Towards an Integrated Methodology to Develop KM Solutions with the Support of Agents

Renata S. S. Guizzardi[1], Virginia Dignum[2], Anna Perini[3], Gerd Wagner[4]
[1]Computer Science Department, University of Twente, Enschede – The Netherlands
Email: souza@cs.utwente.nl

[2] Institute of Information and Computing Sciences, Utrecht University – The Netherlands
Email: virginia@cs.uu.nl

[3] ITC-irst, Trento-Povo – Italy
Email: perini@itc.it

[4] Institute of Informatics, Brandenburg University of Technology at Cottbus – Germany
Email: G.Wagner@tu-cottbus.de

Abstract – *Activities related to Knowledge Management (KM) processes require changes within the organizational processes in order to accommodate new goals and tasks, besides changing the way people view and do their work. In this paper, we argue that agents are suitable for modeling KM contexts, due to their cognitive characteristics, such as goals, beliefs, and reactivity. However, besides a good abstraction, the development of adequate solutions requires a consistent software engineering methodology. To help fill in this gap, we describe some results of our work on an integrated agent-oriented methodology to develop KM system, using a KM typical scenario. We consider all humans, organizations and existing systems as agents. The model enables the designer to understand and describe relations between entities before development of the system. Furthermore, different cognitive characteristics of agents are used in different phases of the development cycle.*

1. INTRODUCTION

Organizational processes have gone through profound changes in the past years, becoming more dynamic and knowledge intensive. These changes have been necessary for businesses to maintain sustainable advantage on the market. On the other hand, the new business models brought about a wide variety of problems to be solved, and a solution is many times based on the development of new information systems. In parallel, the software engineering field faces a big challenge in providing the right kinds of abstractions and methods to model such systems.

Knowledge Management (KM) refers to the processes of creating, codifying, sharing, maintaining and evolving knowledge within an organization. Activities related to these processes tend to require changes within the organizational processes to accommodate new goals and tasks, besides changing the way people view and do their work.

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 information systems designers in understanding the current organizational setting before proposing the development or adoption of particular supporting systems. However, having an appropriate abstraction is not enough for guaranteeing the development of adequate solutions for the organization. For that, a consistent software engineering methodology is needed. Our main focus here regards 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 before actually thinking of developing a system.

Benefits as a result of the application of this methodology may be attributed to our choice of using the proper agent cognitive characteristics in the different phases of the development cycle. Concepts such as agent's beliefs, goals, and plans are vastly discussed in literature and different models have been proposed. However, it is hard to know how to go from theory to 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? 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, by integrating two existing modeling approaches: the Tropos methodology [2] and the Agent-Object-Relationship Modeling Language (AORML) [9]. A consistent transformation process guides the designer on converting their different notations, merging both approaches.

This paper illustrates the proposed methodology, *ARKnowD* (Agent-oriented Recipe for Knowledge Management System Development), using a KM typical scenario. The remaining paper is organized as follows: section 2 presents information systems as KM enabling technology, describing the scenario we use as an example for our modeling approach; section 3 introduces ARKnowD's main principles, presents Tropos and AORML, and proposes a transformation method between these two approaches; in section 4, the architectural design of the scenario is presented and detailed; finally, section 5

concludes the paper and presents some directions for future work.

2. KNOWLEDGE COMMUNITIES

Information systems are key ingredients of a KM solution, playing the role of "enabling technology". Most of the currently adopted KM systems rest on a centralized repository of documents, organized around a single ontology, or requiring the adoption of 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. In addition to that, a centralized view of KM processes are in contrast with results coming from organizational and cognitive studies, which support a distributed KM paradigm that better fits to the distributed, subjective and inter-subjective nature of organizational knowledge [1] The distributed view of KM is based on the assumption that the members of an organizations have their own natural way to share knowledge. They usually gather in groups, based on similar interests, personal affinity and trust. These groups are commonly known as Communities of Practice (CoPs) [10].

The role of CoPs in KM characterizes the scenario that we propose in this paper to illustrate our approach. As emerging from this scenario, communities may be fostered by the organization management, which provides these communities with incentives and support them with appropriate information systems and infrastructures. Deciding which technology could enable CoPs' creation and management becomes a critical issue, which requires a deep understanding of the organization, of the common and the individual goals of its members. For example, the scenario illustrates the case of a company's newcomer who would like to understand better about the content as well as the procedures that should be used in his work. Knowing about the existence of CoPs, this newcomer decides to join one in order to share knowledge with the other workers, thus becoming a CoP member.

The design of a CoP support system must take into account the different perspectives of the newcomer, the CoP and the organization's management. Previous requirements analysis of this scenario [3] has lead us to the proposal of a socially-aware recommender agent named KARe (Knowledgeable Agent for Recommendations). KARe's main feature regards finding, in a peer-to-peer distributed base, possible answers to one's knowledge explanation requests (i.e. natural language questions), whenever possible. The types of artifact contained in this distributed base are both messages (i.e. answers given in natural language by the network peers) and documents (such as articles, spreadsheets, etc.). Consequently, when a previous message or document is not found to answer to an explanation request, KARe refers to someone as the most able peer to fulfill that specific request.

3. AGENT-ORIENTED KM DEVELOPMENT

Towards facilitating the analysis of KM scenarios for the consequent development of adequate solutions, this work proposes *ARKnowD* (Agent-oriented Recipe for Knowledge Management Systems Development). Note that 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. brainstorming, creativity workshops).

The basic philosophical assumptions behind ARKnowD are: a) the interactions between human and system should be understood according to *structuration theory* [7], which claims that humans and communities are self-organizing entities, and that structuration and re-structuration are motivated by human-system interaction cycles, in which 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) also need to be targeted; c) there is no *silver bullet* when pursuing a KM tailoring methodology, so the best approach is combining existing work according to the given domain or situation. Here, we particularly adopt the agent-oriented paradigm, for understanding that agents are appropriate metaphors to represent humans, organizations, and technology in a KM scenario.

In this paper, we show how the principles above may be achieved by the integration of two existing work on agent-oriented software engineering: the *Tropos methodology* [2] for Requirements Engineering and the *Agent-Object-Relationship Modeling Language* [9]. These two approaches are merged to guide KM analysts and system developers when conceiving KM solutions. This combination allows the support of *distributed* approaches to KM, pointed-out in the previous section as preferable, in contrast to centralized solutions. Our integrated methodology emphasizes the earlier phases of software development, the so-called *requirement analysis phase*. 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. This analysis may conclude, for example, that the problems in the domain 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. Hence, a new technological solution (if needed) may be developed on top of legacy systems. This may lead to more satisfaction to end users, who are already familiar with the interface and methods applied in the systems in use. Besides, 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, providing the analyst and designer with a way of how to go from theoretical definitions to practical analysis and design elements.

In this respect, ARKnowD prescribes that: 1) in the requirements analysis phases, the main focus should be on the agent's *goals*; and 2) in the phases of architectural and detailed design, the following agent's concepts should be considered: plans, resources, beliefs, commitments, claims, reactiveness, proactiveness, autonomy, and social ability. This choice may be justified by intuition, i.e. in our daily lives, goals are the motivators of our actions, besides determining if we use specific resources instead of others, if we talk to specific people, etc. In other words, goals determine the processes an agent should pursuit. But this is also supported by KM literature. For instance, Nonaka & Takeuchi [5] mention *intention* as the first driving force for the adoption of KM practices within organizations. Nevertheless, these authors mainly focus on the organization top management's intention, defined as "an organization's aspiration to its goals", facilitating KM initiatives. Here, in contrast, we consider the goals of all stakeholders involved, trying to understand the relations and possible discrepancies between these goals. While in the early phases of domain analysis, we may abstract from all details, placing our attention mainly in understanding the goals of the stakeholders (having also a feeling of the relations between the goals and the agents), the design phase is dedicated to fill all the gaps and provide us with the biggest amount of details needed for system implementation.

3.1. Tropos Methodology

The *Tropos* methodology [2] uses visual modeling language and a set of techniques for goal analysis. Basic constructs of the conceptual modeling language are: *actor*, representing a stakeholder in a given domain, a *role* or a set of roles played by an agent in a given organizational setting, and actor's *goal*, *plan* (or task) and *resource*. Moreover, a *dependency link* between pairs of actors allows to model the fact that one actor depends on another in order to achieve a goal, execute a plan, or acquire a resource. Goal analysis is conducted from the point of view of each individual actor, that is for each actor's goal, we may consider: means to satisfy it (*means-end* relationship); alternative ways to achieve it (*OR* decomposition); possible sub-goals (*AND* decomposition); goals or plans or resources that can contribute positively or negatively to its achievement (*contribution*). This type of information can be graphically depicted in *actor* and *goal diagrams*. Linear temporal logic specification can be used to constraint a model behavior (we refer it as Formal Tropos - FT annotation)

Among the advantages of adopting *Tropos* visual modeling for KMS requirements analysis is the possibility of pointing out the idiosyncrasies of a given environment, as, for instance: a) verifying inconsistencies between models elaborated on the basis on interviews with different actors in the organization; b) realizing that several actors perform the same exact task, thus suggesting that the process can be more efficient if that task is attributed to only one or two actors; c) understanding that too much or too little time and effort are dedicated to KM activities; and d) realizing the problems behind the non-adoption of proposed KM methods and systems [10], i.e. detachment of the system from the daily practices of organizational members, lack of trust and motivation to share knowledge, etc..

3.2. AORML

The Agent-Object-Relationship (AOR) modeling approach [9] is based on an ontological distinction between active and passive entities, that is, between agents and objects. This helps to capture the semantics of complex processes, having agents represent the actors of a given scenario, and objects playing the role of the artifacts manipulated by the domain actors.

In AORML, an entity can be an agent, an event, an action, a claim, a commitment, or an ordinary object. Agent and object 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. Besides AOR models human, artificial and 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 us to model the rights and duties of their internal agents. For further reference, we refer to [9] and to the AOR website: http://aor.rezearch.info/.

3.3. Integrating Tropos and AORML

Reading about Tropos' and AORML notation provides a first feeling on how these two approaches will be applied to fulfill proposals 1 and 2 mentioned above, i.e. using goals in the analysis, and the remaining agents content on design. Table 1 shows ARKnowD viewpoints framework that can be defined as a technique for suppressing unnecessary details according to different abstraction levels, providing us with an appropriate separation of concerns regarding system analysis and design [6].

Table 1 shows for each abstraction level [6], which models are used and for 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. On the other hand, the division in three abstraction levels provide us with an interesting view, showing us that we naturally should target the modeling task from different perspectives: the domain model (CDM), a design model which can 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). Regarding the PSM, if we use Java, we may use UML Class Diagrams for this last abstraction level. On the other hand, if we apply JADE or other agent-oriented framework, we must use other models

that comply with the constructs provided by this specific framework.

Table 1 - ARKnowD Viewpoints

	Viewpoint Aspects		
Abstraction level	*Interaction*	*Information*	*Behavior*
Conceptual Domain Modeling (CDM)	Tropos Actor Diagram, Tropos Goal Diagram	Tropos Actor Diagram, Tropos Goal Diagram	Tropos Actor Diagram, Tropos Goal Diagram with FT annotations
Platform-independent Computational Design (PIM)	Tropos Actor Diagram, Tropos Goal Diagram AOR Interaction Sequence Diagrams, AOR Agent Pattern Diagrams	Tropos Actor Diagram, AOR Interaction Sequence Diagrams, UML Class Diagrams	Tropos Actor Diagram, Tropos Goal Diagram with FT annotations, AOR Agent Pattern Diagrams, AOR Internal Activity Diagrams
Platform Specific Implementation (PSM)	UML Deployment Diagrams, others	UML Class Diagrams, others	UML Class Diagrams, others

Among the advantages of adopting two existing work, for instance, Tropos and AORML, is the existence of supporting tools for both languages. But besides that, there should be some kind of transformation model that can be applied in order to go from Tropos to AORML in a consistent way. In this way, we should provide the analysts and designers with some guidelines on how to convert one to the other, even using semi-automatic process for this transformation. Table 2 depicts the transformation rules, meaning that a concept in one column must be mapped to its counterpart in the other column.

Table 2 - Mapping Tropos to AORML

Tropos Concepts	**AORML Constructs**
actor	agent
goal	-
plan	path for interaction modeling (AOR Interaction Sequence Diagram)
capability	set path for interaction modeling (set of AOR Interaction Sequence Diagrams)
resource	object
dependency	AOR Agent Diagram association relation

Table 2 shows an actor in Tropos, modeling an entity that has strategic goals and intentions 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. On the other hand, Tropos' plans may indicate paths for AORML's interaction modeling, with the use of AOR Interaction Sequence Diagrams (ISDs). Capabilities in Tropos may be seen as a set o plans and, therefore, could be mapped for the set of interaction modeling paths, representing the agent's plans (i.e. a set of AOR ISDs). Analogously, Tropos resources representing physical or information entities of the domain become objects according to AORML conceptualization. Additionally, Tropos prescribes that goal dependencies between two actors indicate that one actor depends on the other in order to attain some goal, execute some plan, or deliver a resource. Such goal dependencies will lead to the establishment of some kind of association relations between these agents in an AOR Agent Diagram.

4. CASE STUDY

Here, we start presenting the model of our scenario, as described in section 2. Note that the analysis model for the same scenario has been previously published [3] and, here, we move forward from the system architectural design. We begin with a first architectural design model, still using Tropos (sub-section 4.1) and, then, we present a refined design model with the use of AORML (sub-sections 4.2).

4.1. The KARe System Architecture

The analysis of KARe's requirements [3] leads to the identification of a possible structure of the system-to-be actor in terms of system's roles (sub-actors), i.e. the global architecture is identified through delegation of main system's goals to internal sub-actors. For instance, the roles of Peer Assistant and User Model Engine may be designed in order to take care of goals respectively related to representing and searching knowledge on behalf of the CoP members (i.e. the KARe Users), and providing personalization and configurability, while a Broker role may be proposed to achieve goals related to matchmaking peers with similar interests as adequate knowledge sources for specific requests. The emerging structure is that of an agent organization (or more generally of a peer-to-peer system [1]), whose high level architecture may be modeled in terms of actor dependencies, according to *Tropos*, as in the example depicted in Figure 1. Note that, in this model, we use technology-oriented terminology, such as *question/ answer service* and *peer-to-peer infrastructure*.

In ARKnowD, *actor* and *goal* are the main elements of requirements analysis. In architectural design, however, we start placing *plans* and *resources*, considered here as design elements. Fig. 2 presents only one plan and three resources, mainly due to lack of space. However, they are enough to exemplify the transformation from Tropos to AORML. Note also that in this model, there is only one domain actor: the CoP Member, while all others are system actors.

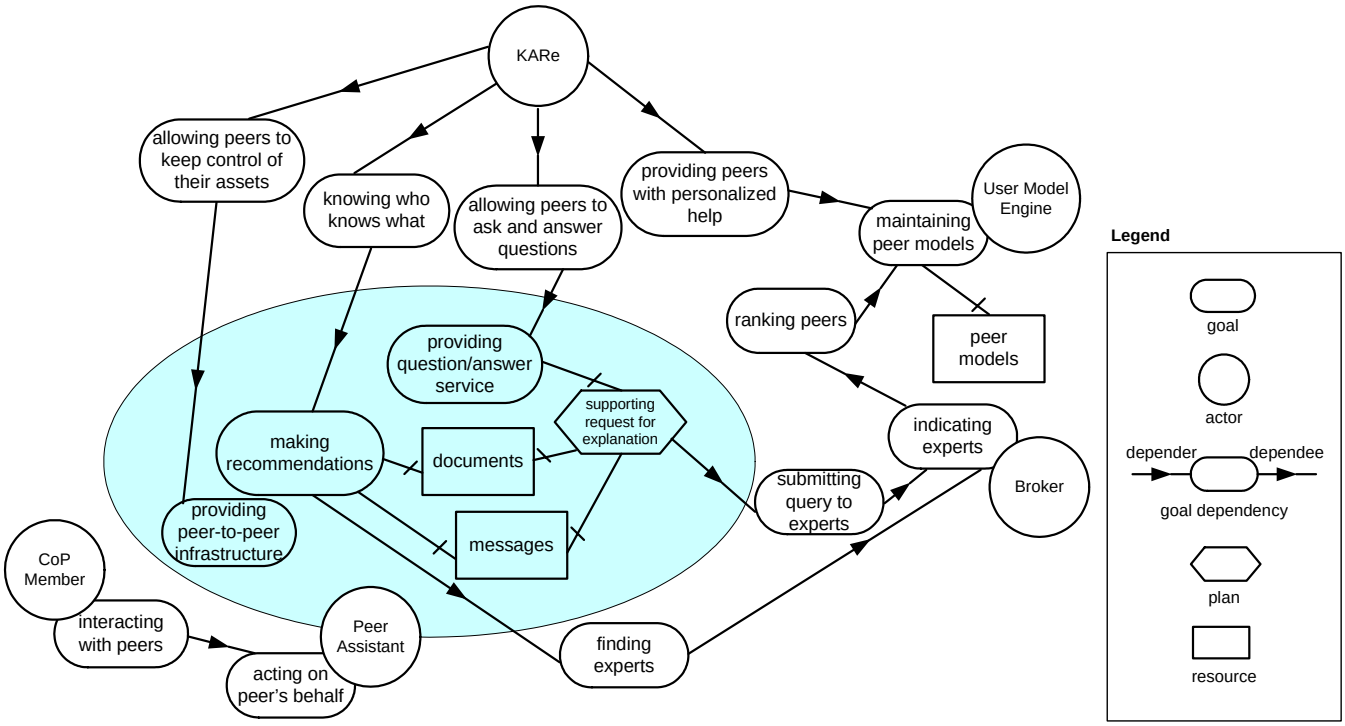

Figure 1 - Actor diagram showing the high level architecture of the KARe system

4.2. The Detailed Architecture: AOR Agent Diagram's Agents, Objects and Relations

Figure 2 presents KARe's AOR Agent Diagram (AD). This diagram includes all human, artificial and institutional agents (distinguished by UML stereotypes) involved in the domain, the domain objects and the relationships between all these entities. Note that this diagram is very similar to the UML class diagram, having however specific notation elements to represent agents and objects.

By referring to Fig.1, note that all *actors* in that model are depicted as *agents* in the AD of Fig 2. And also, all *resources* of the previous model are turned into *objects* here. Moreover, the agents and objects are linked by several types of relations, also derived from the Tropos model of Fig. 1 (*dependencies* there indicate *relations* here). In addition to the derived elements, we can see some extra ones, added by the designer. For instance, here, he defines an auxiliary agent named **Artifact Manager** to help the users organize their knowledge items. The **Community of Practice** institutional agent was also added to provide contextual information. In this case, the **CoP Member** is depicted within it, showing that it is part of the community. Another example of these additions is the specialization of the **Peer Model** in **Interaction** and **Personal Features**. Inserting new agents or objects is a common task in refining a design model. Besides these additions, there are also suppressions in the agent model of Fig. 2. For instance, according to our transformation rules, there should be a relation between **PA** and **Broker**, however Fig 2 does not show it. That is because, in this particular case, the designer thought such relation did not add much to the comprehension of the system. It is also the responsibility of the designer to add all cardinalities and relations' name accordingly.

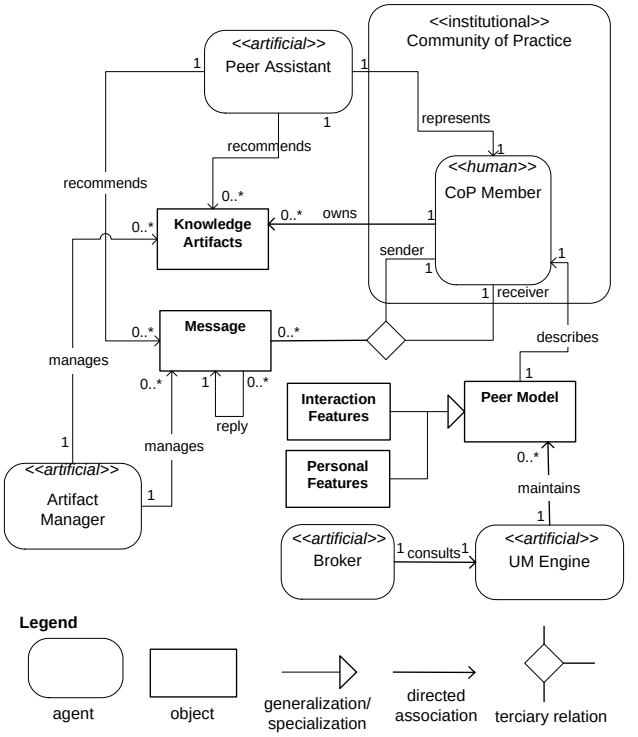

Figure 2 - Agent Diagram of the KARe System

Figure 3 shows a sketch of the AD that could be suggested by ARKnowD's CASE tool, applying the transformation rules of Table 2 on the Tropos model of Fig.1. By using these transformation rules, it is possible to provide semi-automated support for designers, proposing them initial models that they should then refine.

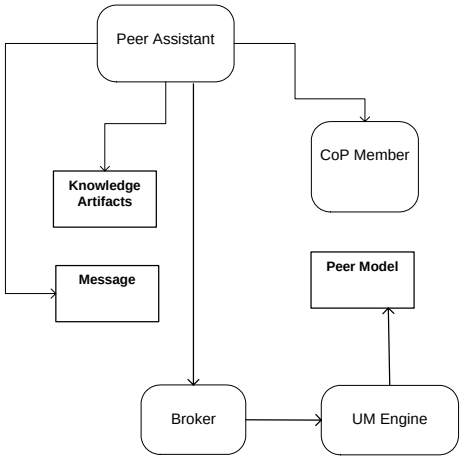

Figure 3 - A sketch of KARe's Agent Diagram

4.3. Interaction Modeling with AOR Interaction Sequence Diagrams

The *supporting request for explanation* plan depicted in Fig. 1 is an example of path for interaction modeling, according to Table 2. This means that the process of interactions among the agents in this model to comply with this plan should be detailed using AOR Interaction Sequence Diagrams (ISDs). These diagrams are similar to UML Interaction Diagrams, depicting a prototypical interaction sequence describing a specific process. However the AOR ISD provides more than just interaction links (which in UML generally mean method calls). In AORML, these interaction links are differentiated, representing agent's *messages*, *non-communicative actions*, *events*, or *claims/commitments*. To illustrate an ISD related to the KARe's system, we refer to [4], which presents the information and interaction model of a previously proposed system that inspired KARe's development.

5. CONCLUSIONS AND FUTURE WORK

Here, we propose ARKnowD, an agent-oriented modeling approach for KM scenarios, based on an organizational theoretical framework. Besides that, ARKnowD takes a distributed view on KM, supporting the proposal of bottom-up KM systems, based on the organizational goals as well as on the objectives of the main stakeholders (i.e. the knowledge holders).

ARKnowD relies on the integration of existing work on agent-oriented software engineering, exemplified with the integration of Tropos and AORML. This specific paper focus on the design of a recommender system to illustrate how a transformation between these two approaches can be achieved, allowing us to provide guidelines to designers using ARKnowD. Such guidelines may also be given in a semi-automated fashion, using a CASE tool. Future work includes the implementation of our transformation rules in an existing CASE tool [8].

Although the KM domain is here used to exemplify our approach, it is our belief that the proposed methodology may be generalized to solve other kinds of problems within organizational settings. To assess if this assumption is true, new case studies should be developed.

6. REFERENCES

[1] M. Bonifacio and P. Bouquet and P. Traverso, "Enabling Distributed Knowledge Management. Managerial and Technological Implications", Novatica and Informatik/Informatique,2002, 3,1.

[2] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., & Perini, A. (2004) Tropos: An Agent-Oriented Software Development Methodology. In JAAMAS – International Journal of Autonomous Agents and Multi Agent Systems 8(3):203–236, May 2004.

[3] Guizzardi, R. S. S., Perini, A., Dignum, V. Providing Knowledge Management Support to Communities of Practice through Agent-oriented Analysis. In Proceedings of the 4th International Conference on Knowledge Management, Graz, Austria, June/2004.

[4] Guizzardi, R. S. S., Aroyo, L., Wagner, G. (2003) Agent-oriented Knowledge Management in Learning Environments: A Peer-to-Peer Helpdesk Case Study. (eds) van Elst, L., Dignum, V., Abecker, A. "Agent-Mediated Knowledge Management" Heidelberg: Springer-Verlag. 2003.

[5] Nonaka, I., & Takeuchi, H. (1995). The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation. New York: Oxford University Press.

[6] Mellor, S. J., Scott, K., Uhl, A., and Weise, D. MDA Distilled (2004). Addison-Wesley Object Technology Series

[7] Orlikowski, W., and Gash, G. Technological Frames: Making Sense of Information Technology in Organizations, ACM Transactions on Information Systems, 1994, 12,2.

[8] Perini, A., and Susi, A. (2004) Developing tools for Agent-Oriented visual modeling. In Proceedings of the 2nd German Conference on Multiagent System Technologies, Erfurt, Germany, September 2004.

[9] Wagner, G. (2003) The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. *Information Systems*, 28:5.

[10] Wenger, E. (1998) Communities of Practice: learning, meaning and identity. New York: Cambridge University Press.