Modeling of CSCW system with Ontologies

Mario Anzures-García[1, 2], Luz A. Sánchez-Gálvez[1,2], Miguel J. Hornos[2], Patricia Paderewski-Rodríguez[2], and Antonio Cid[1]
[1] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, 14 sur y avenida San Claudio. Ciudad Universitaria, San Manuel, 72570 Puebla, Mexico
{anzures, luzsg}@correo.ugr.es
[2] Departamento de Lenguajes y Sistemas Informáticos, E.T.S.I. Informática y de Telecomunicación, Universidad de Granada, C/ Periodista Saucedo Aranda, s/n, 18071 Granada, Spain.
{mhornos, patricia}@ugr.es

Abstract. In recent years, there has been a growing interest in the development and use of domain ontologies, strongly motivated by the Semantic Web initiative. However, the application of ontologies in the CSCW domain has been scarce. Therefore in this paper, it presents a novel architectural model to CSCW systems described by means of an ontology. This ontology defines the fundamental organization of a CSCW system, represented in its concepts, relations, axioms and instances.

Keywords: Ontology, Groupware Application, SOA, Architectural Model, Services.

1 Introduction

In the last two decades, the enormous growth of Internet and the web have given rise to an intercreativity cyberspace, in which groups of people can communicate, collaborate and coordinate to carry out common tasks. Therefore, a great number of groupware applications has been developed using different approaches, including object-oriented, component-oriented, and agent-oriented ones. However, the development of this kind of applications is very complex, because different elements and aspects must be taken into account. Hence, these applications must be simultaneously supported by models, methodologies, architectures and platforms to be developed in keeping with current needs. In the groupware domain, one of the models most used is the Unified Modelling Language (UML) [1], although this has not any element to represent constrains, which are very important in applications so complex as the groupware ones.

There has recently been an increase in the use of ontologies in any domain to model applications. An ontology is presented as an organization resource and knowledge representation through an abstract model. This representation model provides a common vocabulary of a domain and defines the meaning of the terms and the relations amongst them. In the domain of groupware applications, the ontology

provides a well-defined common and shared vocabulary, which supplies a set of concepts, relations and axioms to describe this domain in a formal way.

In this paper, two ontologies for the groupware domain are proposed. The first ontology determines who authorize the registration of users, how interaction is carried out among them, and how the turns for users participation are defined, among other aspects. Moreover, it allows supporting modifications in runtime, such as changing the user role, the rights/obligations of a role, the current policy, etc. The second ontology establishes the necessary SOA-based services to develop groupware applications in accordance with the existing papers in the literature about the development of this type of applications. In addition, these services are clustered in modules and layers with respect to the concern that they represent.

This paper is organized as follows. Section 2 gives an brief introduction to the ontologies. Section 3 describes the ontology-based modeling of the group organizational structure. Section 4 presents an ontological model, which allows us to specify an architectural model for the development of groupware applications. Finally, Section 5 outlines some conclusions and future work.

2 Introduction to the Ontologies

There are several definitions of ontology, which have different connotations depending on the specific domain. In this paper, we will refer to Gruber's well-know definition [2], where an ontology is an explicit specification of a conceptualization. For Gruber, a conceptualization is an abstract and simplified view of the world that we wish to represent for some purpose, by the objects, concepts, and other entities that are presumed to exist in some area of interest, and the relationships that hold them. Furthermore, an explicit specification means that concepts and relations need to be couched by means of explicit names and definitions.

Jasper and Ushold [3] identify four main categories of ontology applications: 1) neutral authoring, 2) ontology-based specification, 3) common access to information, and 4) ontology-based search. In the work presented here, the main idea is to use ontologies to specify the modeling of both the group organizational structure and the architectural model in the groupware domain, since an ontology is a high level formal specification of a certain knowledge domain, which provides a simplified and well defined view of such domain.

Ontology is specified using the following components:

- *Classes*: There is a set of classes, which represent concepts that belong to the ontology. Each class may contain individuals (or instances), other classes or a combination of both, with their corresponding attributes.
- *Relations*: These define interactions between two or several classes (object properties) or between a concept and a data type (data type properties).
- *Axioms*: These are used to impose constraints on the values of classes or instances. Axioms represent expressions (logical statement) in the ontology and are always true inside the ontology.
- *Instances*: These represent the objects, elements or individuals of an ontology.

These four components will be described for the two ontologies proposed in this paper.

In addition, ontologies require of a logical and formal language to be expressed. In Artificial Intelligence, different languages have been developed, like the First-Order Logic-based (which provide powerful primitive for modeling), the Frames-based (with more expressive power but less inference capacity), and the Description Logics-based (which are more robust in the reasoning power) ones.

OWL (Web Ontology Language) [4] is a language based on Description Logics for defining and instantiating Web ontologies based on XML (eXtensible Markup Language) [5] and RDF (Resource Description Framework) [6]. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships among those terms. This language makes possible to infer new knowledge from a conceptualization, by using a specific software called *reasoner*. It has used the tool Protégé [7], which is based on OWL, to define the ontology for group organizational structure.

In the groupware domain, ontologies have mainly been used to model task analysis or sessions. Different concepts and terms, such as group, role, actor, task, etc. have been used for the design of task analysis and sessions. Many of these terms are considered in our conceptual model. Moreover, semiformal methods (e.g. UML class diagrams, use cases, activity graphs, transition graphs, etc.) and formal ones (such as algebraic expressions) have also been applied to model the sessions. There is also a work [8] for modeling cross-enterprise business processes from the perspective of cooperative system, which is a multi-level design scheme for the construction of cooperative system ontologies. This last work is focused on business processes, and it describes a general scheme for the construction of ontologies. However, in this paper, we propose to model two specific aspects: the group organizational structure and the architecture of a groupware application. Consequently, the application domain of both ontologies is groupware, not business processes.

3 Ontology for specifying an architectural model

In order to specify architectural model five concerns are identified: Data, Group, Cooperation, Application, and Adaptation. Consequently, five layers are considered. Four layers are composed by modules and services, while the fifth one, the *Data Layer*, contains repositories with the necessary information to carry out the group work. The services of the architectural model are defined by the concepts' ontology.

3.1. Ontology Concepts

The architecture components are characterized through the concepts' ontology (shown in Figure 1), which will be briefly described below:

- *Registration* is the first action that a user must carry out to can participate in the group work using the collaborative application.
- *Authentication* validates the access to the group and depends on the organizational style defined in the same.

- *Group* is who works in the session to perform work group.
- *Organizational_Style* defines the organizational style that a group will use to carry out the group work.
- *Stage* restricts user's access to the application in accordance with the organizational style defined in it.
- *Session* defines a shared workspace where a group carries out common tasks.
- *Session_Management* manages and controls one or more sessions.
- *Concurrency* manages shared resources to avoid inconsistencies by using them.
- *Shared_Resource* is used by users to carry out basic activities.
- *Basic_Activity* is an action that a user must perform to carry out a task (which can be made up by one or more basic activities).
- *Task* is carried out by the group to achieve a common goal.
- *Notification* notifies one or more users of all events that happen in a session.
- *Group_Awareness* gets the necessary information to supply group awareness to users that take part in a group.
- *Group_Memory* is supplied by the application to facilitate a common context.
- *Application* is used by the users to carry out group work in established session.
- *Configuration* configures the application the first time that it is used and when it is necessary.
- *User_Interface* shows users all the information about the application execution.
- *Environment* modifies the user interface to present the information in accordance with the device used by each user.
- *Adaptation* is a process that allows adapting the collaborative application to the new needs of the group.
- *Detection* monitors the execution environment to detect the events that determine the adaptation process.
- *Agreement* decides whether an adaptation process must be carried out or not.
- *Vote_Tool* is used by users to perform the agreement.
- *Adaptation_Flow* is a set of steps carried out to adapt the collaborative application in accordance with the selected event.
- *Repair* is required when the adaptation process can not be performed.

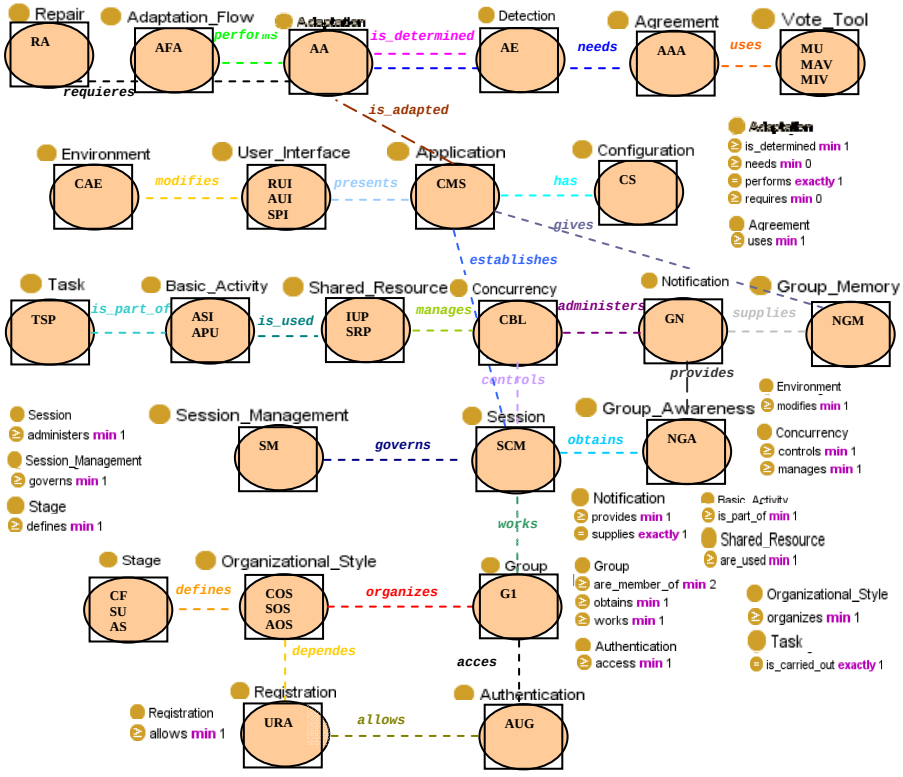

Figure 1. Ontology for specifying an architecture to model collaborative applications.

3.2. Ontology Relations

The architecture relationship to each component and its environment are symbolized with the ontology relations (see Figure 1) listed below:

- *allows (Registration, Authentication):* Only registered users are allowed to authenticate to access to the collaborative application.
- *access* (*Authentication, Group):* Authentication allows users to access to group.
- *depends* (*Registration, Organizational_Style):* Users registration depends on the organizational style defined at a given stage.
- *organizes (Organizational_Style, Group):* An organizational style specifies the way in which the group is organized.
- *defines (Stage, Organizational_Style):* A stage defines an organizational style.
- *works (Group, Session):* A group needs to be connected to a session to work.
- *governs (Session_Management, Session):* The session management governs a session.
- *controls (Concurrency, Session):* The concurrency service controls the existing interaction in a session.
- *manages (Concurrency, Shared_Resource):* The concurrency service manages the shared resources to guarantee mutually exclusive usage of these.
- *is_used (Shared_Resource, Basic_Activity):* The shared resources are used by basic activities.
- *is_part_of (Basic_Activity, Task):* A basic activity is part of a task.
- *administers (Session, Notification):* The session administers the notification.
- *provides (Notification, Group_Awareness):* The notification process provides group awareness.
- *obtains (Group, Group_Awareness):* A group obtains group awareness to avoid inconsistencies in the collaborative application.
- *supplies (Notification, Group_Memory):* The notification process supplies group memory.
- *gives (Application, Group_Memory):* The application gives group memory.
- *establishes (Application, Session):* An application establishes a session.
- *presents (Application, User_Interface):* An application presents an user interface so that users can use the collaborative application.

- *modifies (Environment, User_Interface):* The environment modifies the user interface according to the device used by each user.
- *has (Application, Configuration):* Each application has a configuration process, which is carried out by users.
- *is_adapted (Application, Adaptation):* An application is adapted by the adaptation process.
- *is_determined (Adaptation, Detection):* The adaptation process is determined by the detection process.
- *needs (Adaptation, Agreement):* The adaptation process needs an agreement process to decide whether the adaptation is carried out or not.
- *uses (Agreement, Vote_Tool):* The agreement process uses a vote tool to carry out the agreement.
- *performs (Adaptation, Adaptation_Flow):* The adaptation process performs an adaptation flow to appropriately adjust the application.
- *requires (Adaptation, Repair):* When the adaptation process can not be performed, it is required to repair the application to avoid inconsistencies in it.

3.3. Ontology Axioms

Finally, the principles governing design and evolution of the architectural model are represented by ontology axioms (see Figure 1):

- An authentication must have only one registration, i.e. an user is authenticated only if she/he is registered.
- A registration depends on an organizational style, i.e. an user is registered with accordance to organizational style established in the group work.
- An organizational style organizes at least one group.
- A group works at least in one session.
- An application establishes at least one session.
- A session administers at least one notification process.
- A group obtains group awareness.
- An application gives group memory.
- The concurrency service controls at least one session.
- The concurrency service manages at least one shared resource.
- A shared resource is used by at least one basic activity.
- A basic activity is part of at least one task.
- An application has at least one possible configuration.
- An application presents at least one user interface.
- An environment modifies at least one user interface.
- An application can be adapted by an adaptation process.
- An adaptation process is determined by at least one detection process.
- An agreement process is carried out only if there is an adaptation process in a non-hierarchical organizational style.
- An agreement process uses at least one vote tool.
- An adaptation process performs only one adaptation flow.
- An adaptation flow must verify at least one pre-condition and post-condition to carry out the adaptation.

- An adaptation process can require a repair process, if this has not finished.

3.4. Ontology Instances

In order to show the architectural functionality, this section presents a set of instances (see Figure 1), derived from the definition of the application instance, which is a Conference Management System (CMS). A CMS is a web-based application that supports the organization of scientific conferences. It can be regarded as a domain-specific content management system. Nowadays, similar systems are used by editors of scientific journals.

This type of systems generally has four stages: submission, assignment, review, and acceptance/rejection of papers, and this paper adds the stage of application configuration. CMS supports three user groups: Authors (A), Program Committee Members (PCM) and Program Committee Chairs (PCC). The first user group (A) corresponds to people who can submit papers (at the submission stage) through the Internet, and who receive the review results and the final decision via an email (at the acceptance/rejection stage). It is the largest user group (its average number is normally between 100 and 400 people for most conferences). The second user group (PCM) is made up of people who must evaluate some of the submitted papers and send the result to the PC Chairs via the Internet (at the review stage). Its number is about 20–50 persons in average. People in the last user group (PCC) are in charge of allocating papers to reviewers (at the assignment stage) and making the final decision on papers, as well as a number of other operations. This is the least numerous group, being usual 1–3 PCC per conference. Therefore:

- *Session* instance is Session of the Conference Management (SCM).
- *Session_Management* instance is Session Management (SM).
- *Stage* instances are configuration (CF), and submission (SU), assignment (AS), review (RE) and acceptance/rejection (AC) of papers.
- *Authentication* instance is user authentication in the group (UAG).
- *Registration* instance is user registration in the application (URA).
- *Users* instances are U1, U3 and U4 as A, U3 as PCM, and U2 as PCC.
- *User_Interface* (UI) instances are Registration UI (RUI), Authentication UI (AUI), Submitting Paper UI (SPI), Configuration UI (CUI), etc.
- *Environment* instance is collaborative application environment (AE).
- *Organizational_Style* (OS) instances are Configuration OS (COS), Submission OS (SOS), Assignment OS (AOS), Review OS (ROS), and Acceptance/rejection OS (POS). In the ontology shown in Figure 3, SOS is the unique OS considered by simplicity reasons.
- *Group* instance is G1, which is made up of three users, U1, U3 and U4, because U2 does not participate at SOS.
- *Concurrency* instance is locks mechanism (LM).
- *Shared_Resource* (SR) instances are paper (SRP), and uploading paper (IUP).
- *Basic Activity* (BA) instances are submitting information (ASI), and uploading paper (AUP).
- *Task* instance is submitting paper (TSP).
- *Notification* instance is group notification (GN).

- *Configuration* instance is configuration of the system (CS).
- *Adaptation* instance is adaptation application (AA).
- *Detection* instance is adaptation event (AE).
- *Agreement* instance is adaptation agreement of the application (AAA).
- *Vote Tool* instances are majority vote (MV), maximum value (MAV) and minimum value (MIV).
- *Adaptation_Flow* instance is adaptation flow of the application (AFA).
- *Repair* instance is reparation of the adaptation (RA).

4 BPM to Manage the Ontoloy-based Architectural Model

BPM [16] is a set of methods, tools and technologies used to design, perform, analyze and manage operational business processes, by means of different phases. It also facilitates service composition. In this paper, BPM is based on ontological approach [17] and is composes for three phases, which are: *Process Modeling*, *Process Implementation*, and *Process Execution*. The ontology is used in order to simplify the task of governing the behaviour of BPM; it enables to BPM to uses concepts to describe the models and the entities being controlled, thus simplifying their description and facilitating the analysis and the careful reasoning over them; and it allows dynamically calculating relations between business processes and environment, supporting modifications in runtime. BPM and SOA make the integration faster and easier than never, it is not necessary to discard of the investments already made, everything can be reused.

4.1. Process Modeling

This phase identifies the participating elements in a business process. Unlike other existent models, we use an ontology (shown in the Figure 1), to clearly specify the semantics of the tasks and the decisions in the process flow. Therefore, four layers of the architectural model, (see Figure 2), are composed of the services, which were defined as concepts, in the ontology. The *Group Layer* includes three modules, which are *Access, Group* and *Session*. The *Access Module* has two services: *Registration* and *Authentication*. The *Group Module* presents three services: *Group, Organizational Style,* and *Stage*. The *Session Module* contains two services: S*ession Management,* and *Session.* The *Cooperation Layer* has the *Context Module* and *Interaction Module*. The former includes four services: *Concurrency, Shared Resource, Activity Basic,* and *Task.* The later encompasses the *Group Awareness Service,* the *Group Memory Service,* and the *Notification Service.* The *Application Layer* comprises only the *Application Service,* the *Configuration Service,* the *User Interface Service,* and the *Environment Service.* Finally, the *Adaptation Layer* involves the *Pre-adaptation Module* and the *Adaptation Module.* The former encompass the *Detection Service,* the *Agreement Service,* and the *Vote Tool Service.* The latter comprises the *Adaptation Flow Service,* and the *Repair Service.*

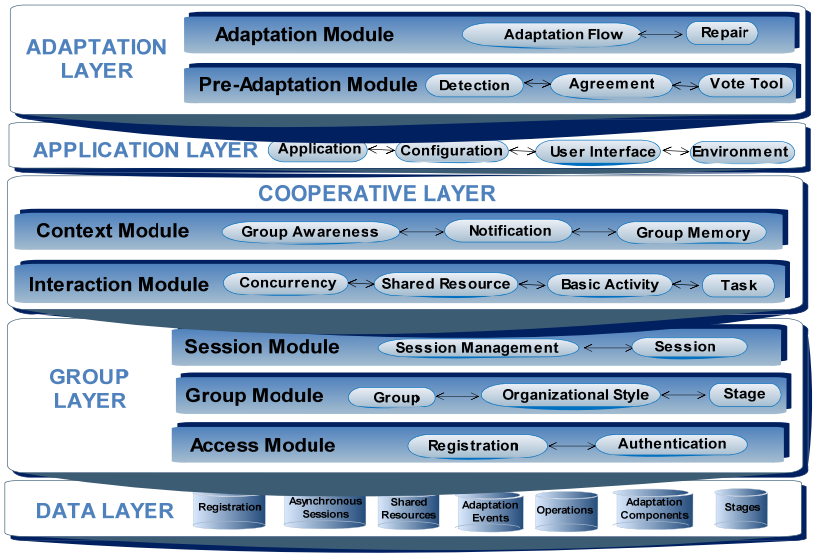

Figure 2. Architectural model for developing collaborative applications.

4.2. Process Execution

In this phase, the business process model is transformed into an executable process model, which can be deployed to a process engine for its execution. Figure 3 shows a sequence diagrams (that represents an executable process model), when the author submits papers to the CMS. In this figure, the CMS users are consumer services, invoking different services and only are considering some services of the architectural model by simplicity.

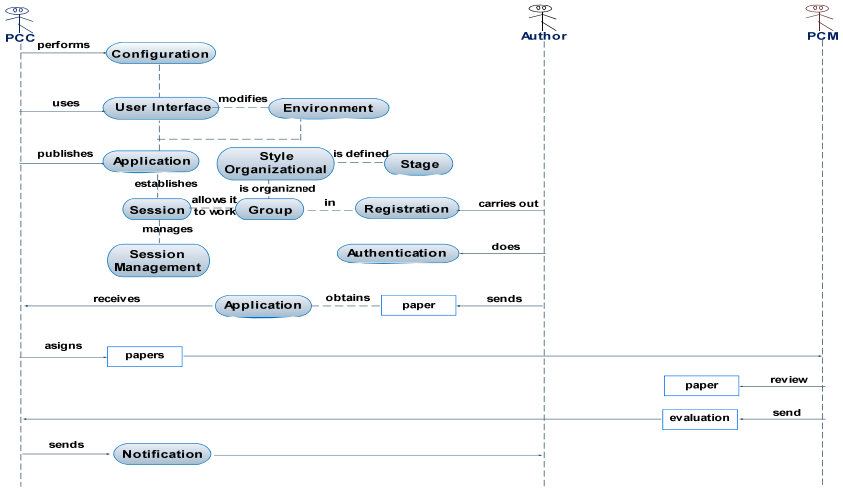

Figure 3. Sequence diagram of papers sent.

4.3. Process Implementation

In the process execution phase, the process engine executes a process model by firstly creating a process instance and then navigating through the control flow of the process model. In order to ensure seamless interaction when navigating through the control flow of the process model, this phase provides mechanisms for the discovery, selection and invocation of services. The module dynamically discovers and selects the appropriate service of the architectural model basing on the task description, and invokes it on behalf of the process engine, which plays the role of a *requester service* when it invokes the service to perform a task. Moreover, this module carries out a process monitoring that provides relevant information about running process instances. If during the process execution a failure arises, such as network faults, server crashes, or application-related errors (e.g. unavailability of a requested service, errors in the composition or missing data, etc.), reconfiguration actions are carried out, such as duplication (or replication) or substitution of a faulty service. The first case involves addition of services representing similar functionalities; this aims at improving load balancing between services in order to achieve a better adaptation. The second case encompasses redirection between two services; applying this action means the first one is deactivated and replaced by the second one.

5 Conclusions and Future Work

The current work has presented an ontology-based architectural model, which facilitates the development of collaborative applications. The ontology describes the components, their relationships to each other and the environment, and the principles governing architectural design and evolution. For that reason, we think that the ontology is a proper model to describe architectures. BPM is used to manage and control the interaction between the services that make up the architectural model. In addition, BPM also is based on the ontology proposed. These services are designed using SOA that together with BPM, facilitates the application's integration. The future work will consist on extending the existent reconfiguration actions of the service-based collaborative applications.

References

1. Garlan D., Shaw, M.: An introduction to software architecture. Advances in Software Engineering and Knowledge Engineering, 1--39, (1993)
2. Perry, D.E., Wolf, A.L.: Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes 17(4), 40--52, (1992)
3. Architecture working group: Recommended practice for architectural description of software-intensive systems. IEEE Std 1471 (2000)
4. UML 2.0 Superstructure Specification (OMG). Ptc/03-08-02, 455—510, (2003)
5. Spivey, J.M.: The Z Notation: A Reference Manua., Prentice Hall, (1989)
6. Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press, (1996)

7. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. I. J. Human Computer Studies 43-(5/6), 907--928 (1995)
8. Gómez-Pérez, A., Fernández-López, M, Corcho, O.: Ontological Engineering with Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web, Springer, (2004)
9. Uschold, M., Grüninger, M.: Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11(2), 93–155 (1996)
10. Farquhar, A., Fikes, R, Rice, J.: The Ontolingua Server: A Tool for Collaborative Ontology Construction. I. J. Human Computer Studies 46(6), 707–727, (1997)
11. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Working Draft. http://www.w3.org/TR/owl-ref/ (2003)
12. Protegé: http://protege.stanford.edu/
13. Noguera, M., Hurtado, V, Garrido, J.L.: An Ontology-Based Scheme Enabling the Modeling of Cooperation in Business Processes. In; Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops 2006. LNCS vol. 4277, pp. 863—872. Springer, Heidelberg (2006)
14. Erl, T.: SOA: Concepts, Technology and Design. Prentice-Hall, (2005)
15. Howard, S., Fingar, P.: Business Process Management: The Third Wave, Meghan-Kiffer, (2003)
16. May, M.: Business Process Management: Integration in a Web-Enabled Environment, Prentice Hall, (2003)
17. Hepp, M., Roman, D.: An Ontology Framework for Semantic Business Process Management, In: 8th International Conference on Wirtschafts Informatik, Vol. 1 pp. 42--440, (2007)