Computing and Informatics, Vol. 30, 2011, 7–30

TOWARDS A SEMANTIC-AWARE COLLABORATIVE WORKING ENVIRONMENT

María A. Martínez Carreras, Juan M. Marín Pérez
Jorge Bernal Bernabé, José M. Alcaraz Calero
Gregorio Martínez Pérez, Antonio F. Gómez Skarmeta

Departamento de Ingeniería de la Información y las Comunicaciones
University of Murcia
Facultad de Informática, Campus de Espinardo
30071 Murcia, Spain
e-mail: {amart, juanmanuel, jorgebernal, jmalcaraz, gregorio, skarmeta}@um.es

Revised manuscript received 22 October 2010

Abstract. Collaborative Working Environments (CWEs) enable an efficient collaboration between professionals, specially those settled in different locations of a company or stakeholders from different companies. This can be of great help for small and medium enterprises (SMEs), as an effective way to share information. However, it can be difficult for SMEs to have access to a fully integrated CWE providing different tools (e.g., videoconferencing, instant messaging, etc.). Currently, they may define a CWE as a combination of heterogeneous and non-integrated tools which are not able to share information between them. An integrated CWE would provide SMEs with the necessary means to collaborate, making information exchange easier.

Keywords: Collaborative working environments, semantic web, ontologies, interoperability, SOA

Mathematics Subject Classification 2000: 68T30, 68M14, 68Q55, 68T35, 68U35

1 INTRODUCTION

Nowadays organizations deal with complex social network for the collaboration among other companies, customers and their own employees. For maintaining these relations, enterprises need the use of collaborative applications which constitute the underpinnings for communication and coordination between their stakeholders [1]. Collaboration can bring important advantages such as improvements in innovation and creativity, reducing costs and obtaining new knowledge from other partners [2]. One of the major benefits of adopting collaborative systems is that it makes easy to communicate among geographically dispersed people, providing virtual places where people can share opinions, information or ideas. Prominent examples of these systems in the business area are Content Management Systems (CMS), Electronic Meeting Systems (EMS) or Instant Messaging (IM).

However, the main problem behind using different kinds of collaborative tools is the lack of interoperability between them [3]. In this sense, organizations may need to create their Collaborative Working Environment (CWE) which consists on a range of computer and collaborative technologies such as email, instant messaging, chat rooms, discussion boards, shared whiteboards, mobile communication, media spaces and videoconferencing, among others [4, 5]. However, one of the main problems concerning CWEs is the lack of standards and common definitions for mediating the exchange of information and ease the communication between different kinds of collaborative applications. Therefore, the use and adoption of collaborative applications to achieve a fully interoperable CWE may be difficult for small and medium enterprises (SMEs), since its deployment may imply considerable efforts regarding economic resources, learning trends and new development issues. The achievement of a fully integrated CWE which enables the integration of legacy tools currently used by the SMEs is one of the research challenges addressed by this proposal.

The design of interoperable systems and Service-Oriented Architectures (SOA) is currently going in the direction of using Web services standards [6]. Nevertheless, these standards are usually based on XML descriptions which may not provide enough expressiveness to achieve complete interoperability. Additionally, semantic information may be also needed to describe the data and information managed by these systems, especially when different tools have to understand it. Therefore, the use of ontologies seems to be a promising option for obtaining a whole integrated CWE. Some ontologies have emerged with the aim of representing collaborative aspects such as Friend of a Friend (FOAF) [7], Semantically Interlinked Online Communities (SIOC) [8] and Online Presence Ontology (OPO) [9]. These ontologies are useful to represent information related to people, systems and online presence, respectively. However, they do not deal with the actions that users can perform in a CWE which may be useful to achieve interoperability. Thus, taking into account such actions in the ontologies, it is possible to execute actions for a given application as a result of other actions taken in the CWE. For instance, considering a scenario where a SME is currently using a Content Management System (CMS) to manage

shared documents and an Instant Messaging (IM) application to provide user communication, current approaches do not support a semantic-aware integration of these already existing tools to enable users notification by IM upon document changes in the CMS.

The creation of a semantic-aware CWE which allows to define and control the flow of information between collaborative applications is an important challenge, offering facilities to organizations (in particular to SMEs) and enabling users to customize the behavior of the CWE to their needs. Semantic Web technologies enable the representation of semantics and endow with reasoning capabilities which can be used to face this challenge. This paper presents an architecture which makes use of ontologies and rules to enable interoperability between heterogeneous collaborative tools, in order to achieve a fully integrated CWE.

The remainder of this paper is structured as follows. Section 2 presents some current works related to CWE integration, including outstanding ontologies and models in this field. Section 3 delves into our semantic representation approach describing the proposed ontology. The architecture designed for integrating different collaborative tools is described in Section 4. Section 5 includes a sample scenario, which makes use of our approach in order to show the collaboration of different project members using different collaborative tools. Some performance results regarding the feasibility of the proposal are exposed in Section 6. Finally, conclusions and future work lines are described in Section 7.

2 RELATED WORK

There exist several CWEs which integrate a set of collaborative tools aiming to allow a group of people to share information, ideas, etc. Although these environments have integrated some collaborative tools, there are others that have to be used separately without any integration within the CWE, such as e-mail and Instant Messaging. Examples of this kind of environments are Basic Support for Cooperative Work (BSCW) [10], Alfresco [11] and EMC Documentum [12].

These environments support document uploading, group management and forums, using the folder concept to group a set of documents or other kinds of objects. In the case of BSCW, this system does not follow any standard or specification to represent the data. In order to integrate external tools, it provides an XML-RPC API and WSDL services. On the contrary, Alfresco and EMC Documentum offer their services following the Content Management Interoperability Services (CMIS) [13] which is currently an OASIS standard. This specification defines a set of generic services for interoperability between this kind of systems. Thus, both environments provide a set of WSDL services which enable them to interoperate with other applications. In this sense, interoperability between some elements in EMC Documentum and Alfresco can be achieved by means of CMIS. However, these CWEs lack any mechanism to enable its integration with other collaborative tools, especially those providing synchronous collaboration like Instant Messaging, which are quite used

in daily work and provide several advantages such as fast communication and increasing productivity of employees [14]. Furthermore, CMIS does not include any support to integrate these tools in the CWE.

Another example of tools which help to manage the notion of group and offer similar features than the systems analyzed before is the GoogleWave [15]. More precisely, it includes several other Google tools which allows the collaborative management of wikis, messages from e-mail or instant messaging and social networks. Instead of using the concept of space or folder, this tool presents the concept of wave for indicating the container of some objects that collaborators need. Concerning the creation of a fully integrated CWE where different legacy applications can be integrated in the system, this tool does not offer mechanisms for interoperability with external tools.

In this sense it is important to establish mechanisms for integrating diverse systems in the CWE. With the aim of solving problems of this kind avoiding recoding existing tools and data for being integrated in a CWE, the definition of an ontology which fulfills this gap is needed.

Several approaches like Friend of a Friend (FOAF) [7], Semantically Interlinked Online Communities (SIOC) [8] and Online Presence Ontology (OPO) [9] make use of ontologies with the aim of providing interoperability between collaborative systems.

Concerning the definitions of ontologies in the collaborative field, the Friend of a Friend (FOAF) [7], Semantically Interlinked Online Communities (SIOC) [8] and Online Presence Ontology (OPO) [9] ontologies are remarkable example. FOAF is an RDF ontology aimed to facilitate sharing the information about people and their activities since it allows to transfer information between different systems with a common vocabulary. SIOC is an ontology intended for enabling the interoperability of online communities, offering a common vocabulary for representing data in such a kind of communities. Likewise, OPO is an ontology to describe the vocabulary involved in the representation of online presence information regarding users of Instant Messaging tools. As can be noticed, the previous ontologies only describe information about state and data regarding the relations of the different entities. Therefore, the use of the above-mentioned ontologies for the creation of rules which indicate the action to be performed according to an expected behaviour of the CWE is not possible. For this reason we have developed an ontology able to include the actions allowed in the CWE, which in turn integrates concepts and relations of the above-mentioned ontologies.

Some approaches which take into account actions and events in collaborative systems are proposed by Viei et al. [16] and Haake et al. [17]. They present a domain collaborative model based on the Model View Controller (MVC). However, we consider that the management of views is the responsibility of the application providing this view and it should not interfere with other applications to achieve interoperability in a CWE. Moreover, the actions they present only reflect the management of workspaces systems, and our proposal deals with different kind of applications and current standards such as CMIS.

Other work which describes the use of actions for managing computer mediated interactions is provided by Schummer [18]. More precisely, this work details the actions concerning the graphical user interface (GUI) and the actions regarding services. The latter is the kind of action we are going to use in our model as we indicate in Section 3.2.

Other interesting works with regard to CWE interoperability are the Ecospace [19] and the InContext [20] projects. The former provides an open reference architecture for creating an interoperable and integrated CWE. However, this architecture is so open and wide that it may be difficult for SMEs to obtain their integrated and interoperable CWE. The latter is oriented to obtain adaptive CWEs and the use of context information is focused on selecting the accurate service for performing an action. The architecture presented implies the creation of new applications for managing existing collaborative tools, which will be selected according to some context information in the system. On the contrary to this schema, our proposal allows users to manage directly existing collaborative tools, and our architecture is responsible for listening the actions performed in these tools and analyzing whether further actions should be done, such as the notification via Instant Messaging. Additionally, our proposal supports the obtainment of new knowledge from contextual information of the environment or specific situations occured in it.

3 SEMANTIC WEB REPRESENTATION

With the aim of representing generic rules for different kinds of collaborative applications, an ontology has been defined which offers a common vocabulary for the different applications taking part in the CWE. This ontology enables the definition of the rules which govern the behavior of the CWE in order to get an integrated execution of the involved applications.

3.1 Rule Representation

The Ontology Web Language 2 (OWL 2) [21] is a W3C standard which enables the specification of ontologies, defining class hierarchies and their relationships, associated properties and cardinality restrictions. Models defined using OWL 2 can be enriched by means of the Semantic Web Rule Language (SWRL) [22]. This language is used to represent rules on the Semantic Web and it extends OWL 2 in order to provide a way to express conditional knowledge.

The combination of OWL ontologies and SWRL to specify rules offers the advantage of allowing automated reasoning. This is carried out by a reasoner, referring to a specific piece of software which performs reasoning processes. These processes constitute a remarkable added value of the usage of Semantic Web technologies, since they are able to infer new knowledge, that is, deriving additional information not explicitly specified in the ontology.

The solution presented in this paper makes use of these technologies. OWL is used to provide an ontology which represents the different applications which are

part of the Semantic-Aware CWE. SWRL rules are then specified, based on the concepts and relationships described in this ontology, to describe the desired behavior of the Semantic-Aware CWE in the organization. This enables the description of rules like *"if a user receives an instant message and is off-line, then send this message to the SMS service in order to forward it to his/her mobile phone"*.

3.2 CWE Ontology

This section presents the ontology used in this proposal to describe the concepts and relationships which are used by the rules to specify the behavior of the Semantic-Aware CWE. The ontology integrates SIOC, FOAF and OPO, as well as other concepts and relationships which have been identified, trying to provide a wider conceptualization of a CWE. Fostering extensibility, organization and clarification, the ontology has been divided into several parts or modules. A *Core module* defines the main relationships and concepts, including *Collaborative Applications*, *Actions* and *Policies*, among others. Then, other modules extend this core definition, focusing on each one of these main concepts.

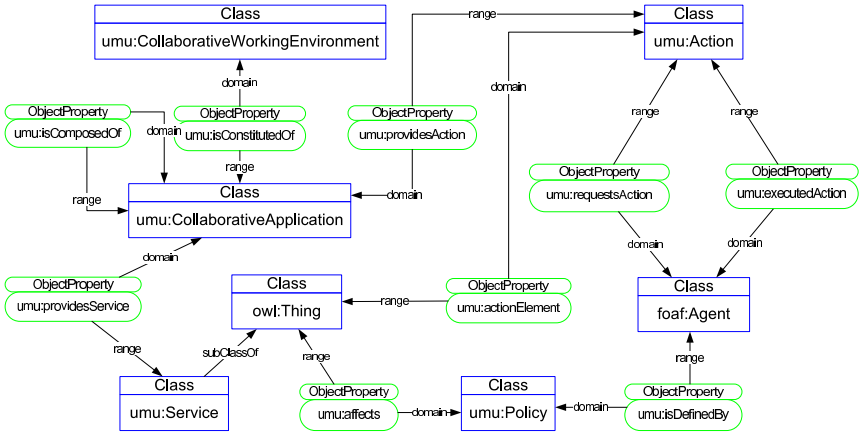


Fig. 1. Ontology core

Figure 1 shows the *Core* module we have designed in our ontology. According to [4] and [5], a *Collaborative Working Environment* is defined as the composition of different *Collaborative Applications* which, in turn, provide a set of *Services*, as it is depicted in 1. For instance, a *VideoConference* application is composed of different applications, such as *Chat*, *Video* or *Whiteboard*. These applications may provide different services. For example, a *Video Conference* application may offer *Video Call Service*. For representing this structure we have included the concept *CollaborativeWorkingEnvironment* which is related with *Collaborative Applications* by means of the relation *constitutedOf*. With the aim of indicating that *Collaborative Applications* provides some services, we have included the relation *providesServices* as well.

As a consequence, following this respresentation we can identify what services are provided for a specific tool.

Considering the work of Schummer [18] each application provides a set of *Actions* which may be executed, hence we have reflected it using the *providesAction* relationship between the concept *Action* and the concept *CollaborativeApplication*. Observing Figure 1 we can appreciate that an *Agent* (human or computer) can execute *Actions* over anything managed in the CWE, like *Services*, *Collaborative Applications* and so on. Finally, we have included the concept *Policy* indicating that it is defined by an *Agent* including the relation *isDefinedBy*. Since the policies may affect any component in the system we have integrated the relation *affects*. More precisely, these policies represent the rules that govern the collaborative behavior of the different applications which form the CWE.

Although SIOC, FOAF and OPO do not cover the necessary elements to represent a whole CWE domain, they provide meaningful concepts related to collaborative applications. Thanks to the flexibility of ontologies, these concepts can be imported and they are used in our proposed ontology. In this regard, it should be noted that Figure 1 uses a "Prefix:Concept" notation, where the prefix determines the source on which the concept has been described. Thus, *sioc*, *foaf* and *opo* prefixes are referred to these ontologies, respectively, whereas the concepts with the *umu* prefix are introduced in this approach.

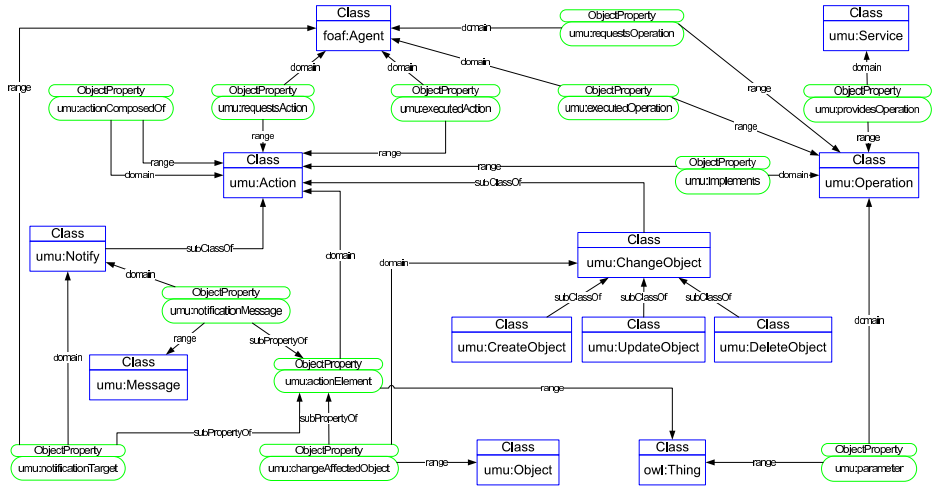


Fig. 2. Actions representation

With regard to the Action module, it is represented in our ontology as depicted in Figure 2. This module describes a common vocabulary for the actions that different applications can provide and users can invoke. Actions represent the generic notion of something which can be performed by the system and they are linked with more specific aspects regarding the *Operations* that can be used to perform the

actions and the *Services* that provide these operations. As an example, the *Notify* and *ChangeObject* actions are shown in the figure. In this respresetantion we have considered some concepts of the CMIS structure. Indeed, the subsequent concepts *CreateObject*, *UpdateObject* and *DeleteObject* are taken from this standard. What is more, the ontology offers the existing relations regarding the *Action*, *Object* and *Agent*. Analyzing business we can observe the existence of simple and complex *Action*. In fact, the latter refers to the composition of simple *Action*, which is depicted in our ontology using the relation *actionComposedOf*.

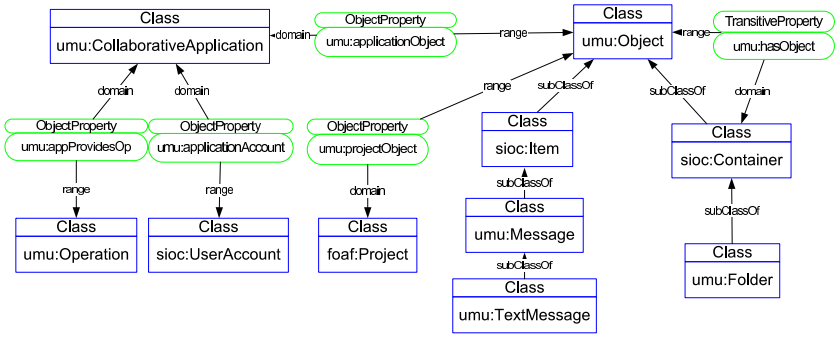


Fig. 3. Collaborative Applications representation

Figure 3 depicts a portion of the *CollaborativeApplication* module of our ontology. It extends the definition of *Collaborative Application*. The figure also shows the integration of some concepts from SIOC and FOAF. Considering collaborative application we can state that a *Collaborative Application* manages different objects or resources which are the artifacts for offering information in the collaboration. This fact is reflected in our ontology by means of the relation *applicationObject* which links *Collaborative Application* and *Object*. An object can be a simple object denoted as *Item* or a collection of objects denoted as *Container*; for instance, imaging the case of BSCW, Alfresco or Documentum which manages the concept of folder (or space) and documents. In these cases a document can be considered as an Item, while a folder or space can be considered as a Container. Concerning the *Container* concept, it is linked with *Object* by means of the *hasObject* property. This property identifies the objects that belong to the container and it is defined as transitive relationship. Furthermore, we distinguish the *Folder* concept as a kind of *Container*. Thus, the ontology represents that a *Folder* may contain other objects, including other *Folders*. This concept models a folder managed in environments such as Alfresco, BSCW or EMC Documentum.

Additionally, *UserAccount* is also imported from SIOC to represent user accounts in applications. Usually, the notion of projects is used in companies working groups, thus we have created the concept *Project* in our representation. More concretely, when workers collaborate in different projects they could use different collaborative applications according to the goals of the project. Therefore, different objects from

CollaborativeApplication can be managed in a *Project*; thus, we depict this relation by means of the *projectObject* property, as depicted in Figure 3. The *UserAccount* and *Project* concepts also appear in the *User* module, where they are associated to other user-related concepts. Figure 4 depicts a portion of this module.

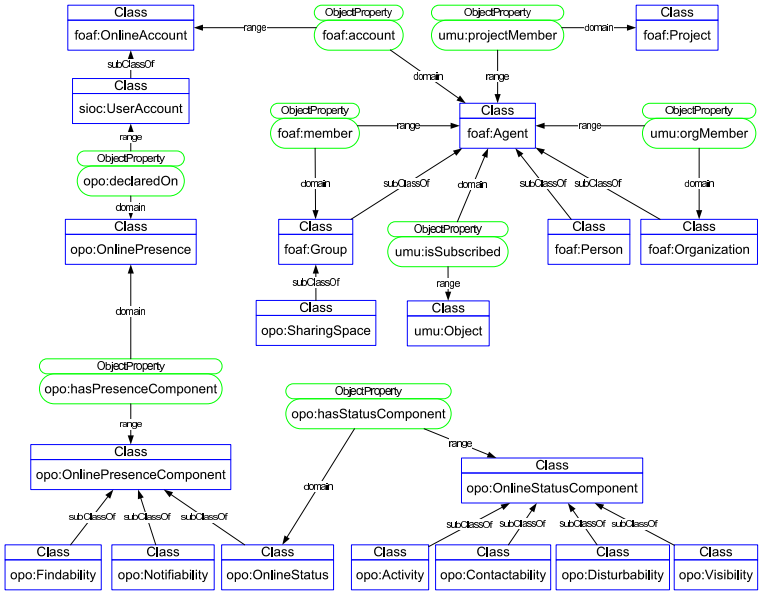


Fig. 4. User-related concepts representation

The *User* module includes several concepts imported from the FOAF ontology. This module is focused on the *Agent* concept, specifying different subkinds of this concept like *Person*, *Group* and *Organization*. An *Agent* can be a member of a *Project*, a *Group* and an *Organization*. Moreover, an *Agent* can be subscribed to *Objects*, indicating its interest in being notified upon object changes. On the other hand, the module also links with the OPO ontology to represent online presence characteristics of users. Thus, an *Agent* may have one or more *OnlineAccounts* on which *OnlinePresence* aspects can be defined. *OnlinePresence* may be composed of several *OnlinePresenceComponents*. Among them, the *OnlineStatus* is used to represent the attitude of an *Agent* towards the possibility of communicating with other *Agents* and, in turn, it may also be composed by several *OnlineStatusComponents*.

Other modules not exposed in this section due to space limitations include *Service* and *Policy*. The first one extends the *Service* concept, including subtypes like *WebService*, commonly used in distributed systems. The *Policy* module defines the concepts and relationships about rules, linking with the ontology provided by SWRL to enable the description of SWRL rules.

The modular design of this ontology allows its extension with specific definitions for different kinds of collaborative applications. Two specific modules have been

defined in order to provide a concrete scenario (depicted in Section 5) and to test the proof of concept implementation exposed in Section 6. Concretely, these two modules model an Instant Messenger application and a Content Management System (CMS). This last one, for example, includes a set of specific subtypes of the *Operation* concept, based on the operations defined by the CMIS specification.

4 ARCHITECTURE

This section describes the proposed architecture to enable the integration across heterogeneous collaborative tools. Figure 5 shows an overview of the different layers available in the architecture.

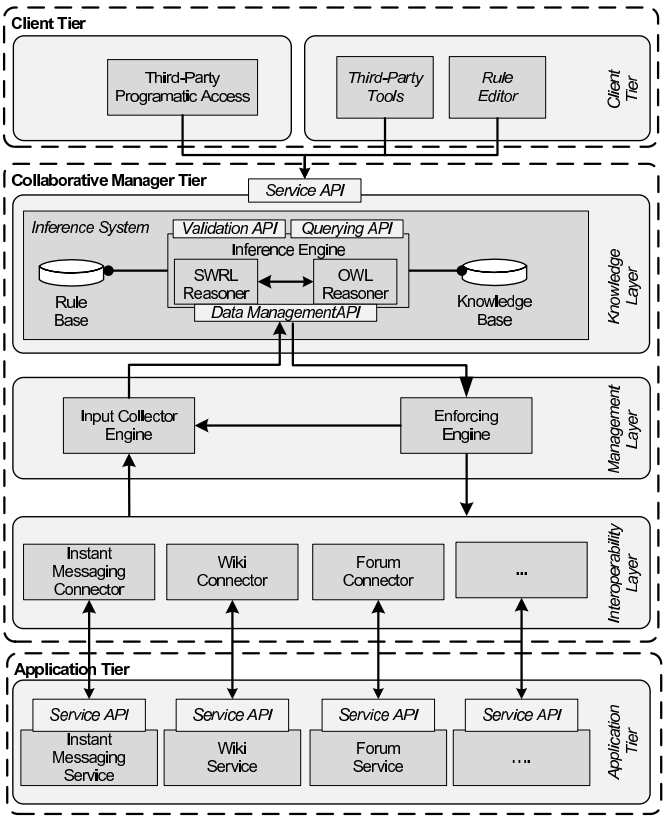


Fig. 5. Architecture overview

The *Application Tier* is composed of the set of heterogeneous collaborative applications that are managed in the Semantic-Aware CWE. These applications are not integrated and they do not provide any interaction capability between them.

Usually, these applications expose an API to manage them externally. *MSN*, *Skype* and *Alfresco* are examples of these applications. The *Application Tier* is used by the *Collaborative Manager*, which is the main component of the architecture. It provides the capabilities to perform smart collaboration between the different applications and it is structured in three layers: the *Interoperability Layer*, the *Management Layer* and the *Knowledge Layer*.

The *Interoperability Layer* provides an extensibility mechanism to enable the incorporation of new collaborative tools in the architecture. This layer homogenizes the interaction with the tools, providing a common language for knowledge representation and a common interface for interacting with these tools. This layer is composed of a set of *Connectors*, which interact with the *Service API* of the different tools in order to provide a representation of the knowledge extracted from them in OWL language, using the ontology exposed in Section 3.2. Moreover, this layer is also able to interpret the knowledge inferred from higher layers in order to enforce it by means of method invocations to the APIs of the tools. As an integration example, a connector has been developed for the implementation used for the testbed exposed in Section 6 to connect with the IM service. The *Service API* exposed by this service is defined by means of a WSDL specification. Thus, the connector makes use of Web Services technologies to interact with it. Moreover, in order to receive notifications from the IM service, the connector implements the WS-Notification specification. This enables the connector to be subscribed to the service and to receive events providing status information. Upon the reception of an event, the status information is represented by the connector in RDF and passed to the *Input Collector Engine* of the *Management Layer*.

The *Management Layer* collects and aggregates the information from the different tools and provides it to the *Knowledge Layer*. This process is done in the *Input Collector Engine* available in this layer. Moreover, the *Management Layer* also splits and routes the inferred knowledge generated on the *Knowledge Layer*, sending it to the associated connector in order to synchronize the tools accordingly and enabling collaboration among them. This process is done by the *Enforcing Engine* component available in this layer. Moreover, this layer controls that the inferred knowledge is correctly enforced in the tools and it provides feedback to the *Input Collector Engine* about possible errors or exceptions which may be raised by the different architecture components or tools during enforcing, including data inconsistency errors, unsupported semantics or application failures.

The *Knowledge Layer* provides an inference system to manage the knowledge shared across the applications and the behavior of the system. The *Knowledge Layer* is the core layer of the proposed architecture. It receives the information provided by the different tools and keeps it in the *Knowledge Base* (KB). This information, together with the rules presented in the *Rule Base*, is used to infer the new knowledge and actions to be enforced to applications. The *Rule Base* is a repository of SWRL rules to be applied to the inference system. The inference process is done in the *Inference Engine* component. As a result of this process, new inferred knowledge is obtained and it is sent to the *Management Layer*.

Finally, the *Client Tier* represents different parties using the proposed architecture. This includes management tools, which may be used to control different aspects of the *Collaborative Manager*, such as configuration, monitoring and management. For instance, ORE [23] is a rule authoring tool developed in our research group intended for making easy the creation, management, debugging and testing of SWRL rules, providing an intuitive interface for these purposes. This tool can be used in order to define, test and validate the collaborative behaviors among the different tools and to enforce these rules using the service API exposed by the *Collaborative Manager*. Moreover, this service API can also be used by some tools aimed to aid non-expert users to define policies and manage the system. These tools may include policy templates exposed via Web interfaces or even simplified GUI applications with wizards designed to guide users with little SWRL knowledge in the definition of policies and rules.

The *Inference Engine Component* carries out the reasoning process using an OWL and SWRL reasoner. This reasoner uses both the *Knowledge Base* and the *Rule Base* in order to perform the reasoning processes. On the one hand, the *Knowledge Base* contains the information provided by all the applications managed in the system. These applications use an homogeneous way to represent their information by means of the ontology exposed in Section 3.2. On the other hand, the *Rule Base* contains the set of rules which define interactions, interchange and behaviors that specify the way in which the applications may collaborate. From the knowledge perspective, the rules are also part of the *Knowledge Base*.

The main operations of this reasoner are:

- Inference. New knowledge is inferred using the information available in the ontology. Moreover, SWRL rules are applied as part of the inference process. These rules derive new individuals or OWL instances which are used to determine the information related to the collaboration among applications.
- Validation. The OWL language allows constraints to be expressed; the validation operation is used to detect when such constraints are violated by a data set. In other words, the validation consists in a global check across the schema and instance data looking for inconsistencies.
- Querying the ontology, including instance recognition and inheritance recognition. The former consists in testing if an individual is an instance of a class and the latter in testing if a class is a subclass of another class or if a property is a sub-property of another one. This enables the formulation of generic queries referring to abstract concepts, being the system able to recognize instances belonging to concrete subclasses or sub-properties of such concepts.

The inference process is carried out by the reasoner when the *Input Collector Engine* provides new information to the *Knowledge Base*. The reasoner uses the *Knowledge Base* to apply the semantics defined in the OWL ontology and the SWRL rules. As output, the inferred information is obtained and stored in the knowledge base as well. This information is used to determine the set of actions that the system

has to enforce in order to produce the collaboration between applications. Validation is also performed, checking the ontology for inconsistencies or constraint violations. Finally, the inferred knowledge is sent to the *Enforcing Engine* which interprets this knowledge and performs the needed invocations on the target applications.

For this reasoning process to be efficient, an incremental approach is used. Some OWL reasoners like Pellet [24] support incremental consistency. This functionality enables the reasoner to take into account only the new knowledge which may have been added since the last reasoning performed with the same *Knowledge Base*. In this approach, the *Knowledge Base* is updated and the new knowledge inferred during the reasoning process is also kept in it. Since this KB remains the same, incremental consistency can be used to make the reasoning process faster.

The inference engine provides some advantages to the architecture. It performs formal validation of the knowledge available in the *Knowlege Base*, avoiding any inconsistency on the data. This enables the definition of policies to customize the application behavior and to perform detection (and provide a potential resolution) of conflictive situations.

5 SCENARIO DESCRIPTION

In order to illustrate the approach presented in this paper, a sample scenario has been described as working example. Let us suppose an organization which uses diverse tools to enable collaboration between employees which are even in different locations. Concretely, the organization is involved in a project in which a Content Management System (CMS) is used to share documents between participants, as well as an Instant Messenger (IM) and an email service which are used for communication. In this situation, for these tools the organization is willing to interoperate, getting users notified when a document is modified in the CMS. Suppose that Bob and Alice, who are located in different offices, are working on the creation of a leaflet for the company. Thus, they should have a shared folder in the CMS for delivering the needed documents and the IM for notification and communication.

To this end, some rules are defined by the administrator to specify that users should be notified about changes. Namely, Rule 1 is defined to subscribe users which are member of a project to the CMS folders belonging to that project. In turn, Rule 2 is defined to notify users about changes in the folders to which they are subscribed. These rules can be defined by means of several tools like ORE, designed to aid in SWRL rule authoring. It is worth mentioning that these rules make use of different concepts related with project membership, object subscription, notification and online presence, among others. These concepts are defined in the ontology exposed in Section 3.2, although some of them are not explicitly depicted for clarity reasons.

Rule 1 states that if an agent (person) is member of a project (1), then, for every CMS folder belonging to that project (2), the agent is subscribed to that folder (4). Rule 2 states that if an object (document or folder) has changed (1) and an agent is

$$
\begin{aligned}
&Agent(?a) \wedge Project(?p) \wedge projectMember(?a, ?p) \wedge && (1)\\
&Folder(?f) \wedge projectObject(?p, ?f) \wedge && (2)\\
&\rightarrow && (3)\\
&isSubscribed(?a, ?f) && (4)
\end{aligned}
$$

Rule 1: Subscribe project members

$$
\begin{aligned}
&ChangeObject(?co) \wedge Object(?o) \wedge changedObject(?co, ?o) \wedge && (1)\\
&Agent(?a) \wedge Object(?s) \wedge isSubscribed(?a, ?s) \wedge hasObject(?s, ?o) && (2)\\
&\rightarrow && (3)\\
&Notify(?n) \wedge notificationTarget(?n, ?a) \wedge requestsAction(\#System, ?n) \wedge && (4)\\
&TextMessage(?m) \wedge notificationMessage(?n, ?m) \wedge text(?m, \text{“}Document\ changed\text{''}) && (5)
\end{aligned}
$$

Rule 2: Notify changes

subscribed to (any parent of) this object (2), then the agent should be notified (4) about this change (5). In this rule, it should be noted that the *hasObject* property is defined as transitive, getting any person subscribed to a folder being notified about changes in any document belonging to that folder.

Moreover, Rule 2 specifies user notification by means of the *Notify* action. Some rules can be defined to establish different ways in which users should be notified, selecting the specific operation to be used for notification. These rules can be defined by the administrator to apply for every user or they may be defined by users by means of a guided wizard, allowing them to specify their preferences for notification. In this case, Rule 3 would be defined by the administrator to state that notifications should be preferably delivered by the IM if the user to be notified is available online. Finally, a fourth rule (not depicted for clarity reasons) would specify notification by email for users which are not available online and cannot be notified by IM.

$$
\begin{aligned}
&Notify(?n) \wedge requestsAction(\#System, ?n) \wedge Agent(?a) \wedge notificationTarget(?n, ?a) \wedge && (1)\\
&TextMessage(?m) \wedge notificationMessage(?n, ?m) && (2)\\
&InstantMessenger(?im) \wedge UserAccount(?ua) \wedge applicationAccount(?im, ?ua) \wedge account(?a, ?ua) \wedge && (3)\\
&OnlinePresence(?op) \wedge declaredOn(?op, ?ua) \wedge && (4)\\
&OnlineStatus(?os) \wedge hasPresenceComponent(?op, ?os) \wedge hasStatusComponent(?os, \#Available) \wedge && (5)\\
&SendInstantMessage(?s) \wedge appProvidesOp(?im, ?s) && (6)\\
&\rightarrow && (7)\\
&requestOperation(\#System, ?s) \wedge sendIMMessage(?s, ?m) \wedge sendIMTarget(?s, ?ua) && (8)
\end{aligned}
$$

Rule 3: Notify by IM when available

Rule 3 states that if an agent should be notified (1) with a given message (2) and there is an IM application for which the agent has an account (3) whose online presence (4) has an online status which indicates that the agent is available (5), then the operation provided by the application to send instant messages (6) should be executed to deliver the message to the agent (8).

Considering this scenario, let us suppose the use case in which Alice adds a new document to a project folder in the CMS. Figure 6 depicts a sequence diagram showing the different components of the architecture which are involved and how they interact for this use case.

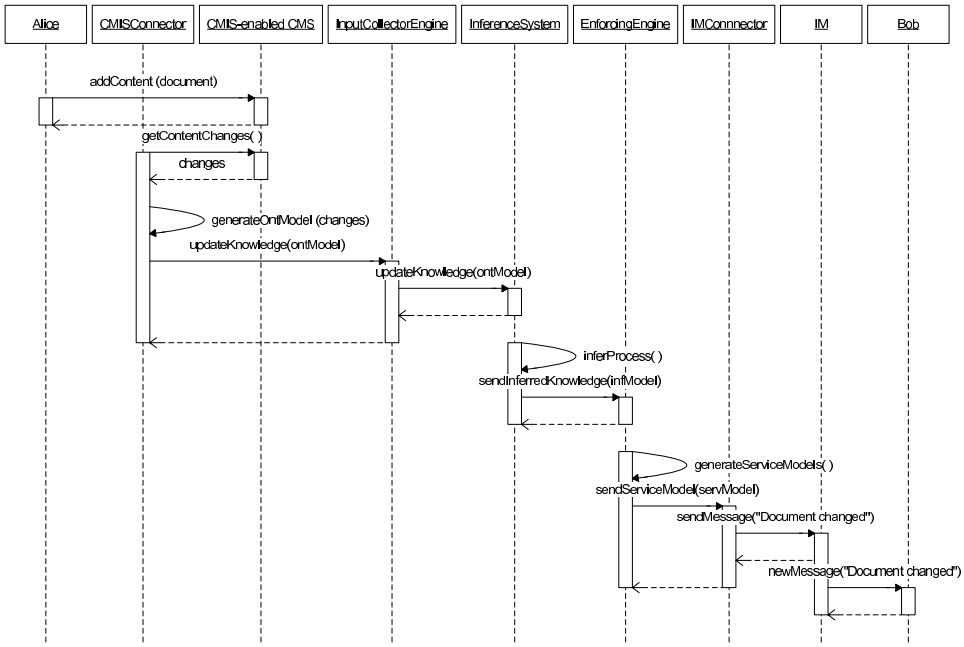


Fig. 6. Sequence diagram

First of all, Alice adds the document to the shared folder in the CMS by means of the particular CMS interface. To illustrate the use of standard connectors in the architecture, we suppose that the CMS is compliant with the CMIS specification. Thus, a CMIS connector is associated with the CMS in the architecture and it gets the changes using CMIS operations. It processes those changes, generating the ontology model which represents them. In this case, an instance of *Document* and the *CreateObject* operation are generated, together with their corresponding properties, representing that Alice has added the document to the folder. Then, the connector makes use of the *Input Collector Engine* to update the knowledge in the *Inference System* with the ontology instances representing the changes.

When the *Inference System* receives new knowledge from the *Input Collector Engine*, it starts an inference process. During this process, the SWRL rules are applied and the semantics defined in the ontology model are taken into account to infer new knowledge. In this case, since Bob is a member of the project, Rule 1 infers that he is subscribed to the shared folders of that project in the CMS. Moreover, the change made by Alice has been detected by the CMIS connector and collected by the *Input Collector Engine*. Thus, Rule 2 infers that Bob should be notified about changes. In turn, assuming that Bob is available online, Rule 3 infers that the notification should be done via IM.

The inferred knowledge is forwarded to the *Enforcing Engine* for performing the required operations in the services. This component splits the received ontology model in submodels for each service which is involved. In this case, just the IM service is needed for the notification. So, it is contacted through the IM connector, which receives the submodel that applies to this service. The connector interprets the ontology model and performs the corresponding operations in the service. Concretely, in this use case, a *sendMessage* operation is needed to notify Bob. Thus, the connector invokes this operation in the IM service. At this point, if no error is raised by the enforcing module or the IM service during the invocation, the message is delivered by the IM service to Bob. In case of an error, it is reported to the *Enforcing Engine*, which makes use of its connection with the *Input Collector Engine* to inform the *Inference System* for it to take the necessary actions at knowledge level.

As can be seen, the approach enables the interoperation of different kinds of collaborative tools, achieving a CWE composed by heterogeneous applications which were initially unable to interact. In this scenario, a standard-based connector based on CMIS is used in the case of the CMS. In case of proprietary services not supporting standard interfaces, a particular connector can be developed for that service. That is the case of the IM, which makes use of a specific connector for this particular service.

Moreover, the semantic approach by means of OWL ontologies and SWRL rules provides high flexibility to the system. For instance, although Rule 3 has been defined for any person (agent) for clarity reasons, it could be modified to take some other criteria into account, for example, to notify only the users belonging to a given project or group. With this approach, the organization may establish what changes should be notified (second rule of the scenario shown in Rule 2), while users can establish their preferences about how they prefer this notification to be carried out (Rule 3).

Finally, the constructs provided by OWL-DL to represent semantics combined with SWRL rules enable the definition of highly expressive rules. For instance, the *TransitiveProperty* constructor is employed in the ontology model for the *hasObject* property. Thus, Rule 1 can make use of the semantics defined for this property in line 2 to notify changes in any document belonging to any subfolder of the folder to which the user is subscribed, regardless of the level of the document in the folder hierarchically.

6 IMPLEMENTATION AND PERFORMANCE

In order to evaluate the feasibility of the proposal, the scenario defined in Section 5 has been taken as reference to carry out different tests. Due to the fact that the *Knowledge Layer* is the most important and critical element in the architecture presented in Section 4, the measurements have been focused on the times required by this layer to carry out its tasks. In this regard, a proof of concept Java application has been developed to deal with the inference system and manage the ontology. The application defines the interfaces allowing the management of the CWE ontology, and providing, at the same time, elements to emulate the rest of the architecture.

Therefore, since the Inference System consumes most of the time required by the overall CWE architecture, this evaluation delves into the statistics obtained by the reasoner in charge of performing the inference process. This inference process takes into account both the ontology instances present on the KB and the rules that model the desired behavior. With the aim of evaluating the performance and the scalability of the proposal, several executions have been done using different amounts of individuals, i.e. OWL instances.

Regarding Description Logics (DL) reasoners, currently there are several suitable implementations such as Pellet [24], Jena [25], KAON [26] or the one proposed by FaCt++ [27]. Our application makes use of Jena (version 2.6.2) as a general Java API to manage ontologies and Pellet (version 2.0.0) as DL reasoner. We have chosen Pellet as reasoner as it supports high expressiveness dealing with OWL 2 ontologies and it is also able to perform incremental consistency checking. Likewise, Jena is nowadays the standard de facto Java library to manage ontologies.

The simulations have been run in a Core 2 Duo T7500 at 2.2 GHz with FSB 800 MHz and 4 GB RAM, with Ubuntu Linux version 8.0.4 and the Sun JDK 1.6 configured to have up to 2 GB of maximum heap size.

The number of different kinds of individuals contained in the KB for a specific execution is referred as population. Thus, a population represents the knowledge handled by the CWE at a given moment. Each population is composed by individuals representing instances of different OWL classes defined in our ontology. Figure 7 depicts the percentages of every kind of individual which have been used to make up each population. These percentages have been selected in order to represent a realistic CWE also taking into account the specific individuals related to the scenario. Thus, the individuals for a given population have been randomly generated, but in a driven way in order to achieve the desired distribution which complies with the scenario. Since the complexity of the rules can influence the performance results significantly each population also contains the SWRL rules described in Section 5.

As can be seen in Figure 7, there are some individuals such as the *TextMessage* or the *InstantMessenger* which are specifically included in the populations because of its relationship with the described scenario. Moreover, an important part of each population refers to actions that take part in the CWE, like notifications or changes in objects.

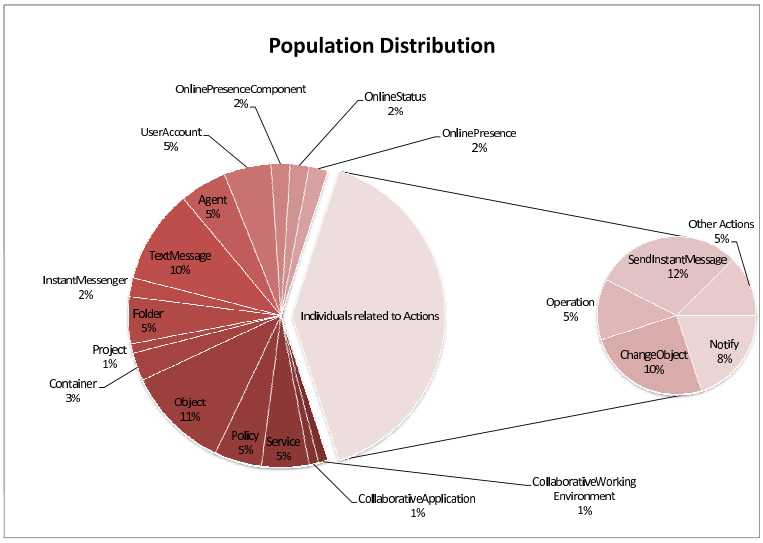


Fig. 7. Population distribution

Populations are used to quantify the time that the CWE architecture spends in making the inference process and checking the KB consistency. Different executions are performed using different size populations. Thereby, further complexity in the system is achieved by introducing new knowledge in the KB, since the greater the KB is, the more time it takes the reasoner to deal with it. As can be seen in Table 1, the tests make use of 10 incremental populations ranking from 300 up to 45 000 individuals. These individuals are represented by the reasoner by means of statements, where a single individual may need several statements to be represented. Thus, population 1 holds 300 individuals which require 1 200 statements for their representation. In turn, the 45 000 individuals held in population 10 are represented by nearly 150 000 statements. It is worth noting that the size of the biggest population has been calculated taking into account that, for this amount of statements, the reasoning process requires more than 10 seconds to perform the consistency checking, which we considered an unaffordable time for a CWE. The function which establishes the size of the different populations follows an exponential distribution in order to lead the system to the extreme situation of 45 000 individuals while showing intermediate results of the tests from the initial population of 300 individuals.

Population	1	2	3	4	5	6	7	8	9	10
Individuals	300	800	1 322	2 200	3 500	5 900	9 700	16 000	26 500	44 500
Statements	1 200	2 300	4 000	6 800	11 500	19 400	32 500	54 200	90 100	149 400

Table 1. Number of individuals and statements by population

As stated before, these populations are generated following an individual distribution which tries to represent the scenario described in Section 5. For instance, population 5, which has 3500 individuals, could represent a concrete situation of the scenario where, according to the percentages depicted in figure 7, the following individuals are represented: 35 different projects (1 % of the population) with 175 different shared folders (5 %) holding 385 documents (Objects 11 %) which are accessible to different users (Agents 5 %). Likewise, 175 collaborative services (5 %) such as the e-mail service and the InstantMessenger service described in the scenario lead the system to deal with 1 400 actions (40 %), like notifications to users carried out when certain documents are updated in the Content Management System.

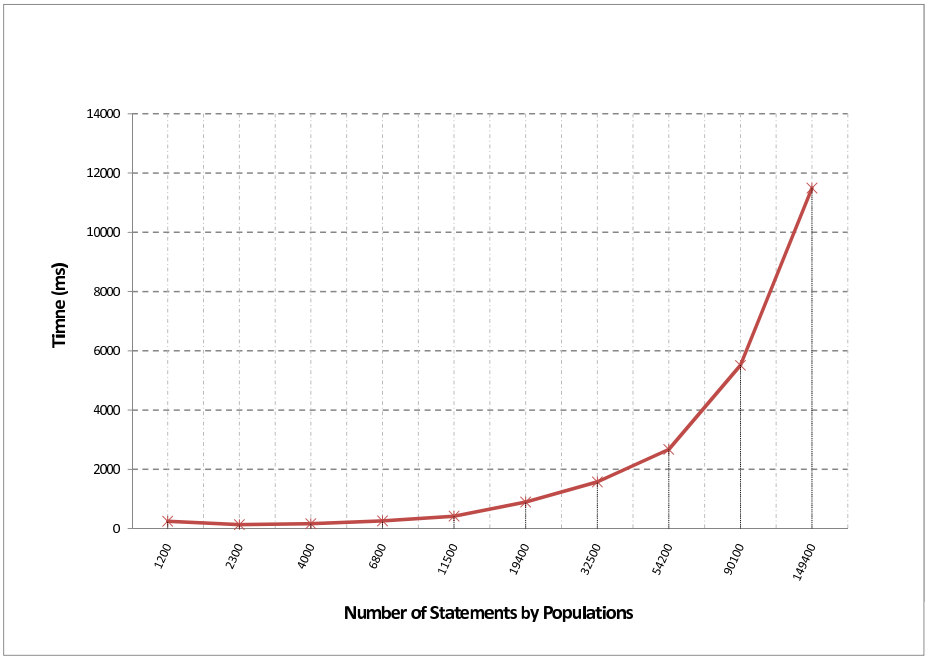


Fig. 8. Inference system performance

Figure 8 depicts the reasoner performance for the time it spends making the inference process throughout the different populations. The abscissa axis of the graph represents the amount of statements that contains each of the aforementioned 10 populations. To move from one population to the following one, the statements representing the individuals are added incrementally, in such a way that populations grow by adding new individuals, but keeping the ones present in the previous population. This emulates a CWE which adds new information to its existing KB. Moreover, this allows to take advantage of using the incremental consistency checking feature provided by the reasoner. Looking at the results, and bearing in mind the huge amount of statements managed, it can be said that the performance of the

reasoning process fits well in the times handled by a common CWE. Taking into account that SME would usually deal with medium sized populations, we consider that, making use of our proposal, SMEs would lead to important benefits for the creation of an integrated CWE.

7 CONCLUSIONS AND FUTURE WORK

The work presented in this paper provides the basis for the creation of a fully integrated Semantic-Aware CWE composed of diverse collaborative applications. In our solution, the use of semantic web technologies is crucial for obtaining integrated CWEs where different applications may react according to previous actions produced by other applications and/or the rules determining the business goals of the organization. We propose a CWE ontology based on OWL2 for describing the collaborative applications regarding the data and actions managed by them. Futhermore, the combination of this ontology with SWRL rules enables to define the behaviour of the CWE.

With the aim of obtaining and communicating the needed information of several collaborative applications, we have defined a layered architecture which obtains the actions and information occured in the collaborative applications, tranforms it to the knowledge undestandable in the CWE, and reacts by means of inference processes according to the rules defined and the information gathered in the system. An important advantage of this architecture, especially in SMEs, is that it includes the use of connectors that facilitates the integration of collaborative applications in it.

As a proof of concept we have described a typical scenario where the information uploaded in a CMS can be notified via IM according to the current status and preferences of the users and the organization. In order to analyze the impact of using semantic technologies in CWE, we have conducted some tests in which we have observed that our proposal result in acceptable times for common workloads. These results indicate that the use of the proposed semantic-aware architecture for CWE can be of interest for SMEs.

Considering that our work is a first step towards the creation of Semantic-Aware CWE, some issues, which constitute the future work of this research, are still open such as the integration of more collaborative applications in the ontology; the integration of more advanced policies for governing the whole system; or including more intuitive tools for integrating rules in the CWE. It is also worth mentioning the work we are doing on the evolution of the current ontology definition and the integration of conflict detection and resolution mechanisms as part of our approach.

Acknowledgment

Thanks to the Fundacion Seneca for sponsoring this research under its post-doctoral grants and the project 04552/GERM/06. Thanks to the Spanish MEC for partially supporting this research under the project TIN2008-06441-C02-02 SEISCIENTOS.

REFERENCES

[1] Stanoevska Slabeva, K.—Hoegg, R.: Classifications of Collaborative Working Environments in Organizations. In CollECTeR: Collaborative Electronic Commerce Technology and Research, June 2006, pp. 173–183.

[2] Pallot, M.—Sandoval, V.: Concurrent Enterprising: Toward the Concurrent EnterPrise in the Era of the Internet and Electronic Commerce. Springer Berlin/Heidelberg 1999, pp. 292–297.

[3] Prinz, W.—Martínez-Carreras, M. A.—Pallot, M.: From Collaborative Tools to Collaborative Working Environments. International Journal of e-Collaboration, Vol. 6, 2010, No. 1, pp. 1–14.

[4] Fontaine, M. A.—Parise, S.—Miller, D.: Collaborative Environments: An Effective Tool for Transforming Business Processes. Business Journal – Improving the Practices of Management, May/June 2004, pp. 1–10.

[5] Laso Ballesteros, I.: Research Perspectives on Collaborative Infrastructures for Collaborative Work Environments. In WETICE: 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2006.

[6] Erl, T.: Service-Oriented Architecture. Concepts, Technology, and Design. PrenticeHall, Crawfordsville 2005.

[7] Brickley, D.—Miller, L.: FOAF Vocabulary Specification 0.97. Namespace document, January 2010.

[8] Breslin, J. G.—Decker, S.—Harth, A.—Bojars, U.: SIOC: An Approach to Connect Web-Based Communities. Int. J. Web Based Communities, Vol. 2, 2006, pp. 133–142.

[9] Stankovic, M.: Modeling Online Presence. In Proceedings of the First Social Data on the Web Workshop, Karlsruhe, Germany October 2008, Volume 405, CEUR Workshop Proceedings.

[10] Appelt, W.: WWW Based Collaboration With the BSCW System. In SOFSEM99: Theory and Practice of Informatics, LNCS Volume 1725, Springer 2010, p. 762.

[11] Shariff, M.: Alfresco. Enterprise Content Management Implementation. Packt Publishing 2009.

[12] Kumar, P.: Documentum 6.5 Content Management Foundations. Packt Publishing 2010.

[13] Microsoft Corporation EMC Corporation, IBM Corporation. Content Management Interoperability Services (CMIS). Standard, OASIS, 2010.

[14] Licari, J.: Best Practices for Instant Messaging in Business. Network Securirity, 2005, No. 5, pp. 4–7.

[15] Google Inc. Deloitte Working Together As One With Google Wave. Case study, Google Inc, 2010.

[16] Veiel, D.—Haake, J. M.—Lukosch, S.: Extending a Shared Workspace Environment With Context-Based Adaptations. In Groupware: Design, Implementation, and Use, 15th International Workshop, CRIWG 2009, pp. 174–181, September 2009.

[17] Haake, J. M.—Husseing, T.—Joop, B.—Lukosch, S.—Veiel, D.—Ziegler,J.: Context Modelling for Adaptive Collaboration. Technical report, Universität Duisburg Essen, 2009.

[18] Schummer, T.—Lukosh, S.: Pattern for Computer-Mediated Interaction. Chapter: From Patterns to a Pattern-Oriented Development Process. pp. 21–64, John Wiley and Sons 2007.

[19] Peristeras, V.—Martínez Carreras, M. A.–Gómez Skarmeta, A. F.—Prinz, W.—Nasirifard, P.: Towards a Reference Architecture for Collaborative Work Environments. International Journal of E-Collaboration, Vol. 6, 2010, No. 1, pp. 14–32.

[20] Truong, H.—Dustdard, S.—Baggio, D.—Dorn, C.—Giuliani, G.—Gombotz, R.—Hong, Y.—Kendal, P.—Melchiorre, C.—Moretzky, S.—Peray, S.—Polleres, A.—Reiff-Marganiec, S.—Schall, D.—Stringa, S.—Tilly, M.—Yu, H.: inContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms. In The 2008 International Symposium on Applications and the Internet (SAINT 2008), IEEE Computer Society, Turku 2008.

[21] W3C OWL Working Group. OWL 2 Web Ontology Language: Document overview. W3C recommendation, W3C, October 2009.

[22] Horrocks, I.—Patel-Schneider, P. F.—Boley, H.—Tabet, S.—Grosof, B.—Dean, M.: SWRL: A Semantic Web Rule Language combining OWL and RuleML. Technical report, W3C, 2004.

[23] Muñoz Ortega, A.—Alcaraz Calero, J. M.—Botía Blaya, J. A.–Martínez Pérez, G.—Garcia Clemente, F. J.: Knowledge authoring with ORE: Testing, Debugging and Validating Knowledge Rules in a Semantic Web Framework. Journal of Universal Computer Science, Vol. 16, 2010, No. 9, pp. 1234–1266.

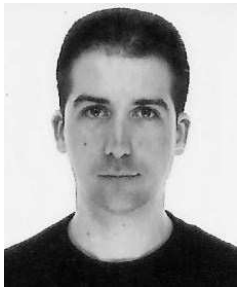
[24] Sirin, E.—Parsia, B.—Cuenca Grau, B.—Kalyanpur, A.—Katz, Y.: PELLET: A Practical OWL-DL Reasoner. Journal of Web Semantics, Vol. 5, 2007, No. 2, p. 10.

[25] Carroll, J. J.—Dickinson, I.—Dollin, C.—Reynolds, D.—Seaborne, A.—Wilkinson, K.: JENA: Implementing the Semantic Web Recommendations. In Proceedings of the 13th International World Wide Web conference, ACM Press 2004, pp. 74–83.

[26] Bozsak, E.—Ehrig, M.—Handschuh, S.—Hotho, A.—Maedche, A.—Motik, B.—Oberle, D.—Schmitz, C.—Staab, S.—Stojanovic, L.—Stojanovic, N.—Studer, R.—Stumme, G.—Sure, Y.—Tane, J.—Volz, R.—Zacharias, V.: KAON – Towards a Large Scale Semantic Web. In EC-Web, 2002, pp. 304–313.

[27] Tsarkov, D.—Horrocks, I.: Automated Reasoning. Volume 4130 of Springer Lecture Notes in Computer Science, Chapter FaCT++ Description Logic Reasoner: System Description. Springer Berlin/Heidelberg 2006, pp. 292–297.

M. Antonia Martínez-Carreras has a Ph. D. in computer science from the University of Murcia. She is an Assistant Professor for Collaborative Environments at the same university. Her research area is devoted to cooperative systems, CSCW, groupware, CSCL and interoperability. She has been working in the Department of Information and Communication Engineering since 1999 until now, where she was involved in several research projects funded by the European IST such as ITCOLE, COLAB and ECOSPACE.

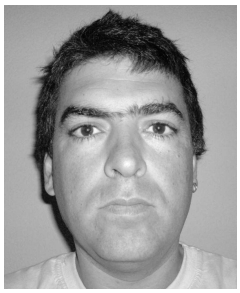


Juan M. Marín Pérez is research fellow in the Department of Information and Communications Engineering of the University of Murcia. He received Engineering and MSc degrees in computer science from the University of Murcia. He has been working in several European projects while doing his Ph. D. His research interests include security and management of cloud computing and distributed systems.

Jorge Bernal Bernabé received the B. Sc. in computer engineering and the M. Sc. in computer science from the University of Murcia. Currently, he is a research fellow in the Department of Information and Communications Engineering of the University of Murcia. He has been working in several European projects while doing his Ph. D. His scientific activity is mainly devoted to security and management of cloud computing and distributed systems.

José M. Alcaraz Calero received his Ph. D. in computer science from the University of Murcia. He has been working at University of Murcia since 2004 in several European and international projects. His research interests include cloud computing, security, semantics and distributed systems. Currently he is research fellow at Hewlett Packard Laboratories.

Gregorio Martínez Pérez is an Associate Professor in the Department of Information and Communications Engineering of the University of Murcia. His research interests include security and management of distributed communication networks. He received an M. Sc. and Ph. D. in computer science from the University of Murcia.



Antonio F. Gómez Skarmeta received the M. Sc. degree in computer science from the University of Granada and B. Sc. (Hons.) and the Ph. D. degree in computer science from the University of Murcia. He is a Full Professor in the same Department and University. He has worked on different research projects at regional, national and especially at the European level in areas related to advanced services like multicast, multihoming, security and adaptive multimedia applications in IP and NGN networks.