, the Searcher class is responsible for consulting the index file when a particular query is submitted.

The remaining class (KAReOntology) concerns the use of the JADE framework for the system implementation. To communicate in the JADE agent platform, agents need to agree on the semantics of the messages exchanged by them. For that, the agents share an ontology that defines the conceptualization of the agents' messages for a particular domain. In our case, we developed the KareOntology as indicated by the respective class. The attributes of this class reflect the structure of the communication ontology as described in section 6.4.1.

The AD of Fig. 6.8 shows the attributes of all agent and object classes. Concerning the agent classes, besides the domain-specific attributes (e.g. name, taxonomy, knowledgeBase, etc.) we included the platform specific attributes (e.g. myLocation, ontology). These last ones are essential for the agent's interaction in the system.

6.4.1 Agent Communication Ontology

As seen in section 3.5, an ontology is composed of a set of concepts and relations, aiming at creating a shared understanding of a particular domain. In this way, ontologies can be used as an agreed-upon vocabulary for exchanging information. This approach is adopted in JADE, which prescribes that agents should communicate using a shared ontology. Figure 6.9 shows the communication ontology used in KARe.

Fig. 6.9 shows five different entities specific to KARe, and the relations among them. The ArtifactManager and PeerAssistant entities represent the two KARe agents implemented in JADE. Agents play the central role on any ontology defined to the JADE platform since most events and other entities should be related to them.

There are three entities that are central to the communication process:

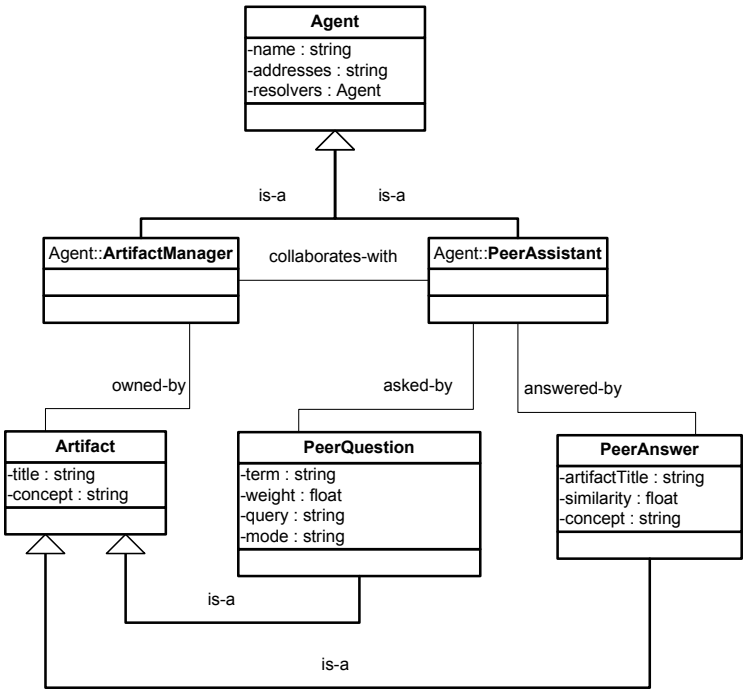


Figure 6.9: KARe's communication ontology

the Artifact, a Peer Question and a Peer Answer. An Artifact is owned by the Artifact Manager. Note that in our conceptual AD diagram (Fig. 5.8), the Peer owns the Knowledge Artifacts. We differ this here because the implementation does not consider human agents. Since the Artifact Manager controls the access to the artifacts through the recommendation algorithm, we modeled it as the owner of the artifacts in the eyes of the system. Most importantly there are the Peer Question and Peer Answer entities, which respectively represent the questions and answers exchanged in the system. Both entities are Knowledge Artifacts asked and answered by Peer Assistants.

6.4.2 Interaction Modeling

Most of the interactions among KARe agents were previously modeled in section 5.3.1. Here, we present only an AOR Interaction Frame Diagram (IFD) that is especially suitable to explicitate the interface between two artificial agents. In this way, Figure 6.10 depicts all interaction possibilities between

the Peer Assistant and the Artifact Manager, the two agents targeted so far in KARe's implementation. These interactions may be directly inferred from the previously created AOR interaction diagrams (refer to chapter 5).

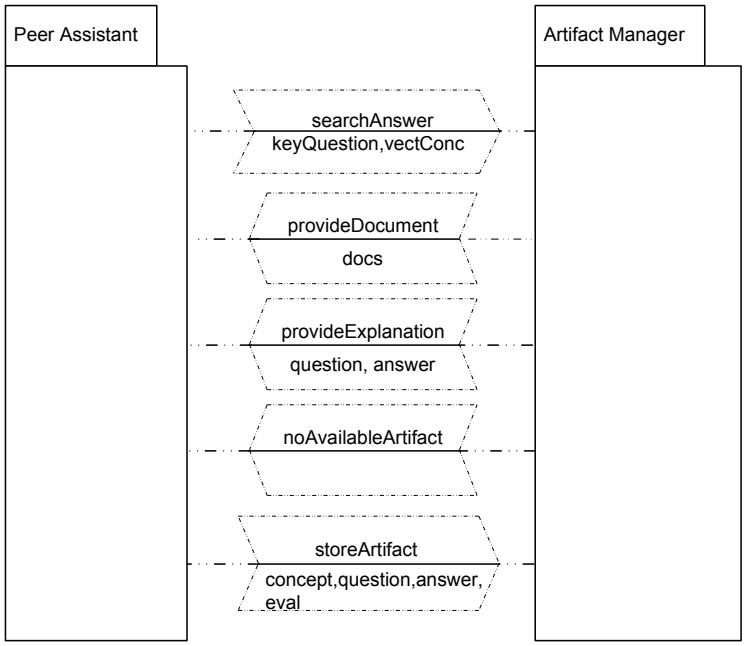


Figure 6.10: AOR Interaction Frame Diagram explicitating interface between PA and AM

Aiming at illustrating how such interaction is materialized into code, listing 6.2 illustrates how the SearchAnswer message from the PA to the AM is implemented in JADE. Such interaction is achieved through speech act, according to the FIPA ACL[5] format adopted in JADE. Following this standard, the message has a content, a sender, a receiver, is written in a specific content language (in this case, FIPA SL), and uses the vocabulary specified in a particular ontology (here, the communication ontology previously described in section 6.4.1)

In the remaining of this section, we model the interactions between the system objects instead of agents using UML Sequence Diagrams. Note that in such a diagram, messages between objects are actually method calls, which directly access the object's code. This contrasts with AOR ISDs,

[5]http://www.fipa.org

where messages between agents are speech acts, which should be treated by each agent, before perfoming an action in response.

Listing 6.2: Coding agent communication through speech act

```
//create the question, identifying
//communication performative
ACLMessage msg = new ACLMessage
                      (ACLMessage.QUERY_IF);
// set receiver's agent identifier
msg.addReceiver(new AID(artifactManagerName,
                           AID.ISLOCALNAME));
// set sender's agent identifier
msg.setSender(this.myAgent.getAID());

// set the language used to write the msg
msg.setLanguage(codec.getName());

// set the communication ontology
msg.setOntology(ontology.getName());

// add the content of the message, i.e. the
// SearchAnswer message parameters
// keyQuestion, vecConc
manager.fillContent(msg, keyQuest_vecConc);

// send message
this.myAgent.send(msg);
```

Indexing

We start by analysing the indexing process, depicted in Figure 6.11. The indexing process is triggered by the Artifact Manager (AM). The Indexer is the class that receives the method call createIndex from the AM and is responsible

for handling the process of index creation. For that, the Indexer receives two parameters: 1) a list of documents to be indexed and 2) the taxonomy that classifies these documents. The first step towards the creation of the index is to parse each Concept of the Taxonomy. The parseConcept method triggers the parseArtifact method, to parse each Knowledge Artifact contained in each given Concept. This step is necessary to create the vocabulary and index terms of the system. At this point, the TextPreProcessor is called to perform stemming and stopwords removal (parseToken method) in each artifact.

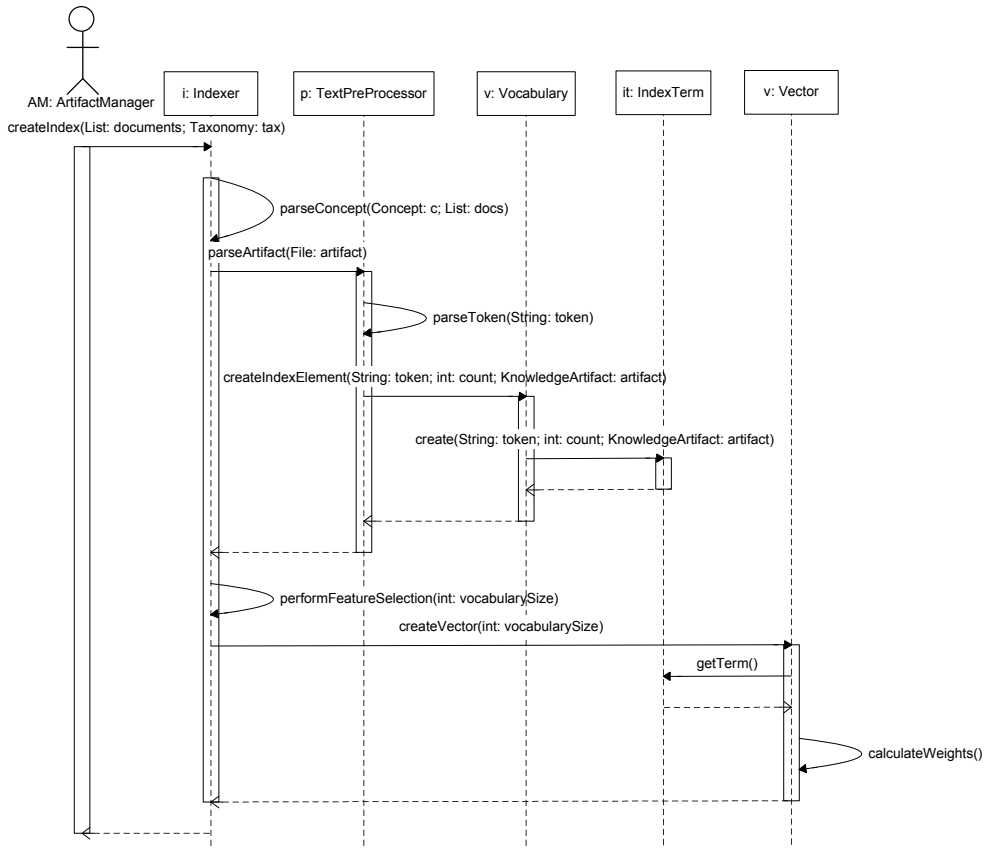


Figure 6.11: UML Sequence Diagram modeling the indexing process

Next, each keyword is then inserted in the vocabulary (createIndexElement and create methods). For that, we create an IndexTerm instance to represent the processed keyword. Following this, the Indexer should perform a *feature selection* operation (see section 6.2.2) to reduce the index file to an appropriate size (performFeatureSelection method). Once the final

index is determined, the Indexer can create the Vectors for the concepts and knowledge artifacts. On this step, the weight for each term in each artifact and in each concept is calculated and stored on the Vectors (getTerm and calculateWeights methods).

Searching

Figure 6.12 depicts the object's interaction for performing a search in the system. The searching process describes how the system finds suitable knowledge artifacts to respond to a question submitted by the AM. This process corresponds to the *SimilarArtifacts* function presented in the textual description of R1 rule of Fig. 5.18, implementing the reaction taken by the AM when receiving an incoming knowledge request.

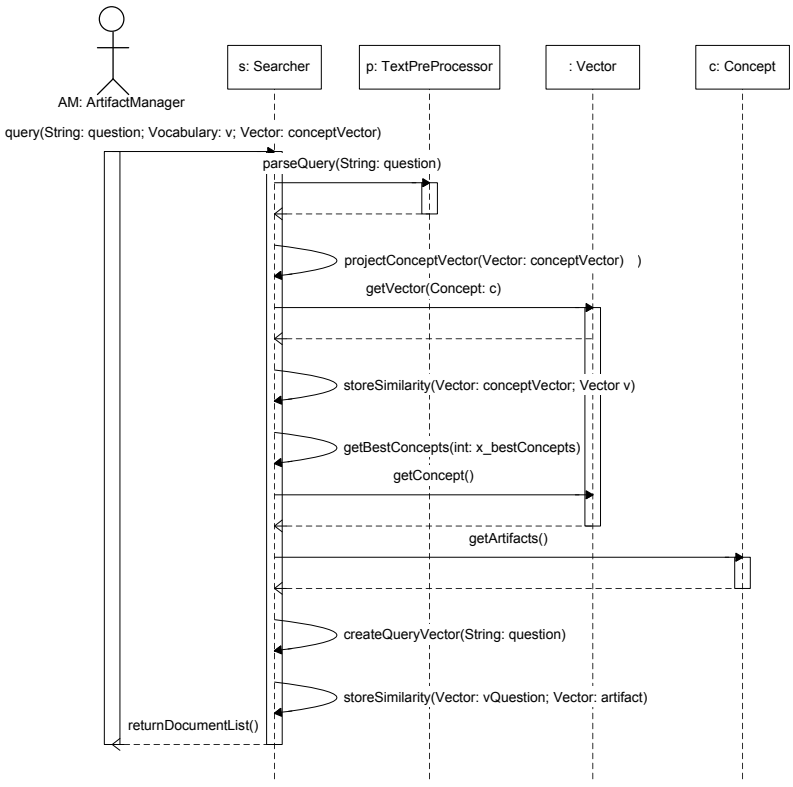


Figure 6.12: UML Sequence Diagram modeling the searching process

The main component here is the Searcher, which centralizes the query handling process and interacts with the user. When a question is made, the

first step is to pre-process it (parseQuery method). In this way, the question has the same format as an artifact, i.e. it has the stopwords removed and stemming performed. After parsing the question, it is important to project the questioner's vector into the responder's vocabulary (projectConceptVector method). In other words, the concept vector should be translated to the responder's vocabulary.

At this point, the actual search begins by finding a suitable concept on the responders taxonomy, similar to the questioning concept. In other words, the query scope reduction is executed. In this step, the getVector, storeSimilarity and getBestConcepts methods are executed. First, the questioning concept vector should be compared to each concept vector on the destination taxonomy. For that, each responder's concept vector should be retrieved and checked against the questioning concept vector. On the end of this process, we will be able to retrieve the best matching concepts with the questioning concept. To calculate the similarity, we apply equation 6.1, already described in section 6.2.4.

Once the algorithm finds the right concept, the artifacts classified under this concept are retrieved (getConcept and getArtifacts methods), enabling their comparison with the question itself. Next, the Searcher creates a vector containing the question keywords. This vector is compared with the vectors of the retrieved knowledge artifacts (storeSimilarity method). The similarity between each artifact and the incoming question is used as a ranking function. The ranked artifacts are finally returned to the user, presenting them in descending order according to the artifacts similarity measure when compared to the query.

6.5 Prototypes

Two prototypes of the KARe system were implemented: a desktop computer version and one for access in a handheld device. These two prototypes are described in the sequence.

6.5.1 Desktop Prototype

The main purpose of having KARe as a desktop system is to allow organizational members to exchange knowledge while organizing their personal knowledge items locally (Ludermir et al., 2005). The system agents should mediate knowledge exchange in the peer-to-peer network, providing recommendations according to the algorithm presented in section 6.3.

Figure 6.13 shows a screenshot of the desktop prototype. On the left part of the window, the figure depicts a user taxonomy, showing in a tree of concepts, how the user structured his/her knowledge. On the top, there is a text box where the user enters his question, followed by a "Search" button. Having inserted the question, the user may press this button to trigger the searching mechanism. The "Results" of the search are shown in the right side of the screen, classified by peer (the peer from which the artifact was retrieved), and ordered by the similarity of the artifact regarding the question submitted by the user.

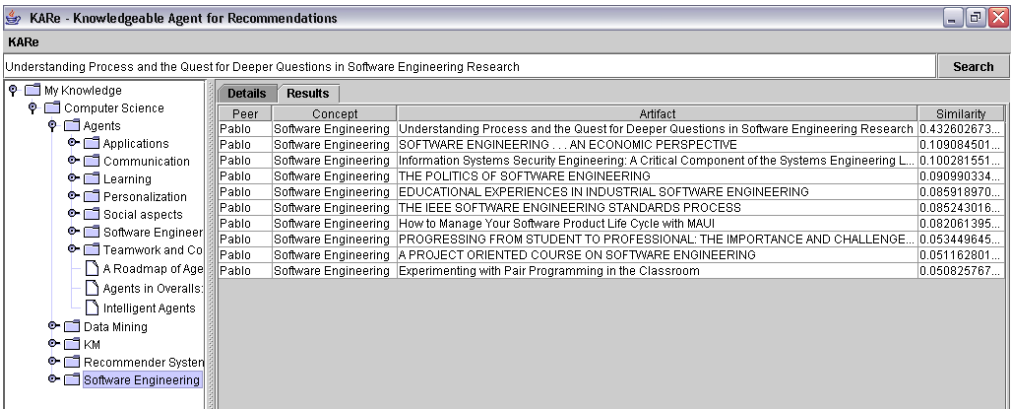


Figure 6.13: A screenshot of the desktop prototype

The desktop prototype was developed as two integrated components, as shown in Figure 6.5.1. Such component scheme aims at providing a pluggable architecture with replaceable parts. The components communicate with each other via well defined interfaces facilitating the adaptability of new components into the architecture. For instance, the information retrieval component could be replaced by any other "searching" mechanism

if more appropriate techniques are developed in the future. Moreover, we could use any agent platform that conforms to the FIPA specifications to compose the recommender agents component.

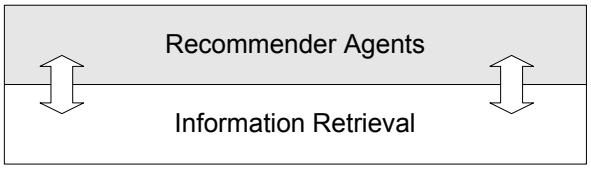


Figure 6.14: Two components composing the desktop prototype

The prototype was completely implemented in Java. The *Recommender Agents* component was implemented using the JADE framework [6]. JADE works as a middleware for the agents communication. The agents are implemented via Java classes that communicate with each other via Java RMI. To enable their communication, an ontology was developed, as presented in section 6.4.1. This ontology has been designed using the Protégé Ontology Editor [7], and implemented in Java classes using the Beangenerator Protégé plug-in [8].

The implementation of the *Information Retrieval* component is based on the use of the Lucene library [9]. Lucene is a search engine library that contains implementations of well-known algorithms and components used in our system, such as: the inverted file index, a stopword remover component and the stemming algorithm. Persistence of the relevant metadata regarding knowledge artifacts was achieved with the use of XML[10] files. The taxonomy is also represented in an XML file, structured as prescribed in a particularly developed XML schema.

[6] http://jade.tilab.com/
[7] http://protege.stanford.edu/
[8] http://acklin.nl/page.php?id=34
[9] http://lucene.apache.org/
[10] http://www.w3c.org

6.5.2 Handheld Prototype

The KARe handheld prototype (Ludermir, 2005) is based on the assumption that suitable responders to a specific question can be selected based on their geographical proximity to the questioner. This assumption comes from the realization that people usually share spaces with individuals with whom they share interests, e.g. workmates within an organization, researchers in a conference, and classmates in an educational institution. This version of KARe fits into the category of the so-called *nomadic services*, which comprehends network services that are accessible by mobile computing independently of the user's geographical location (Ludermir, 2005).

By changing user's location, the recommendation is likely to change as well. In contrast, if you try to get a recommendation from the desktop system, the result is the same until the document index is updated. This happens because, as seen in the design of KARe's desktop system (see for example, Fig. 5.13), the system searches for knowledge artifacts by broadcasting the knowledge request to all peers connected to the network. Thus, such design is slightly modified to accommodate the new search mode. After receiving a request from the user, a search is triggered when the system senses the presence of another peer in the vicinity. The request is then submitted solely to this peer. In this way, the handheld prototype also avoids the problems of scaling the system to a great number of peers, which still remains to be targeted in the desktop version.

The development of the handheld prototype adds an extra component on top of the ones previously described in section 6.5.1, as shown in Figure 6.15. The interface between the new *Peer Discovery* component and the *Recommender Agent* component is achieved by wrapping up outputs of the former into Agent Communication Language (ACL) messages that are then sent to the latter.

Figure 6.16 shows the distribution of the components of KARe. The elements are physically distributed in three locations: handheld computers, desktop computers and a server. The server can be federated by many stations, but is here shown as a single entity for simplicity. The dashed

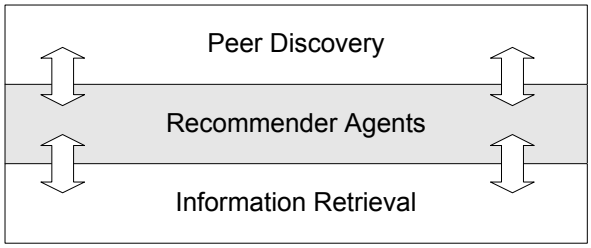


Figure 6.15: Extra component for the development of the handheld prototype

arrows show the dependency between components, and the numbers within the circles show execution ordering.

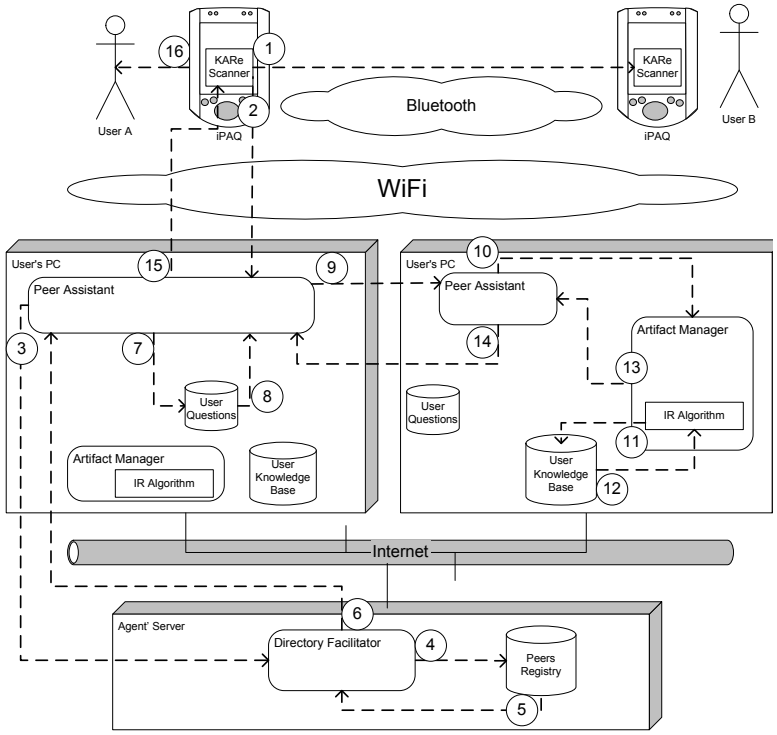


Figure 6.16: Distribution of the fixed and handheld components

As can be noted, each user has access to one handheld and one desktop computer which are connected via a wireless connection (802.11x). Each handheld is responsible for sensing the presence of other devices in the vicinity and to advertise its presence to its neighbors. The device discovery is performed using a *bluetooth link*. When another handheld is detected, the

information about the new device is sent to the user's desktop computer. Such information is forwarded to the server to verify whether the found device is part of KARe peer-to-peer network or not.

When presence of another KARe peer is sensed, the *Peer Assistant* Agent contacts the "respondent" *Peer Assistant* to trigger the recommendation process. If the "respondent" peer recommends any artifact, the user is contacted on his handheld. Bellow we describe each step shown in Fig. 6.16 in detail.

- *Device Discovery (1):* In each handheld there is a module named "KARe scanner". This module collects information about the devices found in the vicinity via bluetooth and triggers the device verification process.

- *Device Verification (2, 3, 4, 5, 6):* Once the handheld is detected and collected identification information about another device, it sends this information to the user's computer (step 2). The user's *Peer Assistant* receives the information and verifies whether it corresponds to another peer or not. For that, it forwards this information to the *Directory Facilitator* agent (step 3) that consults the peers services database to check for the peer's existence (steps 4 and 5). When it knows about the discovered device status, the *Directory Facilitator* sends its findings back to the questioner *Peer Assistant* (step 6). If the discovered device is a peer in the KARe network the recommendation process is triggered, and otherwise it is aborted. Figure 6.17 (A) shows the resulting screen in the handheld after identifying the devices that run KARe.

- *Recommendation process preparation (7, 8):* Before actually starting the recommendation process, the *Peer Assistant* agent consults the **User Questions** database. If there are any questions, the *Peer Assistant* wraps each question with the appropriate format for sending them over the network.

- *Recommendation process (9, 14):* This process is the core of the sys-

tem. It is the actual simulation of the question-answering process. The first peer sends a question (step 9) which the second peer tries to answer (step 14) by recommending a selection of artifacts (e.g. documents and messages).

- *Artifact search (10, 11, 12, 13):* These steps are the mechanism that enable the recommendation amongst peers. When the *Peer Assistant* receives a question, it forwards it to the *Artifact Manager* agent (step 10), since the latter knows how to match the question against the **User Knowledge Base**. This matching is performed by the already described **Information Retrieval algorithm** (steps 11 and 12). Finally the retrieved artifacts are sent back to the *Peer Assistant* (step 13).

- *User notification (15, 16):* After receiving the answer for its question, the *Peer Assistant* sends a notification to the user's handheld to warn him/her that new artifacts were recommended. Figure 6.17 (B) shows the screen with the notification of new recommendation from another peer.

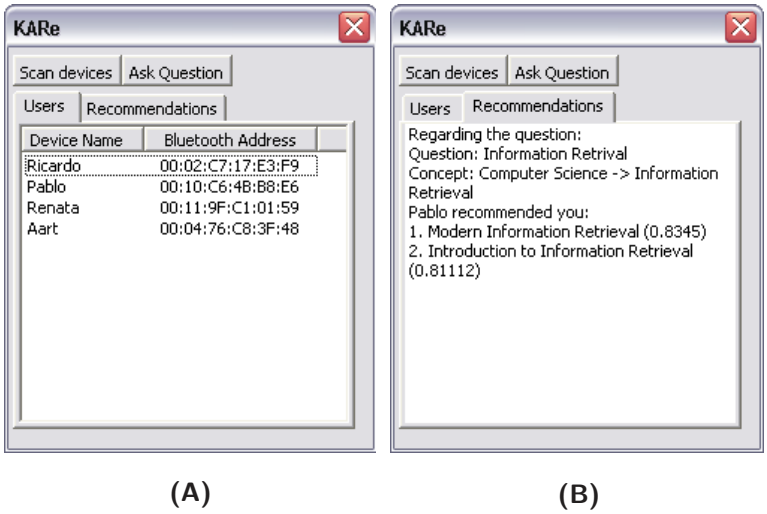


Figure 6.17: Screenshots of the handheld prototype

As in the desktop version, the agents are arranged in a peer-to-peer fashion composing a recommender system running on desktop computers. The *Recommender agents* and *Information Retrieval* components are practically intact. However, a different GUI has been developed to run in the iPAQ handheld device. To overcome problems with the limited resources on such devices, the recommendation service was kept in the desktop. This application communicates with the iPAQ through a wireless link to receive the user's inputs and send back recommendations. The GUI was implemented using the Personal Profile API [11] implementation of the Java 2 Micro Edition version (J2ME). Finally, the *Peer Discovery* component was implemented using the Interconnect architecture (Uiterkamp, 2005), developed to enable HTTP communication between service hosts and nomadic service components.

6.6 Related Work

The most distinguishing feature of KARe is given by the consideration of taxonomic information to recommend knowledge artifacts. We have no knowledge of other initiatives that apply taxonomies to aid the process of questioning and answering, adding to the user's query the contextual information provided by the concept to which this query is assigned. Other than this, KARe's distinction is materialized in the query scope reduction stage of the recommendation algorithm, in which a concept of the questioner's taxonomy is matched with concepts from the responder's taxonomy. Matching taxonomies has been targeted before, having gained considerable strength in the last few years, especially boomed by developments in the Semantic Web. In this section, we just cite two initiatives more closely related to ours.

There are mainly two ways of conciliating two different taxonomies A and B. One focuses on mapping labels associated with a concept of taxonomy A into concept labels of taxonomy B. Among the works that adopt such technique, some use only syntactical information of the labels, simply matching

[11] http://jcp.org/aboutJava/communityprocess/final/jsr062/index.html

keywords, while others go beyond this, considering in addition to syntax, semantic information about the labels, usually supported by a dictionary or thesaurus. This is the case of the CtxMatch algorithm (Bouquet et al., 2003), a linguistic-based approach which adopts WordNet lexical reference system[12] to disambiguate and stem labels. This algorithm indicates the relationship between two matched labels, i.e. it informs if the label found in taxonomy B is equal, less specific or more specific than the selected label in taxonomy A. The problem with this kind of technique is that it usually results in low recall. Although the used dictionaries or thesaurus provide valuable additional information about the labels, this if hardly enough and a match in the responding taxonomy is rarely obtained (Avesani et al., 2005).

Our algorithm adopts a different approach of matching taxonomies, by considering not only the labels representing the concepts of the taxonomy but also the keywords of all documents classified under the concept. This adds a great deal of information to the concept representation, usually improving the algorithm's performance at least in terms of recall. A similar approach to ours is adopted by (Avesani et al., 2005). However, this work tries to identify the semantic relationship between the two corresponding nodes, while our approach limits itself to finding one or a few most similar nodes in the responder's taxonomy. In addition to that, another difference may be highlighted. For functioning properly, the approach of Avesani et al. (2005) requires the two taxonomies to share documents, as the similarity between them is calculated on the basis of this redundancy. Our algorithm does not require such duplication, working well even if there is no redundant information.

6.7 Conclusions and Future Work

The main focus of this chapter was the information retrieval algorithm implemented to generate recommendations in the KARe system. Besides, we presented the remaining of the system's detailed design and details on pro-

[12]http://wordnet.princeton.edu/w3wn.html

totype implementations. In this respect, the reader is able to see how the previously high-level design is transformed into concrete elements of a system (such as XML files and Java code) and then implemented in two different prototypes.

The results of the algorithm evaluation experiment showed considerable gains in the recommendation quality are achieved by using the proposed approach. In the future, we aim at confirming this conclusion by experimenting the algorithm against different and larger datasets. However, we already envision some possibilities of enhancing the query scope reduction performance. Our research agenda for the future includes the experimentation with the *smoothing* technique presented in (Sona et al., 2004) to improve the representation of the taxonomic concepts. This technique uses information from other concepts in the taxonomy (e.g. parent and siblings) in the vector calculation for a given concept C. Concept vectors of neighboring concepts are propagated to the vector representing C, although with reduced weight, according to the distance between these concepts and concept C. This technique attained gains in classifying documents into particular taxonomy nodes (Sona et al., 2004), a different but related application to ours. Smoothing is a technique targeted at situations in which there are many nodes classifying only a few knowledge artifacts. Thus, such technique is suitable for initial stages of system use, when KARe peers are starting to collect their documents and exchange questions and answers.

Something else that remains unanswered here regards the possibility to scale the system to a larger set of peers. The developed algorithm consumes a great deal of resources, so scaling it up should pose a big challenge, especially in the case of the desktop version of the system. We foresee two possibilities to be investigated to enhance system performance. The first one regards the beforehand calculation of the nearest neighbor peer to answer to requests on specific subjects. Nearest neighbor calculation is a common practice in the context of recommender systems and may follow existing algorithms (Montaner et al., 2003). The other idea we could explore alternatively or in addition to this one is to set up a similarity threshold, limiting the number of documents exchanged between Peer Assistants to reduce network traffic.

Other future research direction include the implementation of the Broker agent and KARe's proactive functionalities, not targeted here. The question and answering functionality was selected for first implementation for being in the core of the system's proposal, but also for offering more challenging problems from a technical point of view than the other system features.

Chapter 7

Conclusion

"And this way, arriving and leaving
are only two sides of the same journey.
The train that arrives is the same train
that leaves." Milton Nascimento

This chapter presents a summary of the main conclusions and contributions of this work and outlines a number of directions for further research.

The chapter is organized as follows: section 7.1 presents a general overview of the main results of this. After this brief summary, section 7.2 examines these outcomes more closely, inspecting how each of the research questions were targeted and how they advance related state of the art. Following, section 7.3 sets our future research agenda.

7.1 Results Overview

In the struggle to survive and compete in face of constant technological changes and unstable business environments, organizations recognize knowledge as its most valuable asset. Consequently, they often invest on KM, seeking to enhance their internal processes and available technologies to sustain and disseminate knowledge throughout their environment. This thesis advances the state-of-the-art in this area in two distinct ways:

1. by providing **ARKnowD**, a methodology to guide the development of KM *information systems* and *practices* fitting the particular needs and requirements of an organizational setting;

2. by presenting **KARe**, a socially aware recommender system that supports knowledge creation and sharing by simulating the natural social process one gets engaged in to fulfill a knowledge request.

The work on the ARKnowD methodology was triggered by the realization that although claiming to bring synergy and innovation to organizations, KM solutions are often reported not to fulfill their promises, being consequently abandoned or misused. In this thesis, we examined some of the main challenges of KM settings, concluding that before a solution is developed or adopted, the organizational environment needs to undergo thorough analysis. Such analysis is aimed at identifying from the environment's characteristics, those that may hamper effective KM (and should thus be overhauled) and those that may contribute to it (and should thus be reinforced).

The observation of these challenges also showed that most of them are closely related to the fact that current solutions fail to comply with the organization's social dimension. In particular, we realized that besides focusing on the organization's overall objectives and strategies, seeking for an effective KM solution requires special attention to be payed to knowledge holders, i.e. people working in the organization's several points of actions. This realization led to the adoption of *constructivism* as the theoretical basis of our work. By focusing on constructivist approaches and contrasting them with prominent KM theories, we were able to distill the essence of what characterizes a conducive environment to KM. We called these resulting principles the *Constructivist KM building blocks*. Such building blocks can be used as guidance to facilitate the aforementioned analysis, assisting the identification of KM inhibitors and drivers within the organizational environment.

Aiming at supporting the clear understanding of the organization's social dimension, observing its compliance to the Constructivist KM building blocks, ARKnowD gives special attention to the initial phases of system

development. In this way, the methodology aims at eliciting and modeling the requirements of the system-to-be, by considering both the organization's overall objectives and the knowledge holders' perspective. Furthermore, beyond analyzing the domain, ARKnowD consistently conducts to the design of the proposed solution, modeling the system entities, interaction and internal behavior.

Verifying the appropriateness of the agent paradigm for modeling human organizations, we created ARKnowD as an intrinsically agent-oriented methodology. However, by examining available work on agents, we realized that a single engineering approach is not fit for all domains and cases. Conversely, existing agent-oriented engineering methodologies should be combined on demand, based on the right set of concepts and techniques to target a specific domain or situation. Therefore, we took some effort in understanding which are the agent-related concepts that most suitably describe the KM domain, especially capturing the analysis regarding Constructivist KM building blocks. For that, we built an ontology of agent concepts that we then used to evaluate, adjust and coherently combine the notations adopted in ARKnowD.

In order to evaluate ARKnowD, we provided an experimentation of the methodology, applied to analyze a fictitious scenario that illustrates some of the main KM challenges. As a result of this analysis, proposals for changes in the organization's structure and processes arise. Particularly, the requirements for a recommender system to support such setting are elicited. This led us to the proposal of KARe, the second main contribution of this work. Still following ARKnowD, we fully designed KARe, demonstrating that ARKnowD is able to take the developer from a detailed domain analysis to a consistent design activity.

Besides using it as a case study for ARKnowD's methodology, this work also explored the suitability of the KARe system to support KM. KARe enables users connected in a peer-to-peer network to locally organize their knowledge artifacts, while sharing them through questions and answers. By simulating the question and answering process, naturally undertaken when people seek for knowledge, KARe aims at smoothly fitting into organiza-

tional practices. Up to now, KM systems had mostly targeted the formalization and exchange of *explicit knowledge*, in the form of documents or other physical artifacts, often annotated with metadata, and classified by taxonomies or ontologies. Investigations surrounding *tacit knowledge* have been so far scarce, perhaps by the complexity of the tasks of capturing and integrating such kind of knowledge, since it is usually confined on people's mind. Taking a flexible approach on supporting this kind of knowledge conversion, KARe relies on the real potential of *social interaction* to support knowledge creation and sharing. This emphasis is motivated by the assumption that such a process and, especially question and answer exchanged by community members, may eventually result in the disambiguation of tacit knowledge.

The core of the KARe system regards a recommendation mechanism that mediates the questioning and answering process, providing users with knowledge artifacts to satisfy their knowledge needs. Our work comprised the description, implementation and evaluation of such mechanism. In this respect, this thesis presents an innovative information retrieval technique, based on semantic information encoded in taxonomies that structure the artifacts collection. This information is applied to reduce the search scope, thus diminishing computational complexity when compared to standard approaches, while providing result sets with less noise (i.e. more relevant documents) at the same time. The technique has been implemented in a prototype of the KARe system, and evaluated in comparison to a standard approach. Results of this evaluation showed the gains achieved by applying our technique, also leading to the identification of points of improvement.

7.2 Research Questions Revisited

In this section, we focus more deeply at the realized work, discussing our main findings. More specifically, we revisit each research question, presenting in detail how each of them was addressed and what are the strengths and weaknesses of the proposed solutions in comparison with related work.

7.2.1 Applying Agents to Support Constructivist Knowledge Management

Our first research question (RQ 1) regards the suitability of the agent-oriented paradigm to support Constructivist KM. In chapter 2, we have started targeting this question by analyzing the state of the art in KM, in order to identify:

- the main challenges of such settings;
- background theoretical work that could lead us to the proposal of more effective approaches;

By examining both practical issues and theories and by taking constructivist hypotheses into account, we were able to identify a few principles that may characterize the KM environment, leading to less resistance towards KM systems and practices. In general, such principles refer to a less techno-centric view on KM, focusing more attentively in the social aspects that naturally motivate knowledge sharing and learning in practice. We defined these principles as the Constructivist KM building blocks, claiming that they should be pursued both by KM researchers and practitioners.

Next to this, we also investigated the ability of **agents** to model human organizations, representing important entities inherent from these settings, such as human, organizations, organizational units, and information systems. Although this ability has been largely theoretically advocated, most current agent-oriented approaches still address system development by modeling artificial agents (i.e. the ones composing a system) from the start. By doing so, these approaches fail to link system requirements with the real needs and wants of the system stakeholders, as this important information remains untouched and hindered. Contrarily, when profiting from the agents' inherently social and cognitive nature, the analyst is able to create a domain model that helps uncover such details. Specifically, agents are powerful abstractions for capturing human's beliefs and perceptions, to model their interactions, and for capturing the commitments they establish with

each other on the organization's behalf. The understanding of these elements is paramount for the adoption and proposal of effective KM information systems and practices, confirming the suitability of the agent paradigm for our purposes.

Regarding the suitability of agents, we were particularly interested in understanding to which extent agents can be used to allow capturing and reasoning about Constructivist KM building blocks. This topic was discussed in chapter 2. Moreover, it is further clarified in chapter 3, which provides a deep understanding of the agent concepts in the ontology presented, and in the practical application of our proposed approach in the remainder of this thesis. In particular, chapter 4 applies it for domain and system analysis and chapter 5, for system design.

Still in chapter 2, we analyzed diverse agent-oriented software engineering methodologies and languages. We did not have the purpose of being complete regarding the available work in this area, but rather aimed at illustrating how each of the approaches we explored targets the development of agent-oriented systems. Moreover, we were especially interested in using them as subsidies for the proposal of our own approach, specifically focused on the KM domain.

7.2.2 Developing a Methodology to Support Knowledge Management

In chapter 2, we concluded that each organizational environment is unique and should be closely inspected before a KM solution both in terms of practices and information systems, is proposed or adopted. But how should we proceed in this analysis? This brings us to our second research question (RQ 2), which regards the development of a methodology to support KM.

In particular, we intended to propose a comprehensive methodology that targeted all system development activities. Here, system not only refers to information systems, but also considers human systems. In this way, a KM solution can be understood as a set of practices or as an information sys-

tem to be adopted. In both cases, according to the *ARKnowD methodology*, proposed in chapter 3, system development is an iterative process consisting of the following activities: *requirements elicitation, early and late requirements analysis*, and *architectural and detailed design*. As can be noted by our focus on requirements, great strength was given to the initial development activities, aiming at grasping the idiosyncrasies of each organizational setting.

Given the appropriateness of the agent development paradigm, our methodology was tailored to profit from the agent social and cognitive nature, using its underlying concepts as modeling constructs. In addition to that, one of our main assertions is that in the context of development methodology, no silver bullet exists. Instead, existing work on agent-oriented software engineering should be combined, aiming at exploring their strengths while minimizing their weaknesses, giving each modeling setting or situation.

Having experimented the combination of two specific existing agent-oriented approaches, namely the Tropos methodology and AORML, we are now able to define a general set of guidelines, supporting system developers on merging other methodologies and languages. To sum up, these guidelines are the following:

1. closely consider the characteristics of the targeted domain before selecting the methodologies and languages to be applied. Characteristics of the domain have a direct impact on the approaches selection. For example, if the problem being targeted is life threatening, a formal approach is needed to prevent undesirable failures. However, if the targeted domain comprehends an organization in need of KM support, an approach supporting extensive domain analysis as the one we propose is advisable to enable the organizational environment to be well understood before a solution is proposed;

2. once the methodologies and languages to be combined are chosen, verify if together, they are consistent and cover all important agent-oriented cognitive concepts useful in the targeted case. If not, extend the approaches where needed, providing both a conceptualization of

the missing elements and corresponding modeling constructs. This assessment can be done by using a reference ontology of agent concepts, as the one proposed in chapter 3.

3. provide a clear method to map the concepts of the adopted notations, in order to assist the analyst and designer that are going to apply the combined approach. In this work, we adopted a MDA-inspired transformation method, involving the transformation of a source language to a target notation, having the metamodels of the two languages as input for the transformation process. Other methods may be pursued, including the other ones also proposed by the MDA initiative.

4. provide a set of guidelines for the use of the resulting approach. This includes information on how to apply each modeling construct, and in which modeling activities. Moreover, it should cover information on which step to realize the mapping of concepts, as established by the method mentioned above.

Chapter 3 describes all the topics listed above, i.e. it gives the reasons behind the selection of the two approaches composing ARKnowD, presents an evaluation of ARKnowD's notation, describes a transformation method between the two adopted approaches, and grants ARKnowD's users with guidelines on the use of the methodology. In addition to that, chapter 4 and chapter 5 illustrate the use of ARKnowD in practice. They start with the analysis of a scenario that, although fictitious, exemplifies important issues of KM settings, and finish with the detailed design of a recommender system, whose prototyping is described in chapter 6.

Our second research question has been further refined in two sub-questions dealing with specific methodological issues. From now on, this section targets each of these two sub-questions.

Agent Cognitive Concepts and Development Activities

As mentioned in section 7.2.1, one of the biggest strengths of the agent paradigm for our purposes is given by the social and cognitive nature of

agents. Nevertheless, restating RQ 2.1, how can we connect the concepts underlying agents and the system development activities? If this question remained unanswered, it would be hard to fully profit from this important characteristic of the agent paradigm in practice.

We realized by the study of KM state of the art (reported in chapter 2) that KM settings are complex domains, highly unstructured and greatly influenced by their human dimension. Thus, in one way or another, most concepts characterizing agent's rationale, such as intention, perception and belief, are relevant here. Moreover, other concepts that guide the understanding of the social relations between agents, such as dependency, delegation and commitment are of utmost importance for the domain to be correctly grasped. Other essential concepts are the ones relating to agent's actions and perceptions, such as plans, and communicative actions, which lead agents to collaborate and change their inhabiting environment. Finally, we emphasize the importance of considering not only active entities of the environment but also modeling the passive ones, i.e. the resources that agents use when pursuing their goals. We discussed all these concepts in chapter 3, where we proposed an ontology of agent concepts to enable the design and the evaluation of agent-oriented modeling languages.

However, this discussion does not clarify in which modeling activity each of these concepts are to be used. This is more clear in the proposal of our transformation method between Tropos's notation and AORML. AR-KnowD starts by the analysis of the goals of the stakeholders. This choice was adopted by several work in requirements analysis area, for the realization that stakeholder's goals directly connect to their real wants and needs (Kavakli and Loucopoulos, 2005). Furthermore, a prominent work on KM (Nonaka and Takeuchi, 1995) has appointed organizational intention (which refers to strategies and goals of the organization) as one of the main driving forces behind the adoption of effective KM practices. In the first development stages of requirements analysis, details regarding processes are hardly necessary. Instead, we limit ourselves in understanding the dependencies between the agents of the domain, and determining their plans (a high level view on processes). This maintains the analysis on the right focus, i.e. to de-

termine the requirements of the system under development, preventing the analyst to get lost among unnecessary details of the organizational setting.

The design activity is the point where all details regarding the system should be completely uncovered. In this moment, all relevant information concerning agents' internal characteristics, beliefs, perceptions, actions and commitments to other agents are finally designed. The use of the two notations is tightly coupled, enabled by the adopted transformation method. This allows system functionalities to be connected back to the elicited requirements.

When Agent Cognitive Concepts become Concrete

Agent-oriented underlying concepts have been extensively discussed in literature. However, how these concepts connect to concrete parts of information systems is still open for discussion (refer to RQ 2.2). Our work takes the system developer all the way from requirements analysis to system implementation and we are able to make links from the modeling constructs referring to such concepts and parts of the implemented prototype.

As mentioned before, goals give rise to system's requirements. These requirements are iteratively refined until the point that each goal to be designed has an assigned plan, which abstractly states how each goal should be targeted. This only excludes those goals that were abandoned in the way, or solved by other means. Then, for each plan, we provide diagrammatic descriptions that detail the process followed by the agent, his/her interactions with other agents and their internal behavior. Alongside, agents composing an information system are modeled as agent classes and resources turn into object classes. Later, both agents and objects may be turned into code using an agent-oriented framework or an object-oriented programming language. In this thesis, we experimented the former, using the Java Agent Development Framework (JADE). Moreover, some of the objects realize the system persistence, being converted into database tables or XML files, for example. Besides objects comprehended in the domain (i.e. the former resources), some of the agent's beliefs are also turned into persistent objects, hence

constituting available information for agent's consultation and update.

Agents' interaction happens through message passing, which consist in speech act representation, such as 'inform', 'acknowledgment' or 'request' messages, containing a sender, a receiver and some content parameters. If an agent framework is used for implementation, these messages are directly implemented into codes. Otherwise, such mechanism needs to be coded from scratch. Dependencies between agents early captured in the requirements' analysis or architectural design are transformed into commitments in detailed design. These constructs help regulate contracts between agents, serving during implementation as important indications of where exception handling mechanism should be heavily cared for. Finally, agent's reactive behavior is captured with the use of reaction rules. Such rules may be directly implemented, in case a rule-based programming language is applied. Conversely, as is the case of our system, these rules typically turn into "if-then-else" structures coded into agent's behavior files.

7.2.3 Using KARe to Support Constructivist KM

Research question number three (RQ 3) concerns the second contribution of this work, i.e. the development of a socially-aware recommender agent to support Constructivist KM. The KARe system was carefully tailored to meet the requirements elicited and analyzed in chapter 4, following ARKnowD's application. These requirements mirrored some of the Constructivist KM building blocks defined in chapter 2. Table 7.1 shows the relations between KARe main requirements and the Constructivist KM building blocks.

The first requirement refers to how the knowledge assets are organized and shared. KARe allows knowledge sharing following a peer-to-peer model. In other words, a user connected to others in a network keeps his/her knowledge artifacts stored in his/her own personal computer, thus maintaining full control over them. This gives full *autonomy* to the user, since he/she can share artifacts on will, besides being able to update or take items down. Meanwhile, while these resources are available, the other peers in the network may access them through the recommendation mechanism. This network of

Requirements	Constructivist KM Building Blocks
allowing members to keep control of their knowledge assets while sharing knowledge	autonomy and non-hierarchical knowledge sharing structure
supporting members to ask and answer questions	social interaction and physical meaningful artifacts
providing information on experts regarding particular knowledge	context
providing personalized help to the users	context

Table 7.1: Relation between KARe's requirements and the Constructivist KM building blocks

peers is a flat structure, thus reflecting the *non-hierarchical knowledge sharing* structure we claim is necessary to guarantee effective knowledge flows within the organization. This is based on the assumption that all organization's members are valuable knowledge sources despite of their organizational position or level of experience.

KARe supports knowledge sharing mainly through questions and answers (the second requirement on table 7.1). This is related to the need to support *social interaction* and to provide sharing of *physical meaningful artifacts*. Social interaction is essential to enable tacit knowledge sharing, which is paramount for triggering innovation. Tacit knowledge relates to people's internal mental models, and personal values and experience. This is hardly captured in codified pieces of information but it is possible exchanged through direct interaction between people (Nonaka and Takeuchi, 1995). This way, KARe simulates the natural processes of seeking and providing knowledge within organizations, where people frequently try to solve their doubts and problems by directly asking a nearby or trusted colleague. In addition to that, as suggested by Freire and Fagundez (1992), a question is the "first knowledge sparkle" and is helpful for the knowledge seeker him-

self to reason about the knowledge he has and misses, hence contributing to knowledge creation since this early stage of questioning. Besides supporting the disambiguation of tacit knowledge, this requirement also refers to the need of exchanging knowledge through meaningful and concrete artifacts. In KARe, these artifacts consist the exchanged questions and answers, besides working documents the users maintain in their local knowledge bases. People working together or gathered in communities of practice profit from sharing artifact in diverse ways. For once, they feel important and receive recognition for sharing with others the product of their work or those artifacts they find valuable. Besides, these artifacts are kept in the network, allowing further use when a needing situation presents itself. And finally, artifacts may be replicated in several nodes of the network, guaranteeing that essential knowledge is maintained in the organization even if members leave it. These three aspects are all in the core of KM.

Requirements number three and four refer to the need to provide the right *context* for knowledge sharing. Simply asking people to register how they do their work may seem forced and detached from daily practices. Instead, if knowledge sharing happens as part of the daily routine of the organization, imitating natural social processes as proposed by KARe, knowledge is likely to flow more easily. In this sense, support to find the right piece of knowledge when needed is essential to motivate people to actively use the system. KARe does this by helping users locate expertise knowledge, and by providing them with personalized assistance. This is attained with the use of user models that capture social and cognitive characteristics of users, such as their organizational role, their trusted colleagues, and their expertise and interests. In particular, users' expertise and interest is explicitated through taxonomic structures used to classify users' working documents and to contextualize questions and answers. These tree-like structures describe the domains related to the knowledge artifacts maintained by a specific user, thus being directly related both to their interest and to their expertise. The taxonomies are valuable pieces of knowledge in themselves, besides serving as the foundation for the developed recommendation mechanism.

The remaining of this section specifically focuses on three sub-issues re-

lated to this general KARe-related research question.

Creating Recommendations Based on Organizational Members Social and Cognitive Aspects

One of the main assumptions behind this work is that KM should be supported with less focus on technology and more focus on people. Consequently, the peculiarities of the organizational environment and the personal characteristics of its members must be taken into account (refer to RQ 3.1).

KM systems currently in use within real organizations often give exaggerated importance to the stored knowledge assets, failing to connect them to the people that create and use such artifacts. These same systems often provide searching support based on the content of the available artifacts. Most of the systems relying on user modeling, generally limit themselves in providing hints on user's interest and expertise regarding certain content. KARe goes beyond this when, besides user's expertise and interest, it also takes into consideration the following characteristics:

- organizational role;
- trustability;
- reliability;
- availability;
- collaborative level;
- presentation preferences;
- physical context: time and location.

Some of the characteristics above are initially stated by the user of the system, such as organizational role, availability to ask and answer questions, trustability (captured in a list of trusted colleagues), and presentation preferences. Others are calculated throughout interactions with other users. The collaborative level of other peers in relation to a give user, for instance, is

calculated based on the number of responses the peers provide to the user. Besides, the peers' reliability, i.e. a measure of their expertise regarding specific themes is also attributed by the user after evaluating knowledge received by specific peers. Time and location should support the system in understanding what piece of knowledge to deliver to the user and in which presentation format. Although the proposal and design of these features are clear, their implementation in KARe remains future work. Chapter 5 presents a detailed discussion on each of these cognitive and social characteristics, besides analyzing how they were targeted in other system proposals coming from the KM and e-learning research areas.

KARe works both reactively (i.e. at user's request) and proactively (i.e. by anticipating user's needs). On on hand, reactive recommendations are triggered by the user asking a question. On the other hand, proactive help is granted through the support to proactive periodic search, pending questions (i.e. questions the users have asked and remain unanswered), and by recommending other peers having user's related interests and expertise. A comprehensive description and design of such functionalities can be found in chapter 5.

Agent-oriented Recommender System Architecture

Having learned that KARe supports Constructivist KM based on user's social and cognitive characteristics and following a peer-to-peer model, a discussion on the most appropriate system architecture comes next (RQ 3.2). This discussion is also subject of chapter 5, in which the architectural and detailed design of KARe are presented.

Three agents compose KARe: the **Artifact Manager (AM)**, the **Personal Assistant (PA)** and the **Broker**. In general, the AM and the PA collaborate to accomplish the recommendation mechanism, allowing the user to manage his/her personal knowledge base, handling other peer's knowledge requests and providing recommendations to the user. While these two agents reflect KARe's peer-to-peer model, the Broker provides a kind of centralized support, by endowing the PA with knowledge about the best peers

to answer a particular direct knowledge request (*direct* here refers to direct interaction with another peer), and aiding the PA on finding similar peers on behalf of his/her associated user. Each peer in the network has his/her own PA and AM (installed in his/her personal computer), while the Broker is not associated to any particular peer and may be installed in one or more computers.

Taking the paragraph above into consideration, we conclude that KARe is not a pure peer-to-peer architecture, but a hybrid one instead. This means that although mostly peer-to-peer, this architecture has a server element (here, the Broker) that provides supporting functionalities, generally related to locating a peer (Oram, 2001). In this sense, contrarily to other KM systems following the pure approach (such as KEEx (Bonifacio et al., 2004) for example), KARe aims at profiting from the Broker's flexibility to provide more personalized support in comparison to these systems, taking into account the cognitive and social characteristics described in the previous section. One problem with the hybrid approach relates to safety, i.e. dealing with possible failure in the accessibility of the computer hosting the Broker. Nevertheless, this problem may be overcome by the replication of the Broker in several computers to diminish the probability of non-accessibility. In addition to this, the PA may have alternative ways of handling a direct knowledge request when the Broker is unavailable.

In order to avoid intrusiveness, we elaborated a privacy policy controlling the access to the user model. According to this policy, only the PA has full view and access to the user model. The user is able to set himself the features he/she allows the Broker to view, thus limiting the access of other peers concerning his/her personal characteristics.

Developing an Effective Recommendation Technique

In the core of any recommender system is the mechanism used to provide users with valuable recommendations (RQ 3.3). KARe provides this mainly in two ways. Primarily, it allows users to access existing knowledge assets (documents, and pairs of questions and answers) available in the network of

peers. However, if no available knowledge asset fulfills the user's particular need, a peer can be directly contacted. Chapter 6 describes the algorithm we developed to recommend existing knowledge artifacts satisfying user's knowledge requests. Besides presenting the algorithm in detail, this chapter also describes how it was implemented and accessed.

The developed algorithm is based on information retrieval techniques and profits from the taxonomic structures classifying the system's knowledge artifacts. Results of the performed evaluation experiment shows that there are considerable gains when using our approach compared to a standard retrieving approach. In particular, our technique is superior in finding artifacts that are related to the user's query, obtaining a result set with less noise than the standard approach. Less noise means that less unrelated and more focused items compose the result set. Moreover, the proposed approach attains improvements concerning computational complexity, by reducing the search space of the algorithm. In other words, not all documents of the collection are inspected, but solely those pertaining to regions where a satisfactory answer is likely to be stored.

7.3 Future Work

A good research work is not the one that covers all the gaps, but the one that present consistent results while also opening way for further investigations and endeavors. We hope to have fulfilled both aims. Subsequently to the results just reported, this section is dedicated to the presentation of our future work. As this thesis presents two main contributions, i.e. the ARKnowD methodology and the KARe system, this section is correspondingly subdivided in sub-sections 7.3.1 and 7.3.2.

7.3.1 Moving Forward with the Work on ARKnowD

Further work on ARKnowD may be viewed according to theoretical and practical aspects. Theoretically, we hope to move forward with the work on the fundaments behind our methodology, given by the ontology of agent-

oriented concepts, presented in chapter 3. On one hand, we hope to cover in our ontology, the remaining concepts from the Tropos notation that were not addressed in this thesis, such as the concepts of *softgoal* and *contribution*. On the other hand, we aim at incorporating elements targeted by agent organization frameworks, especially the deontic notions of *responsibility* and *obligation*, and the concept of *norm*. This may result in adding new constructs to ARKnowD's language, possibly also affecting the methodology's life cycle.

As for the practical aspects, two main directions are identified. First, ARKnowD must undergo experimentation in real environments. The scenario applied in this thesis is fictitious, thus resulting in a very controlled testing environment. Although we were careful to be realistic and to illustrate real situations, we are sure that when applied to a real case, some of our assumptions will be confirmed, but also new insights and ideas will emerge to enhance our methodology. Still in the realm of practice, we hope the work on an agent-oriented software engineering environment comprehending both Tropos and AORML (and thus, ARKnowD) proceeds, as this initiative is of great relevance to allow analysts and designers to effectively apply our methodology in practice.

7.3.2 Future Developments on KARe

Additional implementation and experimentation work are necessary to consolidate KARe as a KM supporting system. For now, the idea behind KARe is clear and the system was fully designed, but we have only managed to implement a prototype including KARe's core recommendation mechanism. Hence, more work remains to be done, and the following items summarize KARe's missing elements.

- the Broker agent should be implemented to enable the referral regarding the best peer to respond to a particular knowledge request, enabling users to directly answer incoming questions from other peers. This comprehends the implementation of the complete user model,

including all user's cognitive and social characteristics, since only expertise and interest are considered in the available prototypes;

- the Peer Assistant (PA) should be complemented with further functionality, including proactive periodic searches, handling pending questions (which has only been implemented in the mobile prototype) and recommending similar peers to his/her associated user.

Regarding the implemented prototypes, a few gaps remain to be filled. To begin with, our recommendation technique may be enhanced by the use of *smoothing* techniques proposed by (Sona et al., 2004). Such an approach embeds more information in the representation of KARe's taxonomy concepts, thus improving at least in theory, the performance of the *query scope reduction* step of our algorithm. However, to be completely sure about this assumption, we must implement such techniques in our prototype and submit it to further experimentation.

In addition to this, scalability issues must be targeted before KARe can become a real product. At the moment, we only performed tests using two peers and we can already preview that some problems may arise if more peers are included. This issue particularly regards our desktop prototype and mainly results from the fact that when receiving a knowledge request from the user, the PA broadcasts such request to all other PAs in the network. As previously reported in the conclusion of chapter 6, we foresee two possibilities to overcome this problem. The first possibility regards the beforehand calculation of the nearest neighbor peer to answer to requests on specific subjects. Thus, the PA would know which other PAs are more likely to have the answer it seeks, being able to efficiently forward incoming knowledge requests. The other idea we could explore alternatively or in addition to this one is to set up a similarity threshold, limiting the number of documents exchanged between Peer Assistants to reduce network traffic.

Further work regarding system's experimentation also remains to be done. We have been able so far to conduct an experiment that although highly relevant, limits itself to validating the proposed recommendation technique. Nevertheless, system's usability has yet to be assessed. In this sense, we

believe the remaining functionality of KARe must be implemented before a usability test can be planned. Next to this, the nature of the tasks that KARe supports (i.e. simulating real social processes related to questioning and answering) suggest that instead of using a controlled environment, this usability test is likely to produce better results if the system is adopted in a real organization. After some time experimenting with KARe's document repository and questioning and answering functionalities, the users should be able to attest for sure the validity of the system to enhance their daily practices.

Bibliography

Abecker, A., Bernardi, A., and Sintek, M. (2000). Proactive Knowledge Delivery for Enterprise Knowledge Management. In Ruhe, G. and Bomarius, F., editors, *Learning Software Organizations: Methodology and Applications*, volume 1756 of *LNCS*. Springer-Verlag, Berlin, Germany.

Abecker, A., Bernardi, A., and van Elst, L. (2003). Agent Technology for Distributed Organizational Memories: the Frodo Project. In *Proceedings of the International Conference on Enterprise Information Systems (ICEIS'03)*, pages 3–10, Angers, France.

Adami, G., Avesani, P., and Sona, D. (2003). Clustering Documents in a Web Directory. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management (WIDM'03)*, pages 66–73, New York, USA. ACM Press.

Alavi, M. and Leidner, D. E. (1999). Knowledge Management Systems: Issues, Challenges and Benefits. *Communication of the AIS*, 1(2):1–37.

Allee, V. (1999). The Art and Practice of Being Revolutionary. *Journal of Knowledge Management*, 3(2):121–131.

Allee, V. (2000). Knowledge Networks and Communities of Practice. *OD Practitioner: Journal of the Organization Development Network*, 32.

Applewhite, A. (2004). The View from the Top. *IEEE Spectrum*, November 2004:16–31.

Audi, R. (1998). *Epistemology: A Contemporary Introduction to the Theory of Knowledge*. Routledge, London, UK.

Avesani, P., Giunchiglia, F., and Yatskevich, M. (2005). A Large Scale Taxonomy Mapping Evaluation. In Gil, Y. e. a., editor, *International Semantic Web Conference (ISWC)*, volume 3729 of *LNCS*, pages 67–81. Springer-Verlag, Berlin, Germany.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley, Boston, MA, USA.

Balabanovic, M. and Shohan, Y. (1997). Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, 40(3):66–72.

Bernon, C., Cossentino, M., Gleizes, M., Turci, P., and Zambonelli, F. (2004). A Study of Some Multi-Agent Meta-Models. In Odell, J., Giorgini, P., and Muller, J., editors, *Agent-oriented Software Engineering V, AOSE 2004*, volume 3382 of *LNCS*, pages 62–77. Springer-Verlag, Berlin, Heidelberg.

Boella, G., Damiano, R., and Lesmo, L. (1999). A Utility Based Approach to Cooperation among Agents. In *Proceedings of the Worskhop on Foundations and applications of collective agent based systems (ESSLLI'99)*, Utrecht, The Netherlands.

Bonifacio, M. and Bouquet, P. (2002). Distributed Knowledge Management: a Systemic Approach. In Minati, G. and Pessa, E., editors, *Emergence in Complex, Cognitive, Social and Biological Systems*. Kluwer Academic/Plenum Publishers, New York, USA.

Bonifacio, M., Bouquet, P., Mameli, G., and Nori, M. (2004). Peer-Mediated Distributed Knowledge Management. In van Elst, L., Dignum, V., and Abecker, A., editors, *Agent-Mediated Knowledge Management*, volume 2926 of *LNAI*, pages 31–47. Springer-Verlag, Heidelberg, Germany.

Bottazzi, E. and Ferrario, R. (2005). A Path to an Ontology of Organizations. In *Proceedings of the Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE'05)*, Enschede, The Netherlands. Centre for Telematics and Information Technology (CTIT).

Bouquet, P., Serafini, L., and Zanobini, S. (2003). Semantic Coordination: a New Approach and an Application. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *Proceedings of the Second International Semantic Web Conference*, volume 2870 of *LNCS*, pages 130–145. Springer-Verlag, Berlin, Germany.

Bratman, M. E. (1987). *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, USA.

Brazelton, J. and Gorry, G. A. (2003). Creating a Knowledge-Sharing Community: If You Build It, Will They Come? *Communications of the ACM*, 46(2):23–25.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An Agent-Oriented Software Development Methodology. *International Journal of Autonomous Agents and Multi Agent Systems*, 8(3):203–236.

Brown, J. S. (1991). Research that Reinvents the Corporation. In *Harvard Business Review on Knowledge Management*, pages 153–180. Harvard Business School Press, Boston, MA, USA.

Brown, J. S. and Duguid, P. (2000). Balancing Act: How to Capture Knowledge Without Killing it. In *Harvard Business Review on Knowledge Management*, pages 45–59. Harvard Business School Press, Boston, MA, USA.

Bull, S., Greer, J., McCalla, G., Kettel, L., and Bowes, J. (2001). User Modelling in I-Help: What, Why, When and How. In Bauer, M., Gmytrasiewicz, P., and Vassileva, J., editors, *User Modeling 2001: 8th International Conference*, volume 2109 of *LNCS*, pages 117–126. Springer-Verlag, Heidelberg, Germany.

Bunge, M. (1979). *Ontology II: A World of Systems*. D. Reidel Publishing, New York, USA.

Caire, G., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., and Massonet, P. (2001). Agent Oriented Analysis using MESSAGE/UML. In Wooldridge, M., Weiss, G., and Ciancarini, P., editors, *Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II*, volume 2222 of *LNCS*, pages 119–135. Springer-Verlag, Berlin, Germany.

Castelfranchi, C. (1995). Commitments: From Individual Intentions to Groups and Organizations. In *Proceedings of the First International Conference on Multi-Agent Systems*, Cambridge, MA, USA. AAAI-Press and MIT Press.

Castelfranchi, C. (2004). Trust Mediation in Knowledge Management and Sharing. In Jensen, C., Poslad, S., and Dimitrakos, T., editors, *Proceedings of the Second International Conference on Trust Management (iTrust'04)*, volume 2995 of *LNCS*, pages 304–318. Springer-Verlag, Berlin, Germany.

Castelfranchi, C., Cesta, A., and Miceli, M. (1992). Dependence Relations among Autonomous Agents. In Demazeau, Y. and Werner, E., editors, *Decentralized AI - 3*. Elsevier, Amsterdam, The Netherlands.

Castelfranchi, C. and Falcone, R. (1998). Towards a Theory of Delegation for Agent-Based Systems. *Robotics and Autonomous Systems*, 24(24):141–157.

Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, USA.

Chen, H. and Dhar, V. (1989). Online Query Refinement on Information Retrieval Systems: A Process Model of Searcher/System Interactions. In

Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, pages 115–133, New York, NY, USA. ACM Press.

Chen, J. Q., Lee, T. E., Zhang, R., and Zhang, Y. J. (2003). Systems Requirements for Organizational Learning. *Communications of the ACM*, 46(12):73–78.

Chung, L. K., Nixon, B. A., Yu, E., and Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering*. Kluwer Publishing, New York, NY, USA.

Conklin, J. (1997). Designing Organizational Memory: Preserving Intellectual Assets in a Knowledge Economy. Technical report, CogNexus Institute, USA.

Dam, K. H. and Winikoff, M. (2003). Comparing Agent-oriented Methodologies. In Giorgini, P., Henderson-Sellers, B., and Winikoff, M., editors, *Agent-Oriented Information Systems: 5th International Bi-Conference Workshop*, volume 3030 of *LNAI*, pages 78–93. Springer-Verlag, Berlin, Germany.

Davenport, T. H. and Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston, MA, USA.

Davies, J., Duke, A., and Stonkus, A. (2003a). OntoShare: Evolving Ontologies in a Knowledge Sharing System. In Davies et al. (2003b), pages 161–177.

Davies, J., Fensel, D., and van Harmelen, F., editors (2003b). *Towards the Semantic Web: Ontology-driven Knowledge Management*. Wiley, West Sussex, UK.

de La Taille, Y. (1992). O Lugar da Interacao Social na Concepcao de Jean Piaget (in Portuguese) (The Role of Social Interaction in the Conception of Jean Piaget). In de La Taille, Y., Oliveira, M. K., and Dantas, H.,

editors, *Piaget, Vygotsky e Wallon, Teorias Psicogenticas em Discusso (in Portuguese) (Piaget, Vygotsky and Wallon: A Discussion of Psychogenetic Theories)*, pages 11–21. Summun, Sao Paulo, SP, Brazil, 13 edition.

Delaitre, S. and Moisan, S. (2000). Knowledge Management by Reusing Experience. In *12th International Conference EKAW2000*, volume 1937 of *LNAI*, pages 304–311. Springer-Verlag, Berlin, Germany.

Desouza, K. C. (2003). Barriers to Effective Use of Knowledge Management Systems in Software Engineering. *Communications of the ACM*, 46(1):99–101.

Di Marzo Serugendo, G., Karageorgos, A., Rana, O. F., and Zambonelli, F. E. (2004). *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, volume 2977 of *LNAI*. Springer-Verlag, Berlin, Germany.

Dignum, V. (2004a). *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, Utrecht University, The Netherlands.

Dignum, V. (2004b). An Overview of Agents in Knowledge Management. Technical Report UU-CS-2004-017, Institute of Information and Computing Sciences, Utrecht University, The Netherlands.

Dignum, V., Sonenberg, L., and Dignum, F. (2004). Dynamic Reorganization of Agent Societies. In *Proceedings of the Workshop on Coordination in Emergent Agent Societies at ECAI'04*, Valencia, Spain.

Dignum, V. and van Eeden, P. (2003). Seducing, Engaging and Supporting communities at Achmea. In *Proceedings of the 4th European Conference on Knowledge Management, Oxford, UK*, Oxford, UK.

Doyle, M. D., Ang, C. S., Martin, D. C., and Noe, A. (1996). The visible embryo project: Embedded program objects for knowledge access

creation and management through the world wide web. *Computarized Medical Imaging and Graphics*, 20(6):423–431.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Inteligence*. MIT Press, Cambridge, MA, USA.

Ellis, C. A., Gibbs, S. J., and Rein, G. L. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):38–58.

Esfandiari, B. and Chandrasekharan, S. (2001). On How Agents Make Friends: Mechanisms for Trust Acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, pages 27–34, Montreal, Canada.

Esteva, M., Padget, J., and Sierra, C. (2002). Formalizing a Language for Institutions and Norms. In Meyer, J.-J. C. and Tambe, M., editors, *Intelligent Agents VIII*, volume 2333 of *LNAI*, page 348366. Springer-Verlag, Berlin, Germany.

Farias, C. R. G. (2002). *Architectural Design of Groupware Systems: a Component-Based Approach*. PhD thesis, University of Twente, The Netherlands.

Fensel, D., Staab, S., Studer, R., van Harmelen, F., and Davies, J. (2003). A Future Perspective: Exploiting Peer-2-Peer and the Semantic Web for Knowledge Management. In Davies, J., Fensel, D., and van Harmelen, F., editors, *Towards the Semantic Web: Ontology-driven Knowledge Management*, pages 245–264. Wiley, West Sussex, UK.

Ferber, J., Gutknecht, O., and Michel, F. (2004). From Agents to Organizations: An Organizational View of Multi-agent Systems. In Giorgini, P., Mller, J. P., and Odell, J., editors, *AOSE 2003*, volume 2935 of *LNCS*, page 214230. Springer-Verlag, Berlin, Germany.

Fischer, G. and Ostwald, J. (2001). Knowledge Management: Problems, Promises, Realities, and Challenges. *IEEE Intelligent Systems*, 16(1):60–72.

Fischer, K., Muller, J. P., and Pischel, M. (1996). A Pragmatic BDI Architecture. In Wooldridge, M., Muller, J. P., and Tambe, M., editors, *Intelligent Agents II: Agent Theories, Architectures and Languages*, volume 1037 of *LNAI*, pages 203–218. Springer, Berlin, Germany.

Freire, P. (1970). *Pedagogy of the Oppressed*. Continuum Intl Pub Group, New York, NY, USA.

Freire, P. and Fagundez, A. (1992). *Learning to Question: A Pedagogy of Liberation*. Continuum Intl Pub Group, New York, NY, USA.

Gandon, F., Poggi, A., Rimassa, G., and Turci, P. (2002). Multi-Agent Corporate Memory Management System. *Journal of Applied Artificial Intelligence*, 16(9-10):699–720.

Gangemi, A., Prisco, A., Sagri, M., Steve, G., and Tiscornia, D. (2003). Some Ontological Tools to Support Legal Regulatory Compliance, with a Case Study. In Meersman, R. and Zahir, T., editors, *On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*, volume 2889 of *LNCS*, pages 607–620. Springer-Verlag, Berlin, Germany.

Garcia-Barrios, V. M., Gutl, C., Preis, A. M., Andrews, K., and Pivec, M. (2004). AdELE: A Framework for Adaptive E-Learning through Eye Tracking. In *Proceedings of the 4th International Conference on Knowledge Management (I-Know'04)*, pages 609–616, Graz, Austria.

Garvin, D. A. (1993). Building a Learning Organization. In *Harvard Business Review on Knowledge Management*, pages 47–80. Harvard Business School Press, Boston, MA, USA.

Giorgini, P., Mylopoulos, J., and Sebastiani, R. (2005). Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology. *In Engineering Applications of Artificial Intelligence*, 18(2).

Goguen, J. A. and Linde, C. (1993). Techniques for Requirements Elicitation. In *Proceedings of Requirements Engineering '93*, pages 152–164, Piscataway, NJ, USA. IEEE Computer Society.

Gongla, P. and Rizzuto, C. R. (2001). Evolving communities of practice: IBM Global Services experience. *IBM Systems Journal*, 40(4):842–862.

Good, N., Schafer, J. B., Konstan, J., Borchers, A., Herlocker, B., and Riedl, J. (1999). Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of the 1999 Conference of the Americian Association of Artificial Intelligence (AAAI'99)*, pages 439–446, Cambridge, MA, USA. MIT Press.

Gruninger, M., Atefi, K., and Fox, M. (2000). Ontologies to Support Process Integration in Enterprise Engineering. *Computational and Mathematical Organization Theory*, 6(4):381–394.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. PhD thesis, University of Twente, The Netherlands.

Guizzardi, G. and Wagner, G. (2005). Some Applications of a Unified Foundational Ontology in Business Modeling. In Rosemann, M. and Green, P., editors, *Ontologies and Business Systems Analysis*, pages 345–367. Idea Group, London, UK.

Guizzardi, R., Dignum, V., Perini, A., and Wagner, G. (2005). Towards an Integrated Methodology to Develop KM Solutions with the Support of Agents. In *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS'05)*, pages 221–226, Boston, MA, USA. IEEE.

Guizzardi, R. and Perini, A. (2005). Analyzing Requirements of Knowledge Management Systems with the Support of Agent Organizations. *Journal of the Brazilian Computer Society (JBCS) - Special Issue on Agents Organizations*, 11(1):51–62.

Guizzardi, R. S. S., Aroyo, L., and Wagner, G. (2004a). Agent-oriented Knowledge Management in Learning Environments: A Peer-to-Peer Helpdesk Case Study. In van Elst, L., Dignum, V., and Abecker, A., editors, *Agent-Mediated Knowledge Management*, volume 2926 of *LNAI*, pages 57–72. Springer-Verlag, Heidelberg, Germany.

Guizzardi, R. S. S., Perini, A., and Dignum, V. (2003). Using Intentional Analysis to Model Knowledge Management Requirements in Communities of Practice. Technical Report TR-CTIT-03-53, Centre for Telematics and Information Technology (CTIT), The Netherlands.

Guizzardi, R. S. S., Perini, A., and Dignum, V. (2004b). Providing Knowledge Management Support to Communities of Practice through Agent-oriented Analysis. In *Proceedings of the 4th International Conference on Knowledge Management, Graz, Austria (I-Know'04)*, pages 320–328, Graz, Austria.

Hahn, J. and Subramani, M. R. (2000). A Framework of Knowledge Management Systems: Issues and Challenges for Theory and Practice. In *Proceedings of the 21st International Conference on Information systems (ICIS '00)*, pages 302 – 312, Atlanta, GA, USA. Association for Information Systems.

Hansen, M. T., Nohria, N., and Tierney, T. (1999). What's Your Strategy for Managing Knowledge. In *Harvard Business Review on Knowledge Management*, pages 61–86. Harvard Business School Press, Boston, MA, USA.

Hassan, I. M., Rafea, A. A. E., and Rasmy, M. (2004). Configuration Irrigation Schedule Based on Expert Systems and Operations Research. In *Proceedings of the Fifth International Workshop on Artificial Intelligence in Agriculture (AIA'04)*, Oxford, UK. Elsevier.

Henderson-Sellers, B. (2005). Creating a Comprehensive Agent-Oriented Methodology: Using Method Engineering and the OPEN Metamodel. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 368–397. Idea Group, London, UK.

Henninger, S. (2001). Turning Development Standards into Repositories of Experiences. *Software Process: Improvement and Practice*, Volume 6(3):141–155.

Hubner, J. F., Sichman, J. S., and Boissier, O. (2002). A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In Bittencourt, G. and Ramalho, G. L., editors, *Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, volume 2507 of *LNAI*, pages 118–128. Springer-Verlag, Berlin, Germany.

Iglesias, C. A., Garijo, M., and Gonzalez, J. C. (1999). A Survey of Agent-Oriented Methodologies. In Muller, J., Singh, M., and Rao, A., editors, *Intelligent Agents V: Agents Theories, Architectures and Languages*, volume 1555 of *LNCS*, pages 317–330. Springer-Verlag, London, UK.

Iglesias, C. A., Garijo, M., Gonzalez, J. C., and Velasco, J. R. (1998). Analysis and Design of Multiagent Systems Using MAS-CommonKADS. In Singh, M., Rao, A., and Wooldridge, M., editors, *Intelligent Agents IV*, volume 1365 of *LNAI*, pages 313–326. Springer-Verlag, Berlin, Germany.

Jennings, N. R., Sycara, K. P., and Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–36.

Juan, T., Pearce, A., and Sterling, L. (2002). ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 3–10, New York, USA. ACM Press.

Juan, T., Sterling, L., Martelli, M., and Mascardi, V. (2003). Customizing AOSE Methodologies by Reusing AOSE Features. In *Proceedings of the 2nd Iinternational Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 113–120, New York, USA. ACM Press.

Juan, T., Sterling, L., and Winikoff, M. (2004). Assembling Agent Oriented Software Engineering Methodologies from Features. In

Giorgini, P., Muller, J. P., and Odell, J., editors, *AOSE 2003*, volume 2935 of *LNCS*, pages 198–209. SpringerVerlag, Berlin, Germany.

Kankanhalli, A., Tanudidjaja, F., Sutanto, J., and Tan, B. C. Y. (2003). The Role of IT in Successful Knowledge Management Initiatives. *Communications of the ACM*, 46(9):69–73.

Karlin, S. (2004). Companies find new ways to harness their engineers creativity. *IEEE Spectrum*, November 2004:67–68.

Kavakli, E. and Loucopoulos, P. (2005). Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. In Krogstie, J., Halpin, T., and Siau, K., editors, *Information Modeling Methods and Methodologies*, pages 102–124. Idea Group, London, UK.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction*. Addison-Wesley, Boston, MA, USA.

Labrou, Y., Finin, T., and Peng, Y. (1999). Agent Communication Languages: The Current Landscape. *IEEE Intelligent Systems*, 14(2):45–52.

Lave, J., Wenger, E., Pea, R., Brown, J. S., and Heath, C. (1991). *Situated Learning: Legitimate Peripheral Participation. Learning in Doing: Social, Cognitive & Computational Perspectives*. Cambridge University Press, Cambridge, MA, USA.

Lehner, F., Maier, R., and Klosa, O. (1998). Organisational Memory Systems: Application of Advanced Database & Network Technologies in Organisations. In *Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM'98)*, pages 14/1–14/12, Basel, Switzerland. CEUR-Ws.org.

Loucopoulos, P. and Kavakli, E. V. (1999). Enterprise Knowledge Management and Conceptual Modelling. In *Selected Papers from the Symposium on Conceptual Modeling, Current Issues and Future Directions*, volume 1565 of *LNCS*, pages 123–143. Springer-Verlag.

Ludermir, P. G. (2005). Supporting Knowledge Management using a Nomadic Service for Artifact Recommendation. Master's thesis, University of Twente, The Netherlands.

Ludermir, P. G., Guizzardi, R. S. S., and Sona, D. (2005). Finding the Right Answer: An Information Retrieval Approach Supporting Knowledge Sharing. In *Proceedings of the Workshop on Agent Mediated Knowledge Management (AMKM'05)*.

Luger, G. F. (2005). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, Boston, MA, USA, 5th edition.

Mahoney, M. (2004). What is constructivism and why is it growing? *Contemporary Psychology*, 49:360–363.

Mantovani, G. (1996). Social Context in HCI: A New Framework for Mental Models, Cooperation, and Communication. *Cognitive Science*, 20:237–269.

Montaner, M., Lopez, B., and de la Rosa, J. L. (2003). A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, 19:285–330.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., and Risch, T. (2002). EDUTELLA: P2P Networking Infrastructure Based on RDF. In *Proceedings of 11th World Wide Web Conference (WWW2002)*, pages 604–615, New York, USA. ACM Press.

Newell, S., Scarbrough, H., Swan, J., and Hislop, D. (1999). Intranets and Knowledge Management: Complex Processes and Ironic Outcomes. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Piscataway, NJ, USA. IEEE Press.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, New York, USA.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In *Proceedings of International Conference on Software Engineering (ICSE-2000)*, New York, USA. ACM Press.

Odell, J., Parunak, H. V. D., and Bauer, B. (2000). Extending UML for Agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pages 3–17, Austin, TX, USA.

O'Leary, D. E. (1998). Enterprise Knowledge Management. *IEEE Computer*, 31(3):54–61.

Oram, A., editor (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, Sebastopol, CA, USA.

Orlikowski, W. J. (1992a). Learning from Notes: Organizational Issues in Groupware Implementation. In *Proceedings of The International Conference on Computer Supported Cooperative Work (CSCW92)*, pages 362–369, New York, NY, USA. ACM Press.

Orlikowski, W. J. (1992b). The Duality of Technology: Rethinking the Concept of Technology in Organizations. *Organizational Science*, 3(3):398–427.

Orlikowski, W. J. and Gash, D. C. (1994). Technological Frames: Making Sense of Information Technology in Organizations. *ACM Transactions on Information Systems*, 12(2):174–207.

Orlikowski, W. J., Walsham, G., Jones, M., and DeGross, J. I., editors (1995). *Information Technology and Changes in Organizational Work, Proceedings of the IFIP WG8.2 Working Conference.* Chapman and Hall, London, UK.

Padgham, L. and Winikoff, M. (2002). Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents. In *Proceedings of the workshop on Agent-oriented methodologies at OOPSLA'02*, Seattle, USA.

Papert, S. (1993). *The Children's Machine: Rethinking School in the Age of the Computer*. BasicBooks, New York, NY, USA.

Parunak, H. V. D. (2000). Agents in Overalls: Experiences and Issues in the Development and Deployment of Industrial Agent-Based Systems. *International Journal of Cooperative Information Systems*, 9(3):209–228.

Pedersen, K. V. (2004). Context Based Support for Clinical Reasoning. In *Proceedings of the 4th International Conference of Knowledge Management (I-Know'04)*, pages 397–404, Graz, Austria.

Perini, A., Bresciani, P., Yu, E., and Molani, A. (2004). Intentional Analysis for Distributed Knowledge Management. In van Elst, L., Dignum, V., and Abecker, A., editors, *Agent-Mediated Knowledge Management*, volume 2926 of *LNAI*, pages 351–367. Springer-Verlag, Heidelberg, Germany.

Perini, A. and Susi, A. (2004). Developing Tools for Agent-Oriented Visual Modeling. In Lindemann, G., Denzinger, J., Timm, I., and Unland, R., editors, *Multiagent System Technologies, MATES 2004*, volume 3187 of *LNCS*, pages 169–182. Springer-Verlag.

Piaget, J. and Inhelder, B. (1969). *The Psychology of the Child*. Basic Books, New York, USA.

Preece, A., Hui, K., Gray, A., Marti, P., Bench-Capon, T., Cui, Z., and Jones, D. (2001). KRAFT: An Agent Architecture for Knowledge Fusion. *International Journal on Cooperative Information Systems*, 10(1 and 2):171–195.

Pumareja, D., Bondarouk, T., and Sikkel, K. (2003). Supporting Knowledge Sharing Isn't Easy - Lessons Learnt from a Case Study. In *Proceedings of the Information Resource Management Association International Conference (IRMA'03), Philadelphia, USA*, pages 531–534, Philadelphia PA, USA.

Pynadath, D. V., Tambe, M., Chauvat, N., and Cavedon, L. (1999). Toward Team-Oriented Programming. In Jennings, N. R. and Lesprance, Y., editors, *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, volume 1757 of *LNCS*, pages 233–247. Springer-Verlag, Berlin, Germany.

Quinn, J. B., Anderson, P., and Finkelstein, S. (1996). Managing Professional Intellect: Making the Most of the Best. In *Harvard Business Review on Knowledge Management*, pages 181–205. Harvard Business School Press, Boston, MA, USA.

Rao, A. S. and Georgeff, M. P. (1991). Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, Cambridge, MA, USA. Morgan Kaufmann Publishers.

Reimer, U., Brockhausen, P., Lau, T., and Reich, J. R. (2003). Ontology-based Knowledge Management at Work: The Swiss Life Case Studies. In Davies, J., Fensel, D., and van Harmelen, F., editors, *Towards the Semantic Web: Ontology-Driven Knowledge Management*, pages 197–218. Wiley, West Sussex, England.

Robertson, M., Sorensen, C., and Swan, J. (2000). Facilitating Knowledge Creation with Groupware: A Case Study of a Knowledge Intensive Firm. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Piscataway, NJ, USA. IEEE Press.

Sabas, A., Delisle, S., and Badri, M. (2002). A Comparative Analysis of Multiagent System Development Methodologies: Towards a Unified Approach. In *Proceedings of the 16th European Meeting on Cybernetics and Systems Research*, pages 599–604, Vienna, Austria.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, NY, USA.

Santos, L. O. B. S., Guizzardi, R. S. S., and van Sinderen, M. (2005a). Agent-Oriented Approach to Develop Context-Aware Applications: A

Case Study on Communities of Practice. Technical Report TR-CTIT-05-20, Centre for Telematics and Information Technology (CTIT), The Netherlands.

Santos, L. O. B. S., Guizzardi, R. S. S., and van Sinderen, M. (2005b). Agent-Oriented Context-Aware Platforms Supporting Communities of Practice in Health Care. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS' 05)*, pages 1287–1288, New York, USA. ACM Press.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, Cambridge, MA, USA.

Sen, S. and Weiss, G. (1999). Learning in Multiagent Systems. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 259–298. MIT Press, Cambridge, MA, USA.

Shoham, Y. (1993). Agent-oriented Programming. *Artificial Intelligence*, 60:5192.

Sichman, J. S., Dignum, V., and Castelfranchi, C. (2005). Agents' Organizations: A Cocise Overview. *Journal of the Brazilian Computer Society (JBCS) - Special Issue on Agents Organizations*, 11(1):3–8.

Soller, A., Guizzardi, R. S. S., Molani, A., and Perini, A. (2004). SCALE: Supporting Community Awareness, Learning and Evolvement in an Organizational Learning Environment. In *Proceedings of the 6th International Conference of the Learning Sciences*, pages 489–496, Santa Monica, CA, USA.

Sona, D., Veeramachaneni, S., Avesani, P., and Polettini, N. (2004). Clustering with Propagation for Hierarchical Document Classification. Technical Report T04-04-05, ITC-IRST, Italy.

Sumner, M. (1999). Knowledge Management: Theory and Practice. In *Proceedings of the Special Interest Group on Computer Personnel Research Annual Conference (SIGCPR'99)*, pages 1–3, New York, NY, USA. ACM Press.

Taveter, K. and Wagner, G. (2005). Towards Radical Agent-Oriented Software Engineering Processes Based on AOR Modelling. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 277–316. Idea Group, London, UK.

Tiwana, A. (2003). Affinity to Infinity in Peer-to-Peer Knowledge Platforms. *Communications of the ACM*, 46(5):76–80.

Uiterkamp, E. S. (2005). Nomadic Positioning Services for a Mobile Service Platform. Master's thesis, University of Twente, The Netherlands.

van Elst, L., Abecker, A., and Maus, H. (2001). Exploiting User and Process Context for Knowledge Management Systems. In *Proceedings of the Workshop on User Modeling for Context-Aware Applications*, Sonthofen, Germany.

van Elst, L., Dignum, V., and Abecker, A. (2004). Towards Agent-Mediated Knowledge Management. In van Elst, L., Dignum, V., and Abecker, A., editors, *Agent-Mediated Knowledge Management*, volume 2926 of *LNAI*, pages 1–30. Springer-Verlag, Heidelberg, Germany.

van Lamsweerde, A., Dardenne, A., Delcourt, B., and Dubisy, F. (1991). The KAOS Project: Knowledge Acquisition in Automated Specification of Software. In *Proceedings of the AAAI Spring Symposium Series*, pages 59–62, Cambridge, MA, USA. MIT Press.

Vassileva, J. (2002). Supporting Peer-to-Peer User Communities. In Meersman, R. and Tari, Z., editors, *CoopIS/DOA/ODBASE 2002*, volume 2519 of *LNCS*, pages 230–247. Springer-Verlag, Berlin, Germany.

Vygotsky, L. (1978). *Mind in Society*. Harvard University Press, Cambridge, MA, USA.

Wagner, G. (2003). The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. *Information Systems*, 28(5):475–504.

Wagner, G. (2005). AOR Modelling and Simulation: Towards a General Architecture for Agent-Based Discrete Event Simulation. In Bresciani, P., Giorgini, P., Henderson-Sellers, B., Low, G., and Winikoff, M., editors, *Agent-Oriented Information Systems: 5th International Bi-Conference Workshop, AOIS 2003*, volume 3030 of *LNAI*, pages 174–188. Springer-Verlag, Berlin, Germany.

Weiser, M. (1994). The World is not a Desktop. *ACM Interactions*, 1(1):7–8.

Weiss, G., editor (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning and Identity*. Cambridge University Press, New York, USA.

Wiig, K. M. (1994). *Knowledge Management: The Central Management Focus for Intelligent-Acting Organizations*. Schema Press, Arlington, TX, USA.

Winograd, T. (1995). From Programming Environments to Environments for Design. *Communications of the ACM*, 38(6):65–74.

Wooldridge, M. (2002). Methodologies. In *An Introduction to Multiagent Systems*, pages 225–244. Wiley & Sons, Chichester, England.

Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.

Wooldridge, M. J. (1999). Intelligent Agents. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–77. MIT Press, Cambridge, MA, USA.

Wooldridge, M. J. and Ciancarini, P. G. (2001). Agent-Oriented Software Engineering: The state of the art. In Ciancarini, P. G. and Wooldridge, M. J., editors, *AOSE 2000*, volume 1957 of *LNCS*, pages 1–25. Springer-Verlag, Berlin, Germany.

Wooldridge, M. J. and Jennings, N. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152.

Yen, J., Yin, J., Ioerger, T. R., Miller, M. S., Xu, D., and Volz, R. A. (2001). CAST: Collaborative Agents for Simulating Teamwork. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1135–1144, Seattle, WA, USA. Morgan Kaufmann.

Yin, J., Miller, M. S., Ioerger, T. R., Yen, J., and Volz, R. A. (2000). A Knowledge-based Approach for Designing Intelligent Team Training Systems. In *Proceedings of Agents'00*, pages 427–434, New York, USA. ACM Press.

Yu, B. and Singh, M. P. (2002). An Agent-based Approach to Knowledge Management. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM '02)*, pages 642–644, New York,USA. ACM Press.

Yu, E. (1995). *Modeling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Canada.