Methontology-based Ontology Representing a Service-based Architectural Model for Collaborative Applications

Mario Anzures-García[1], Luz A. Sánchez-Gálvez[2], Miguel J. Hornos[2], and Patricia Paderewski[2]

[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
[2] Universidad de Granada, C/ Periodista Saucedo Aranda, s/n, 18071 Granada, Spain
{mario.anzures, sanchez.galvez}@correo.buap.mx,
{mhornos, patricia}@ugr.es

Abstract. Nowadays, the usage of ontologies to model systems has been extended to several domains, since ontology facilitates the modeling of complex systems and presents axioms which can be used as rules, policies or constrains to govern the system behavior. This paper presents ontology to represent a Service-based Architectural Model for Collaborative Applications, such as a Conference Management System (CMS), a simple social network, a chat, or a shared workspace. The development process of this ontology is based on *Methontology*, which is a well-structured methodology to build ontologies from scratch that includes a set of activities, techniques to carry out each one, and deliverables to be produced after the execution of such activities using its attached techniques. In order to show the development of the ontology using *Methontology*, the concepts, relations, axioms and instances of this ontology are specified for the collaborative application chosen as a case study, which is a CMS.

Keywords: Ontology, ontology construction process, methontology, collaborative application, architectural model.

1 Introduction

Recently, there has been an increase in the use of ontologies to model applications in any domain. 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 a collaborative domain, 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 domain, the ontologies have mainly been used to model tasks or sessions. Different concepts and terms, such as group, role, actor, task, etc., have been used for the design of tasks and sessions. 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 [1] for modeling cross-enterprise business processes from the perspective of a 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. In the literature, there are several papers on ontologies to model collaborative work; however, ontologies to support architectural modeling for developing collaborative applications have not been presented. In this work, the ontology will be used to facilitate the service composition of the architectural modeling by means of a Business Process Modeling (BPM), which is based on the ontological approach. This is the main reason to build the ontology; however, the BPM explanation is outside the scope of this article. Instead, this paper will focus on the ontology construction process using Methontology, since this methodology presents tasks set, which allow us to simplify this construction process and same time the development of collaborative application. These tasks detail each concept, the relations between these, as well as the rules and the axioms that govern its interaction, so it carries out the correspondent tables to these activities facilitate the development of the collaborative applications.

The ontology construction process must be based on methodologies, which specify techniques and methods that drive the development of the corresponding ontology. This allows us to define common and structured guidelines that establish a set of principles, design criteria and phases for building the ontology. *Methontology* [2, 3] is a methodology that supports the ontology construction process from scratch or the reuse of existing ontologies. For this reason, this paper proposes a Methontology-based ontology to represent a Service-based Architectural Model for Collaborative Applications (SAMCA). As a collaborative application example, a Conference Management System (CMS) is modeled by means of this ontology, to show how the CMS construction process is simple when it elaborates the table of each activity of the Methontology.

In the following sections, this paper presents a brief introduction to ontologies (Section 2), a review of the methodologies used in the ontology construction process, and particularly of *Methontology*, which is the one we applied to this work (Section 3), a concise explanation of SAMCA (Section 4), an example taken from the collaborative application CMS to show the use of *Methontology* (Section 5), and finally, the conclusions and future work (Section 6).

2 Introduction to Ontologies

There are several definitions of ontology, which have different connotations depending on the specific application domain. In this paper, we will refer to Gruber's well-known definition [4], where an ontology is an explicit specification of a conceptualization. *Conceptualization* refers to an abstract model of some phenomenon in the world by

identifying the relevant concepts of that phenomenon. *Explicit specification* means that the type of concepts used, and the constraints on their use are explicitly defined.

The ontologies require of a logical and formal language to be expressed. In Artificial Intelligence, different languages have been developed, including First-order Logic-based, Frames-based, and Description Logic-based ones. The first ones provide powerful modeling primitives; the second ones have more expressive power but less inference capacity, while the third ones are more robust in the reasoning power.

Ontology is specified using the following components [4]:

- *Classes*: Set of classes (or concepts) that belong to the ontology. They may contain individuals (or instances), other classes, or a combination of both with their corresponding attributes.
- *Relations*: These define interrelations between two or more classes (object properties) or a concept related to a data type (data type properties).
- *Axioms*: These are used to impose constraints on the values of classes or instances. Axioms represent expressions in the ontology (logical statement) and are always true if they are used inside the ontology.
- *Instances*: These represent the objects, elements or individuals of ontology.

Based on these components used to represent domain knowledge, the ontology community distinguishes two types of ontologies: lightweight ontologies and heavyweight ontologies [5]. On the one hand, *lightweight ontologies* include concepts, concept taxonomies, relationships between concepts, and properties that describe concepts. On the other hand, *heavyweight ontologies* add axioms and constraints to lightweight ontologies. Axioms and constraints clarify the intended meaning of the terms gathered on the ontology. Heavyweight and lightweight ontologies can be modeled with different knowledge modeling techniques and they can be implemented in various kinds of languages [6]. Ontologies can be highly informal, if they are expressed in natural language; semi-informal, if they are expressed in a restricted and structured form of natural language; formal, if they are expressed in an artificial and formally defined language (i.e. Ontolingua [7], OWL -Web Ontology Language- [8]); and rigorously formal, if they meticulously provide terms defined with formal semantics, theorems and proofs of properties, such as soundness and completeness.

The ontology that models a Service-based Architectural Model for developing Collaborative Applications is a heavy and formal ontology, which has been built using Methontology, defined in OWL, which is a description logic-based language that can define and instantiate Web ontologies using XML (eXtensible Markup Language) and RDF (Resource Description Framework), and published, implemented, validated and documented with the Protégé tool [9]. The ontology validation has been done with the software RACER (Renamed Abox and Concept Expression Reasoner) [10], which can work with Protégé. RACER can verify the consistency of the ontology (i.e. non-contradictory knowledge in it) to infer taxonomy and classify knowledge. Finally, the ontology was implemented with OWL, which allows generating a standalone application in Java.

3 Brief Review of Methodologies for Ontology Construction Process: Why We Use Methontology

The use of terms such as methodology, method, technique, process, activity, etc. on the ontological field is made in accordance to IEEE definitions [11], where it is established that a methodology is made up of methods and techniques, in such a way that the methods include processes and are detailed with techniques, and the processes contain activities. Finally, the activities are made up of a series of tasks.

The absence of common and structured guidelines has slowed the development of ontologies within and between teams. This has led to each development team usually follows their own set of principles, design criteria and phases for manually building the ontology. However, in the last decades, a series of methods and methodologies have been applied to the ontology construction process, such as: some general steps and some interesting points about the Cyc development [12]; the first guidelines for developing ontologies were proposed in the domain of enterprise modeling [5, 13]; a method to build an ontology in the domain of electrical networks was presented in [14]; the Methontology methodology appeared in [2, 3, 5]; a new method was proposed for building ontologies based on the Sensus ontology [15]; and the On-To-Knowledge methodology appeared in [16]. From all of them, only some methodologies were created for the ontology construction process from scratch in any domain, which are: Methontology, Sensus and On-To-Knowledge. In this paper, we use Methontology, since it is a well-structured methodology used to build ontologies from scratch as well as the reuse of existing ontologies. Moreover, this methodology defines different activities related to the ontology development process and its lifecycle, allowing us to adapt these activities to the knowledge representation needs in a domain. In this way, it is possible to modify the knowledge representation in each activity by adding, removing or changing some of the concepts, relations and instances previously presented. One more reason that has influenced the choice of this methodology is that it organizes and converts an informally perceived view of a domain into a semi-formal specification using a set of intermediate representations based on tabular and graphical notations that can be understood by domain experts and ontology developers. All this provides the necessary flexibility and simplicity to the ontology construction process.

Methontology, which was developed within the Ontology group at the Technical University of Madrid, enables the construction of ontologies at the knowledge level, having its roots in the main activities identified by the software development process [17] and in knowledge engineering methodologies [16]. This methodology includes: the identification of the ontology development process, a life cycle based on evolving prototypes (because it allows adding, changing, and removing terms in each new version) and techniques to carry out each activity in the three categories of activities identified in the ontology development process, which are [5]: ontology management, ontology development and ontology support. This process refers to which activities are performed

when building ontologies. Methontology includes a set of tasks (described in Section 5) for structuring knowledge within the conceptualization activity [5].

4 SAMCA

Our architectural model, namely SAMCA, is focused on solving a major problem in the development of collaborative applications, which is their lack to adapt to the different needs and collaborative scenarios that usually arise while these applications are running, i.e. these applications have not enough capability to be reused and adapted to different and dynamic collaborative scenarios. Therefore, a change in the group work objectives, the participants involved, the group structure, etc. can make a previously successful collaborative application unsuitable for the new situation. SAMCA allows us to develop collaborative applications that are both adaptable (the adaptation process adjusts the application functionality by the user's direct intervention) and adaptive (the adaptation process is automatically carried out). In order to achieve both kinds of adaptations, SAMCA considers the adaptation processes are focused on adjusting the architectural model and/or the group organizational structure. In the former case, the fact that the architectural model is based on SOA (Service-Oriented Architecture) [18], allows us to adapt it by replacing or even only modifying one or several application services in order to change solely that part of the application that did not fit the characteristics of the new scenario. In the latter case, such structure is modeled using other ontology (presented in [19]), which supports and manages the necessary dynamic changes and the different working styles of several groups to carry out the group work adequately.

SAMCA have a layered architectural style (see Fig. 1), containing the *Data, Group, Cooperation, Application* and *Adaption* layers. Each of them abstracts the corresponding concerns related to collaborative applications and allows a top-down development, in such a way that the lower layer provides resources to the upper layer, which uses them in accordance with its particular needs.

Data Layer contains different repositories with information, which will be used by the different services of the rest of SAMCA layers. *Application Layer* has only services; while the other layers are made up of modules, which contain services. *Group Layer* supplies a series of modules and services which support the necessary activities for carrying out the group work in a collaborative application. *Cooperation Layer* supports mechanisms that allow users to establish a common context and to coordinate group interactions. *Application Layer* contains the collaborative applications (e.g. CMS, chat, e-mail, etc.) which will be used to carry out the group work. *Adaption Layer* controls how the SAMCA services will be adapted when a state change requiring the modification of the collaborative application takes place, so that the application functionality can be preserved.

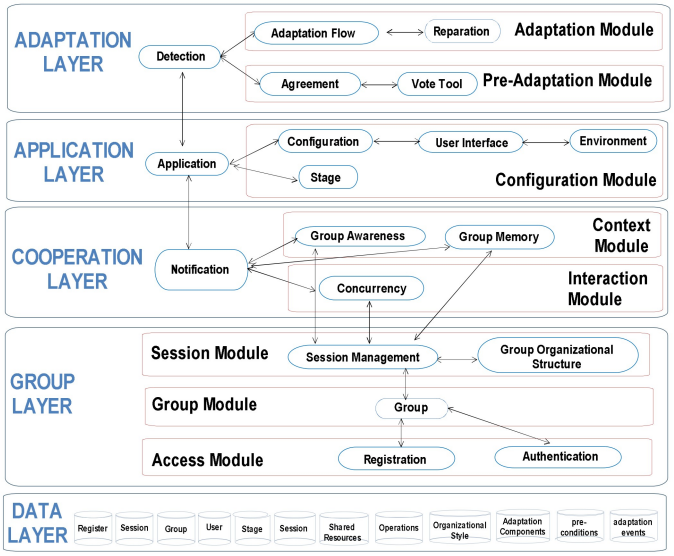

Fig. 1. General scheme of SAMCA

5 Methontology-based Ontology

This section shows how to apply Methontology to build an ontology-based CMS in according to the eleven tasks of this methodology [5]. It presents the tasks and how they are used to build a CMS.

Task 1: *To build the glossary of terms:* It identifies the concepts and relations to be included in the ontology, as well as the concept instances. Therefore, Table 1 presents the concepts (which are SAMCA services and are represented by an oval in Fig. 2), theirs relations (drawn by arrows joining the ovals in Fig. 2) and theirs instances (for *Group_ Organizational_Style* service, only the corresponding instances to the paper submission stage of the CMS and to the author role are shown, by reason of space limitation).

Task 2: *To build concept taxonomies:* It builds the concepts that make up the ontology taxonomy to develop the CMS, which is shown in Fig. 2. Note that this hierarchy has its root in the *Application* concept.

Task 3: *To build ad hoc binary relation diagrams:* It establishes relationships between concepts of different ontologies. Fig. 2 shows how two ontologies (represented with solid and dashed lines respectively) are related.

Task 4: *To build the concept dictionary:* It specifies the relations that describe each concept of the taxonomy in a concept dictionary (see Table 1).

Task 5: *To define ad hoc binary relations in detail:* Each binary relation is specified in detail, expressing relation name, source and target concept, and inverse relation, since all ontology relations are symmetrical, for example, the ***makes up*** relationship that as can see in the first row of the Table 1, which describes this relation and specifies that "it application ***is made up*** of Stages" and in the third row it specifies that "Stage ***makes up*** an Application".

Task 6: *To define instance attributes in detail:* This task describes in detail all the instances included in each concept (see third column of the Table 1).

Task 7: *To define class attributes in detail:* This task describes in detail all the class attributes included in the concept dictionary.

Task 8: *To define constants in detail:* This task describes in detail each of the constants defined in the glossary of terms.

Task 9: *To define formal axioms:* This task identifies the formal axioms needed in the ontology and describes them precisely (see Table 2). Methontology proposes to specify: name, natural language description, logical expression that formally describes the axiom using first order logic, concepts, attributes and ad hoc relations to which the axiom refers and variables used. For example, the first expression in Table 2 shows a formal axiom of our ontology that states that "every application sets up a configuration". The variables used are ?X for *Application* and ?Y for *Configuration*.

Task 10: To *define rules:* This task identifies which rules are needed in the ontology, and describes them.

Task 11: *To define instances:* This task defines relevant instances that appear in the concept dictionary.

The definition of these tasks allows us to establish the interactions of the SAMCA services and deduce the phases that make up this interaction. The first phase is *Services Preparation* (see Fig. 3), in which the responsible user to develop the collaborative application must define it and configure it. In this case, the responsible user is the PCC (see Table 1) and the collaborative application to build is a CMS. The services that must be configured are: Application, Stage, GOS, and User Interface. This configuration is carried out in accordance with data shown in Table 1 for the corresponding concepts (i.e. services).

The second phase corresponds to *Service Binding* (see Fig. 3), where the main services are binding to allows the user to use the CMS, e.g. when PCC invokes the CMS. Therefore, when a user invokes the *Application Service* (the CMS, in this case) to organize and/or participate in a conference, its user interface is displayed (using the *User Interface Service*). On the one hand, the CMS is established by means of a session (which defines a set of geographically distributed individuals who share the interest to achieve a common aim), which is established with the *Session Management Service*.

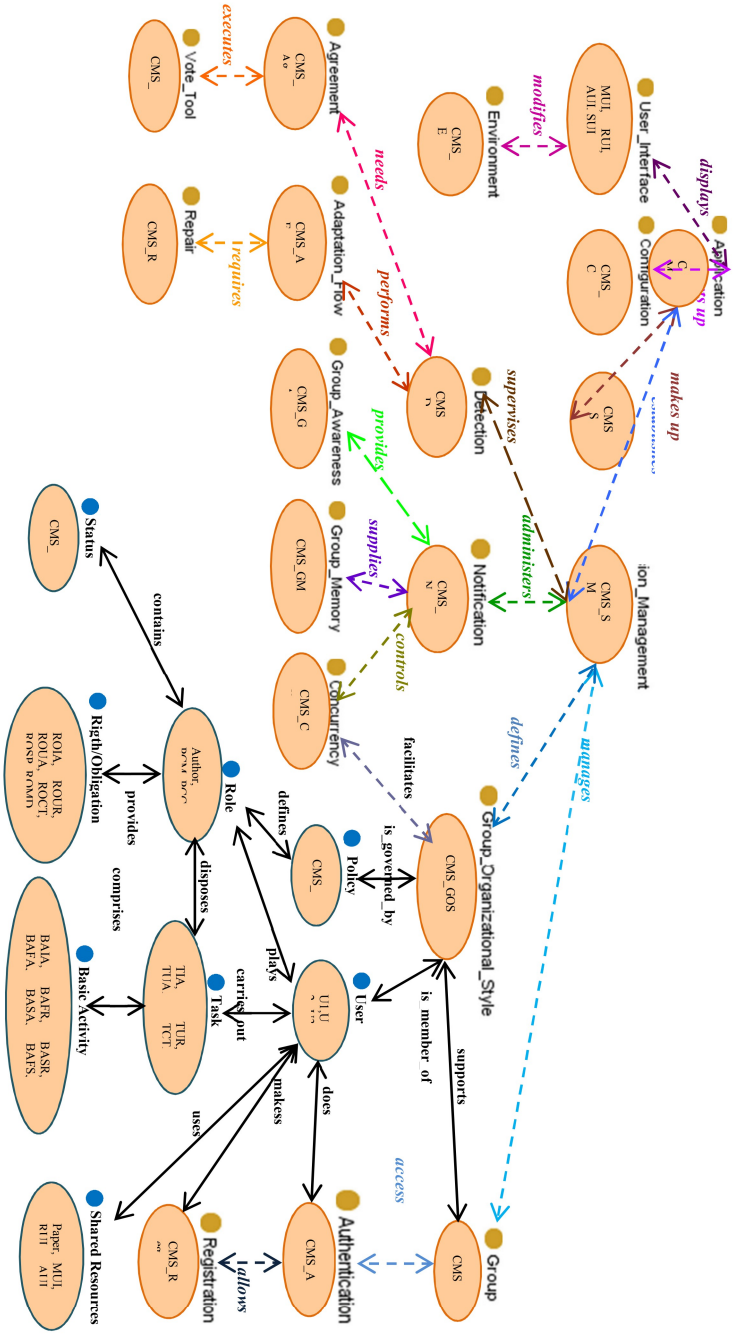

Fig. 2. Ontology developed with Methontology.

Table 1. An excerpt of the *Concept Dictionary* of an ontology to represent SAMCA.

Concept Name	Relations	Instances
Application	It ***sets up*** a *Configuration*; it is ***made up*** of *Stages*; it ***displays*** *User Interface*; it ***establishes*** *Session Management*	Conference Management System (CMS)
Configuration (C)	It is ***set up*** by *Application*	CMS_C
User Interface (UI)	It ***is displayed*** at *Application*; it ***is modified*** by the *Environment*	Main User Interface (MUI), Registration User Interface (RUI), Authentication User Interface (AUI), Submission User Interface (SUI)
Stage (S)	It ***makes up*** an *Application*	CMS_S
Session Management (SM)	It ***is established*** by *Application*; it ***is supervised by*** *Detection*; it ***administers*** *Notification*; it ***manages*** a *Group; it* ***defines*** a *Group Organizational Style (GOS)*	CMS_SM
Detection (D)	It ***supervises*** a *SM*; it ***needs*** *Agreement*; it ***performs*** *Adaptation Flow*	CMS_D
Agreement (Ag)	It ***is needed by*** *SM; it* ***executes*** *Vote Tool*	CMS_Ag
Adaptation Flow (AF)	It ***performs by*** *SM*; it ***requires*** *Reparation*	CMS_AF
Reparation (Rep)	*It* ***is requires by*** *Adaptation Flow*	CMS_Rep
Notification (N)	It ***is administered by*** *SM*; it ***provides*** *Group Awareness;* it ***supplies*** *Group Memory*; it ***controls*** *Concurrency*	CMS_N
Group Awareness (GA)	It ***is provided by*** *Notification*	CMS_GA
Group Memory (GM)	It ***is supplied by*** *Notification*	CMS_GM
Concurrency (Cy)	It ***is controlled by*** Notification; it ***is facilitated by*** *GOS*	CMS_Cy
Group	It ***is managed by*** a *SM*; it ***is supported by*** *GOS*; it ***is accessed by*** *Authentication*	CMG_G
Group_Organizational_Style (GOS)	It ***is defined by*** *SM*; it ***supports*** *a Group*; it ***is governed by*** a *Policy*; it ***has*** *Users;* it ***facilitates*** *Concurrency*	CMG_GOS
Authentication	It ***gives access to*** *Group;* it ***is allowed by*** *Registration*; it ***is done by*** *User*	CMS_Aut
Registration	it ***allows*** *Registration,* it ***is made by*** *User*	CMS_Reg
Policy	It ***governs*** a *GOS*; it ***defines*** *Roles*	CMS_P
User (U)	It ***is member of*** a *GOS*; it ***plays*** *Roles*; it ***does*** *Authentication;* it ***makes*** *Registration;* it ***carries out*** *Tasks*; it ***uses*** *Shared Resources*	U1 U2, U3, U4
Shared Resources	It ***is used by*** *User*	Paper, MUI, RUI, AUI, SUI
Role	It ***is played by*** *Users*; it ***is defined by*** *a Policy*; it ***contains*** a *Status*; it ***provides*** *Rights/Obligations*; it ***disposes*** *Tasks*	Author; Program Committee Members (PCM) and Program Committee Chairs (PCC)
Status (St)	It ***is contained in*** *Role*	Authors_St (U1, U2, U4), PCM_St (U2, U3), PCC_St (U1, U3)
Right/Obligation (RO)	It ***is provided by*** *Role*	RO Invoking Application (ROIA), RO User Registration (ROUR), RO User Authentication (ROUA), RO Choosing conference Topics (ROCT), RO Submiting Paper (ROSP), RO Modifying Data (ROMD)
Task (T)	It ***is carried out by*** *User*;it it ***is disposed by*** *Role;* it is comprised by *Basic Activities*	T Invoking Application (TIA), T User Registration (TUR), T User Authentication (TUA), T choosing Conference Topics (TCT), T Submission Papers (TSP), T Modifying Data (TMD)
Basic Activity	It ***forms part of*** *a Task*	BA Invoking Application (BAIA), BA Filling Registration form (BAFR), BA Sending Registration form (BASR), BA Filling Authentication form (BAFA), BA Sending Authentication form (BASA), BA Filling Submission form (BAFS), BA Sending Submission form (BASS)

Table 2. An excerpt of the *formal axioms* of an ontology to represent SAMCA.

Axiom Name	Description	Expression	Concepts	Binary Relations	Variables
Application Configuration	Every application sets up a configuration	For all(?X, ?Y) ([Application](?X) and ([Configuration](?Y) and [Application](?X) → [sets up Configuration](?Y))	Application Configuration	sets up	?X, ?Y
Establishing of Session	Every application establishes at least one management session	For all(?X, ?Y) ([Application](?X) and ([Management Session](?Y) and [Application](?X) → [establishes Management Session](?Y))	Application Management Session	establishes	?X, ?Y
Administering Notification	Every application administers notifications	For all(?X, ?Y) ([Application](?X) and ([Notification](?Y) and [Application](?X) → [administers Notification](?Y))	Application Notification	administers	?X, ?Y
Making up Application	Every application makes up of stages	For all(?X, ?Y) ([Application](?X) and ([Stage](?Y) and [Application](?X) → [makes up Stage](?Y))	Application Stage	makes up	?X, ?Y
Supervising Detection	Every application supervises the detection mechanism	For all(?X, ?Y) ([Application](?X) and ([Detection](?Y) and [Application](?X) → [supervises Detection](?Y))	Application Detection	supervises	?X, ?Y
Defining Group Organizational Style (GOS)	Every application defines Group Organizational Style	For all(?X, ?Y) ([Application](?X) and ([GOS](?Y) and [Application](?X) → [defines GOS](?Y))	Application GOS	defines	?X, ?Y

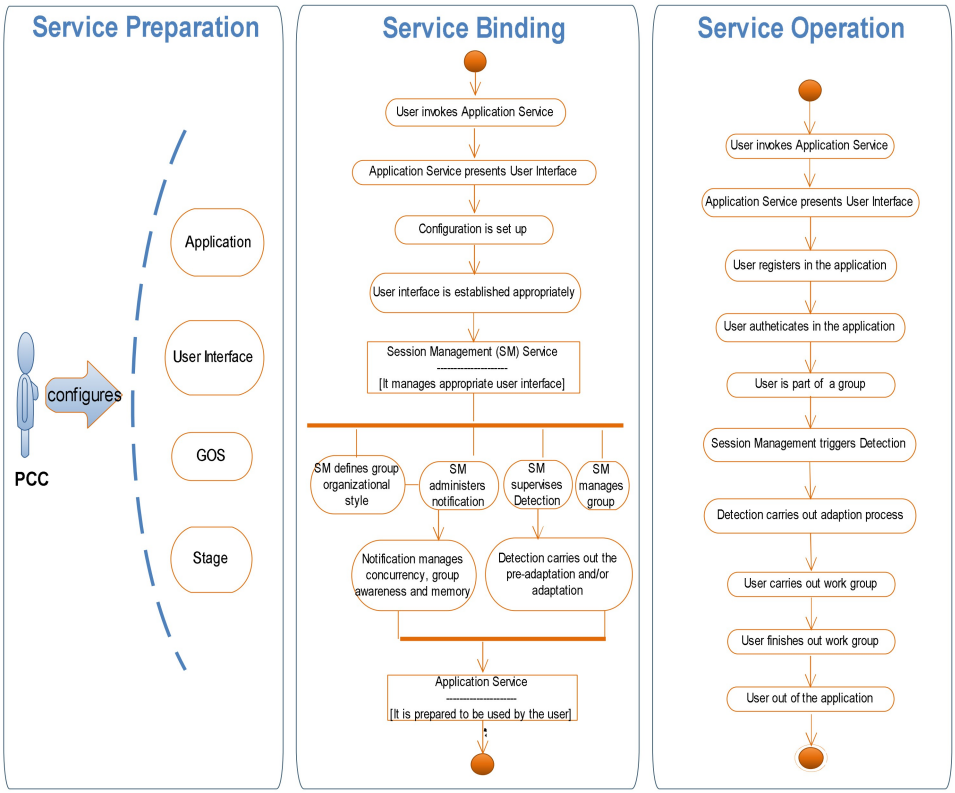

Fig. 3. Interaction phases into SAMCA.

On the other hand, *User Interface Service* is related to tasks to be carried out by a user playing a role according to the defined organizational style (with the *Group Organizational Style Service*). A session represents an execution environment, where the *Notification Service* and the *Detection Service* are always active. The former notifies each event which happens in this environment in order to provide group awareness (*Group Awareness Service*) and memory (*Group Memory Service*), as well as control the concurrency (*Concurrency Service*). The latter is triggered each time that a shared event has occurred, so that it can capture when an adaptation process must be carried out (this is defined by the group). When this process is performed, the *Adaptation Flow Service* is invoked. In case adaptation pre-conditions or pos-conditions are not met, the *Reparation Service* is required. If during the *Services Binding* phase a failure arises, such as unavailability of a requested service, errors in the composition of the collaborative application, missing data or parameters in an execution flow, etc., reconfiguration actions

are carried out, such as duplication (or replication) of services, 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.

The third phase is related with *Services Operation* (see Fig. 3). For example, when an Author wants to send a paper to a conference, s/he has to invoke the CMS, then the registration or authentication user interface (RUI or AUI, see Table 1) is displayed. Supposing that author U4 uses the CMS by first time, s/he must register in it. Once done this, the services *Group* (to add a new user), *Group_Organizational_Style* to assign the Author role, its tasks, its status, etc. – see Fig. 2), *Concurrency* (to manage the new user's permissions), *Group Awareness* (to inform existing users that a new user is in the CMS), *Group Memory* (to the shared resources modified by U4 to the other users or services), *User Interface* (to present the MUI – see Table 1) and *Detection* (to carry out the adaptation process corresponding to having a new user) are notified (*Notification Service*). If these services meet the adaptation pre-conditions and post-conditions, the adaptive process will successfully finish. This is an adaptive process, because it is automatically carried out by the CMS. Methontology facilitates the ontology construction process, since it supplies a set of activities determining its elements (with concepts), the interactions between them (by relations, axioms and rules) and the instances represented. In addition, if different set of instances are specified, different collaborative applications are got. Although a comparison between ontologies and architecture would be interesting, it is outside the scope of this paper.

6 Conclusions and Future Work

This paper has presented an ontology that allows carrying out the development of collaborative applications; a CMS has been used as an example. The ontology construction process is based on Methontology, which allows specifying ontologies from scratch as well as reusing some existent ones. The resultant ontology allows us to specify the services and the relations between them, facilitating the construction of our architectural model in a flexible way. Hence, it allows us to deduce the interactions between the different SAMCA services.

Our future work is orientated to specify a Business Process Management (BPM) based on the ontology proposed in this paper. Its main aim is to control the service composition of SAMCA, facilitating their adaptation in runtime.

References

1. 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. et al. (eds.) OTM Workshops 2006. LNCS, vol. 4277, pp. 863--872. Springer, Berlin (2006)
2. Fernández-López, M, Gómez-Pérez, A, Pazos, A, Pazos, J.: Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems & their Applications. 4(1), 37--46 (1999)
3. Fernández-López, M, Gómez-Pérez, A, Juristo, N.: Methontology: From Ontological Art Towards Ontological Engineering. In: Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33--40 (1997)
4. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: Guarino, N., Poli, R. (eds.) IWFOCAKR. Padova, Italy. Kluwer Academic Publishers, Deventer (1993)
5. 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)
6. Uschold, M., Grüninger, M.: Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11(2), pp. 93--155 (1996)
7. Farquhar, A., Fikes, R, Rice, J.: The Ontolingua Server: A Tool for Collaborative Ontology Construction. I. J. Human Computer Studies 46(6), pp. 707--727 (1997)
8. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Working Draft, http://www.w3.org/TR/owl-ref/ (2003)
9. Protégé Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu
10. RACER, http://www.sts.tu-harburg.de/~r.f.moeller/racer/
11. IEEE Standard Glossary of Software Engineering Terminology: IEEE Std 610-12. IEEE Computer Society, New York (1990)
12. Lenat, D.B., Guha, R.V.: Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project. Addison-Wesley, Boston, Massachusetts (1990)
13. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: Skuce, D. (ed.) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp. 6.1--6.10 (1995)
14. Bernaras, A., Laresgoiti, I., Corera. J.: Building and Reusing Ontologies for Electrical Network Applications. In: Wahlster W (ed.) European Conference on Artificial Intelligence. John Wiley and Sons, Chichester, United Kingdom, pp. 298--302 (1996)
15. Swartout, B., Ramesh, P., Knight, K., Russ. T.: Toward Distributed Use of Large-Scale Ontologies. In: Farquhar, A., Gruninger, M., Gómez-Pérez, A., Uschold, M., van der Vet, P. (eds.) AAAI'97. Stanford University, California, pp. 138--148 (1997)
16. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1) pp. 26--34 (2001)
17. IEEE Standard for Developing Software Life Cycle Processes: IEEE Std 1074. IEEE Computer Society, New York (1995)
18. Erl, T.: Service Oriented Architecture (SOA): Concepts, Technology and Design. Prentice-Hall, Englewood Cliffs (2005)
19. Anzures-García, M., Sánchez-Gálvez, L.A., Hornos, M.J., Paderewski-Rodríguez, P. Ontology-Based Modelling of Session Management Policies for Groupware Applications. LNCS, vol. 4739, pp. 57--64. Springer-Verlag, Berlin (2007)