Virtual Knowledge Communities for Semantic Agents

Julien Subercaze, Pierre Maret

Virtual Knowledge Communities for Semantic Agents

Julien Subercaze
Laboratoire Hubert Curien
Université de Lyon
18 rue du professeur Benoit Lauras.
F-42000 Saint-Etienne
juLink.subercaze@univ-st-etienne.fr

Pierre Maret
Laboratoire Hubert Curien
Université de Lyon
18 rue du professeur Benoit Lauras.
F-42000 Saint-Etienne
pierre.maret@univ-st-etienne.fr

ABSTRACT

Virtual Knowledge Communities are a well suited paradigm for decentralized knowldege exchanges and they have been applied in several domains. In this paper we investigate the implementation of virtual knowledge communities with semantic agents. Using the SAM (Semantic Agent Modeling) approach, we show that agents can exchange community related concepts (in OWL) and behavior (in SWRL). Agents can then learn and adapt new community-related behavior, which is usefull when changing the role or entering into a new environment. For this purpose, we formalize Virtual Knowledge Communities in a set-theoretic way and we implement this formalization in an OWL ontology. Some examples of community representation using our formalization are presented in this paper.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]; I.2.11 [**Artificial Intelligence**]: Multiagent systems

General Terms

Semantic Agent, Virtual Knowledge Communities

Keywords

Semantic Agent, Virtual Knowledge Communities, Set-theoretic formalization, Ontology, OWL, SWRL

1. INTRODUCTION

Agent mediated knowledge management (AMKM), a term coined for the first AMKM workshop in 2003 [17] (followed by two editions [18, 16]) is a specialized research domain of distributed knowledge management. The main idea of AMKM is to take into account the decentralized nature of knowledge and its social aspects inside organizations. Nowadays, the rise of semantic technologies such as RDF,OWL and SWRL have provided a powerful frame for knowledge representation and processing, due to the expressivity of these formalisms and their underlying logic formalisms [5]. A first step in the integration of semantic technologies in the domain of agent-mediated knowledge management has been made in using the knowledge representation formalism. Thus, semantic technologies have been introduced to replace former knowledge representation in mainstream agent platforms like JADE [1] and in agent programming languages like the OWL extension for JASON [2]. A second step has consisted in taking advantage of the logical power of the semantic languages to replace the usual Prolog-like languages used in agent programming. By doing so, agent knowledge and agent programming can both be tackled within a unique paradigm i.e. the semantic technologies. This is known as introducing *de facto* the agent's reflexive abilities: the ability to change its behavior depending on its beliefs. Agents using semantic technologies in both agent knowledge representation and internal programming have been called semantic agents. Several architectures such as Nuin [6], S-APL [9], JASDL [11] and SAM [12] have been proposed.

However, while the semantic models have been introduced in these agent-oriented architectures, the duality between agent knowledge and agent behavior remains. In Nuin and in S-APL, the behaviour and the knowledge base are expressed in differents languages or paradigms. Thus, knowledge (from the knowledge base) has to be converted in order to be used by the behaviour. In Nuin and in S-APL, the behaviour and the knowledge base are expressed in different languages or paradigms. Thus, knowledge (from the knowledge base) has to be converted in order to be used by the behaviour. For Nuin and S-APL, the conversion (from RDF or from OWL) into the agent programming language implies a loss of expressivity since the logics of knowledge bases are more expressive than the ones used in the agent programming.

The aim of this paper is to propose an innovative approach that implements the combination of knowledge and behavior when considering exchanges between agents. Making agents acquire knowledge is not an innovative effort. Our proposal is to describe a model and to implement a system where agents can learn and adapt new community-related behavior (SWRL rules). This is useful when the agent changes role in a community or enters into a new environment. We will consider an existing model for agent exchanges (namely VKC [10]) and will propose an ontology that describes and operationalizes agent knowledge and behavior in this area. This is implemented in the frame of the SAM (Semantic Agent Model) architecture.

In the next section, we will present the model of VKC and

WIMS '11, May 25-27, 2011 Sogndal, Norway

the SAM architecture. In section 3 an innovative formalization of the VKC approach using a set-theoretic model will be proposed. Based on this formalization, we will then define the OWL-based ontology that describes and operationalizes knowledge and behavior for semantic agent exchanges. Related works will be briefly described in section 4 and some concluding remarks will be given in section 5.

2. BACKGROUND

Two main background concepts are concerned with the present paper: Virtual Communities and the Semantic Agent Model. They are here presented for a good understanding of the proposed set-theoretic model and the OWL-based ontology.

2.1 Virtual Knowledge Communities

Research in the field of corporate knowledge management lead by Calmet and Maret introduced a distributed approach for corporate knowledge management based on the agent paradigm. The concept of Virtual Knowledge Communities (denoted VKC) was then introduced in 2004 [10]. Since then, it has been used in several application domains.

Virtual Knowledge Communities are introduced as a means of knowledge dissemination within a society of agents. Each agent owns an internal goal to achieve which, directly or indirectly, may lead it to distribute or to acquire knowledge. Communities are the mechanism used by the agents for these knowledge exchanges. Thus communities increase the efficiency of knowledge dissemination within the society of agents. Through the dynamic processes of VKCs, agents can create, join, feed, use or leave communities, and so they can acquire new knowledge from other agents and infer new knowledge through their cognitive component.

A virtual knowledge community consists of a topic, community members and a policy for message exchanges. Members exchange messages related to the topic of the community, following the community policy (rule-based policy). Messages are exchanged through a shared board system that allows to post new messages and to access previously posted ones. This board system is called the community buffer. In the implementation provided by Hammond [8], agent knowledge is represented with ontologies (i.e. concepts, links and instances). Pieces of knowledge that can be exchanged within agents are subsets of one agent ontology. A community discovery service called the "community of communities" is also provided.

Among the functionalities related to community management, one can cite the evolution of a community topic and the dissolution of a community. Topic evolution is required because communities are created dynamically and they evolve during their lifetime (i.e. participants, messages and topics can all change). The mechanism to change the community topic is provided, it is controlled by the community owner (a member with the appropriate policy) and it consists of an internal updating and sending of a message addressed to the community of communities for updating this kind of yellow pages. Similarly, the dissolution of the community can be made on the initiative of the community leader: a message is sent to the community of communities to remove the existing entry.

The ability of agents to interpret and exchange the acquired knowledge is considered as intrinsic knowledge. This ability can be different for each and every agent. We argue that this design choice takes into account the decentralized nature of knowledge and its consequences in agent-mediated knowledge management.

Community regulation is a human social necessity that is also required in virtual communities. Regulation consists of rules that apply within the community regarding its use, security and confidentiality. To handle this regulation, each agent creates a community and specifies the policy which applies to it. The policy specifies, for instance, the way agents join, leave, share messages and access messages. Also, policies are introduced to handle security issues that can be encountered with malicious agents in an open agent society. For instance, access control to the board system of the community or encryption of messages can be specified.

The VKC paradigm has been succesfully applied to several application domains, including context-aware applications [14], mobile applications [15] and E-Health [7].

2.2 Semantic Agent Model

The Semantic Agent Model [12, 13] (noted SAM) is an architecture of semantic agent that aims at representing both static knowledge and agent behaviour using semantic models. The static knowledge is represented in SAM using ontologies expressed in OWL, which is today the most convenient modelling technique. Regarding agent behavior, while other existing semantic agent models do not support any semantic programming language, the SAM includes a standardized semantic approach, i.e. the behavior is expressed in terms of rules and it is based upon the SWRL (Semantic Web Rule Language). Doing so, the static knowledge and the behavior are expressed in a unique paradigm (semantic modeling), and this makes them fully compatible and avoids translations from one paradigm to another.

The behavior of an agent is implemented as sets of specialized SWRL rules. The antecedent of the rules consists of some conditions concerning the environment and the internal state of the agent. The consequent of the rules contains a sequence of actions that will cause the agent to act on its environment and/or to modify its internal state. Actions are of two types: SWRL built-ins and lower level actions. Lower level actions emcompass actions that are required to support an agent execution (message sending, file access), actions for knowledge manipulation that cannot be expressed with SWRL rules (such as knowledge removal), and agent framework interactions. The execution of the behavior, as well as the the maintainance of the knowledge base's coherence and the inference of new knowledge, is supported by a reasoner (Pellet).

The added value of SAM is that it increases the semantic interoperability of agents: given a limited number of low-level actions that must be available on each agent, any behavior expressed in terms of SWRL rules can be exchanged between agents. This means that agents can now acquire a behavior, and that they can execute this new behavior taking into account their own static knowledge.

As SWRL doesn't support rule ordering, a slightly modified version of the Extended Finite State Machine (EFSM) model [4] has been used. This version guarantees the execution of only one rule at a time. In EFSM, transitions between states are expressed using *if statements*, and they

are linked to some operations. A transition is fired if trigger conditions are valid. Then, the state machine is brought from the current state to the next state, and the set of specified operations is performed.

The following rule is an example of a SAM SWRL rule. It states that if the current agent is in state A (lines 1 and 2 in the rule) and the message received (3) has been sent by `Bob` (4) and this message is an answer (5), then the next state of the agent (8) will receive the value held by y (or `sEnd` which is the final state), a stored list of actions `ActionSequence` will be performed (9) till the end (10), and the content of the message (hold by w) will be added in the knowledge base (11, 12).

$$CurrentState(?x) \quad (1)$$
$$\wedge\, StateValue(x, A) \quad (2)$$
$$\wedge hasReceived(?z) \quad (3)$$
$$\wedge hasSender(z, Bob) \quad (4)$$
$$\wedge hasPerformative(z, Answer) \quad (5)$$
$$\wedge\, NextState(?y) \quad (6)$$
$$\wedge hasContent(z, ?w) \quad (7)$$
$$\Rightarrow\, StateValue(y, sEnd) \quad (8)$$
$$\wedge hasContent(ActionSequence, AddInstance) \quad (9)$$
$$\wedge hasNext(ActionSequence, endOfList) \quad (10)$$
$$\wedge hasParameterName(AddInstance, name) \quad (11)$$
$$\wedge hasParameterValue(name, w) \quad (12)$$

The main benefit of the approach is that knowledge and behaviour are represented using compatible semantic languages. Consequently, an agent's behaviour is also a *piece of knowledge*, operational knowledge, upon which the agent can reason and which can be sent or received to/from other agents.

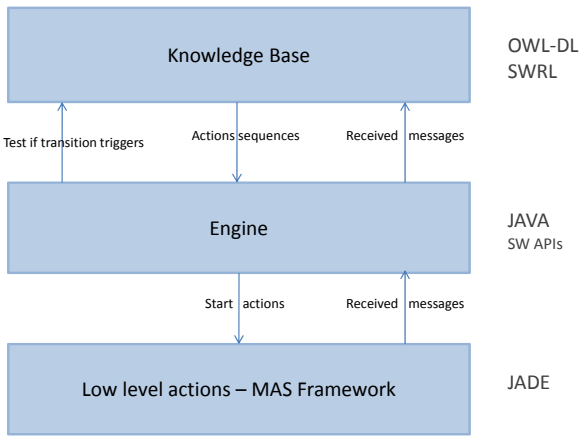

Figure 1: The SAM [12] architecture

3. FORMALIZATION OF VIRTUAL KNOWLEDGE COMMUNITIES

While Virtual Knowledge Communities have been implemented in several application domains, no formalization of the approach or the model has been given. Our aim is to implement VKC in semantic agents. So, the design of a VKC domain ontology is required. For this purpose, we will follow a two-step formalization. In a first step we will formalize the VKC in a set-theoretic way, which is the common abstraction paradigm. In a second step we will adapt this formalization as an OWL ontology, to make it compatible with SAM agents.

3.1 A set-theoretic formalization

We define several sets and relations to formalize VKCs. The first set is M the set of agents that belong to the community. Communities bring together agents around topics (one or several). The set of these elements is denoted S. As previously presented, agents do not have identical roles in communities: they can be a leader, a member or have some other predefined role. We denote R the set of the roles available in the community. A leader can shut the community down, whereas a member can only join or leave it. These actions are grouped in the set of actions called A. To link agents to roles and roles to actions, we define the relations r_r and r_p.

The most significant set in this formalization is P, the set of protocols. These protocols define the communication rules and acts that define interactions between agents within a community. Protocols can be designed in terms of SAM semantic rules, which then make them into executable and exchangeable knowledge for SAM agents. Doing so, agents can acquire and execute VKC protocols and thus participate in communities.

The underlying idea of this approach is to build an operational representation of VKC. In the end, agents should be able to exchange knowledge of communities that can be executed by other agents. These protocols belong to the set P that describes implementation of actions that are associated to roles by the relations r_p.

We propose the following formalization for VKC: a community c belongs to the set of communities C. A community is a tuple: $c \in C = \{M, R, A, P, S, r_r, r_a, r_p\}$ where M, R, A, P, S are sets and r_r, r_a, r_p are relations between these sets.

Sets.

- M is a set of agents belonging to a community. A community owns at least one agent which is the creator of the community. $|M| \geq 1$.
- R is a set of possible roles within the community.
- A is a set of actions that are possible in the community. Classical actions are *join*, *leave*, *post*, ... To be meaningful, A must be non empty, otherwise no action is possible and the creator is the only member.
- P is a set of protocols. A protocol describes the rules for exchanges within the community. A protocol is related to an action. A community protocol is bound to specify the orchestration of actions for a community.

For example, a protocol can dictate that an agent cannot (or can) post a message without being a member of the community. In this example, the community protocol requires a JOIN before a POST. The cardinality of P is greater or equal to 1 (which is a consequence of A being non empty).

- S is the set of topics of the community.

Relations.

- $r_r \subseteq M \times R$ is a relation between the set of agents M and the sets of roles R. This relation describes the different roles in a community an agent can be associated to. Each agent has at least one role. However some roles in a community can be unoccupied. Thus, r_r is left-total [1].
- $r_a \subseteq R \times A$ is a relation between the role set R and the action set A. Each role is associated with the sets of actions permitted for this role in the community. As each role has at least one action related, r_a is left-total.
- $r_p \subseteq R \times A \times P$ is the relation that associates each couple (role, action) to a protocol. For example the action $JOIN$ is different if the member is a user or the host of the community. In the first case, the agent will send the request, in the second one, the host will process the request.

3.1.1 Zero community

The zero community $c_0 \in C$ is the minimal community in which knowledge exchange can happen. The term *minimal* is here to be understood in the sense of a minimal number of actions and roles. One may intuitively think that the minimal community is made of the 3 actions `join, sendMessage, leave`. However, it is possible to define a meaningful zero community with only two actions `sendMessage, leave`. This community is then similar to a mailing-list where the action of sending a message automatically registers the agent into the community. For the agent, leaving the community consists of sending a `leave` message to the community leader agent which removes it from the list of members. The figure 2 depicts this example. Bob is the creator and leader of the community, Chuck and Dave are members. Alice sends a message to Bob following the protocol specified by `sendMessage` for agents willing to join the community. Bob follows the protocol associated with `sendMessage` for the creator and thus broadcasts Alice's message to the community members. Finally, Alice is added to the list of members of the community.

Using the previous formalization, the zero community can be written as follows. Notice that '+' indicates an action initiated by an agent, and '-' indicates an action where the agent receives the action messsage.

- $M = \{Alice, Bob, Chuck, Dave\}$
- $R = \{leader, member\}$
- $A = \{sendMessage, leave\}$

[1] A relation $R \subseteq X \times Y$ is left-total if and only if for all x in X there is one y in Y such as $(x, y) \in R$.

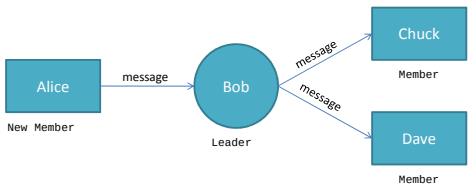

Figure 2: *sendMessage* execution in the community c_0

- $P = \{sendMessage+, leave+, sendMessage-, leave-\}$
- $|S| \geq 1$, the topic
- Roles Agent associations:
  - $(Alice, member) \in r_c$
  - $(Bob, leader) \in r_c$
  - $(Chuck, member) \in r_c$
  - $(Dave, member) \in r_c$
- Roles Actions associations:
  - $(leader, leave) \in r_c$
  - $(leader, sendMessage) \in r_c$
  - $(member, leave) \in r_c$
  - $(member, sendMessage) \in r_c$
- Roles, actions and protocols:
  - $(leader, sendMessage, sendMessage-) \in r_p$
  - $(leader, leave, leave+) \in r_p$
  - $(leader, leave, leave-) \in r_p$
  - $(member, sendMessage, sendMessage+) \in r_p$
  - $(member, leave, leave+) \in r_p$

The association between the role leader and the leave action leads to both + and - protocols. When the leader receives a `leave` request from an agent (i.e. leave-), it removes the agent from the list. In the case where the leader wants to leave the community (leave+), the community is deleted.

3.1.2 Protocols

Protocols specify communication rules between agents. There is a protocol for each action in the community. The communication rules described specify the content and the order of exchanged messages.

3.1.3 View of a community

Depending on their function and history, agents may have different information about a community they are members of. This is what is called the view of a community, and it is held by an agent. Continuing the zero-community scenario, let us take another example: *Alice* has just joined a community initiated by *Bob*. Agents *Chuck* and *Dave* already belong to this community. As *Alice* has just joined, she doesn't know anything about members of this community, the previous content exchanged or the topic evolutions, etc. However, *Bob*, who initiated the community, knows every member and it has been able to hold the history of all

exchanged messages (or it has acquired the content of the messages in its knowledge base).

Let us recall that the SAM architecture handles static knowledge and behavior at the same level (and in the same semantic paradigm). Thus, the agent's view of a commmunity also encompasses the protocols an agent is able to execute related to communities. Agents can then hold different protocols relative to their respective roles in given communities. Such a mechanism is useful for security purposes to limit one's actions in a given environment. Also, since SAM makes protocols exchangeable, an agent (when changing the role) can acquire a new behavior for the related community, which changes its *executable* view of this community.

3.1.4 VKC efficiency measurement

VKCs are a means of sharing knowledge between agents. Since protocols can be exchanged, it is relevant to consider and to compare the protocol's efficiency for knowledge dissemination. We propose to measure this sharing efficiency within communities and we introduce first some basic measurement tools.

- Let A_t be the set of agents in the community at time t.
- Let $K_{T,i,t}$ be the set of knowledge received by agent i on the topic T at time t.
- Let $f_i K \rightarrow [0,1]$ be the function from the set of knowledge K to the interval $[0,1]$. This function represents the interest of the agent i for a piece of knowledge. Zero means no interest and 1 means a maximum of interest.
- Let $Kg_{T,t}$ be the set of knowledge exchanged on the topic T at the time t.

We define the efficiency of the knowledge dissemination with the formula (adapted from [3]):

$$EKD_T(t) = \frac{\sum_{i \in A_t} \frac{|f(K_{T,i,t}|}{|Kg_{T,t}|}}{|A_t|} \quad (13)$$

By definition, EKD ranges into the interval $[0,1]$. EKD reaches its maximum (i.e. $= 1$) when all agents have received only knowledge of interest. In the case where no agent has received any knowledge worthy of interest, the function is equal to zero. The EKD measure represents the efficiency along the time. The normalized integral over time $\frac{\int_{t1}^{t2} EKD_T(dt)}{t2-t1}$ indicates the efficiency average between time $t1$ and $t2$. The derivative $\frac{dEDK_T(t)}{dt}$ gives the instantaneous variation of the efficiency. These formulae give a frame for community benchmarking. Some benchmark results are provided in a previous study [3].

3.2 Ontology

We have presented a set-theoretic formalization of virtual knowledge communities. We have given a specific definition of community protocols, based on rules that define agent behaviour and applicable in the SAM architecture. Now, our formalization will be made available for semantic agents in defining an OWL-based ontology of virtual knowledge communities that will be integrated into each SAM agent's knowledge base.

For the design of the ontology, we translate sets and relations from the previous formalization into ontology concepts, i.e. classes and relations in OWL. This translation goes straight-forward, except for the ternary relation r_p that cannot be directly expressed in OWL. Indeed OWL only supports binary relations, which then restricts our implementation. This restriction can be circumvented thanks to one of the three design methods proposed by the W3C [2]. We decided to use the solution which introduces an intermediate class (denoted AP, for ActionProtocol). Figures 3 and 4 depicts this translation from a ternary relation into OWL double relation.

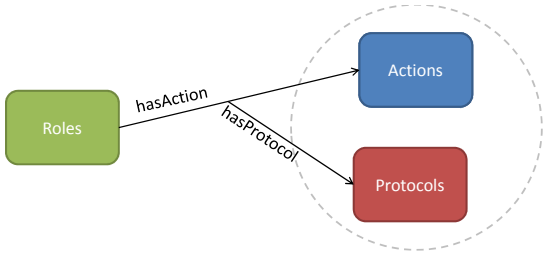

Figure 3: Ternary relation. Not native in OWL

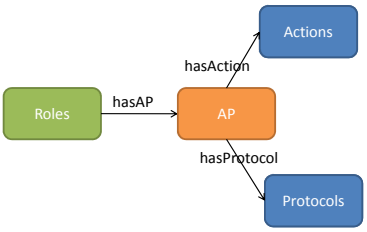

Figure 4: Introduction of the class AP to represent the r_p relation

Classes.

The classes of the ontology are mostly the same as the ones defined in the set-theoretic formalization. `Agent` class represents an agent. `LeaderAgent` is a subclass of `Agent` and represents agents having the leader role in the community. The class `Role` describes the different roles available in the community. These roles are linked to actions through the classes `CommunityAction` and `protocol`. The class `AP` (for ActionProtocol) is the artefact used to describes the ternary relation between a role, an action and a protocol. Finally the class `Topic` describes the topic of the community. The classes are listed in the table 1:

Relations.

The relations in the ontology can be deduced from the ones which have been formally introduced. In OWL, relations have a name, a domain (the class of subject individual) and a range (the class of the object individual). The relation `hasTopic` associates a community to a topic. A community

[2] `http://www.w3.org/TR/swbp-n-aryRelations/`

Name	Subclass	Description
AP	Thing	Action/Protocol
Agent	Thing	Agent
Community	Thing	VKC
CommunityAction	Thing	Action of a community
LeaderAgent	Agent	Leader agent
LeaderRole	Roles	Leader role
Protocol	Thing	Communication protocol
Role	Thing	Roles
Topic	Thing	Topic

Table 1: List of the concepts of the ontology

is linked to its members through the inverse relations `memberOf` and `hasMember`. Roles are linked to both communities (available roles) and agents (roles in the community). The relation between agent and roles is modeled by `memberHasRole` and between role and the community by the relation `hasRole` and its inverse relation is `roleOf`. Roles are also linked to actions and protocols through the relations `hasAP`, `hasAction` and `hasProtocol` that represent the ternary relation. Relations are summarized in the table 2.

3.2.1 View of communities

The community ontology describes the concepts and their relations. The agent's view (composed of static knowledge and behavior) of a community is populated with instances of those concepts. Returning to our example -the zero-community-, agents *Alice*, *Bob* and *Chuck* have the following views:

Alice
- Topic: topic of the community
- Roles: member, leader
- Members: *Alice*,*Bob*
- MemberHasRole: *Alice* → *member*
  Bob → *leader*
- Actions: *sendMessage*,*leave*
- Protocol: *sendMessage+*,*leave+*

Bob
- Topic: topic of the community
- Roles: member, leader
- Members: *Alice*,*Bob*,*Chuck*,*Dave*,
- MemberHasRole: *Bob* → *leader*
  Alice → *member*
  Chuck → *member*
  Dave → *member*
- Actions: *sendMessage*,*leave*
- Protocol: *sendMessage+*,*sendMessage-*,*leave+*,*leave-*

Chuck
- Topic: topic of the community
- Roles: member, leader
- Members: *Alice*,*Bob*,*Chuck*,*Dave*
- MemberHasRole: *Alice* → *membre*
  Bob → *createur*
  Chuck → *member*
  Dave → *member*
- Actions: *sendMessage*,*leave*
- Protocol: *sendMessage+*,*leave+*

It appears that the leader of the community is the only one with a complete view of the community. This is a characteristic of a centralized community like the zero-community. Notice that one could implement several types of communities (decentralized, individualistic, social, etc.) and this would influence the content of the view of the community of agents. Also, a community leader owns the protocols related to the community, but it may not necessarily know the complete set of agent members of the community. This concept of views of community appears as a keystone to maintain the decentralized approach strongly claimed in the virtual knowledge communities approach.

3.2.2 Application: Yahoo! Groups

In order to better illustrate our proposal, we will describe it in the frame of a real-life example: the Yahoo! Groups [3]. Yahoo! Groups are discussion-based communities, created in 1998. This service belongs to the top 5 Yahoo! services in terms of frequency of use. Yahoo! Groups offers a complete set of features to support community interactions. The following features are provided to users:

- Messages: Send an email to the group (and the possibility to delete a previous message)
- Photo album
- File storage
- Links directory: contains a link and a description for each entry
- Polls
- Database: 10 tables allowed
- List of members
- Calendar
- Advertising: generates HTML code to promote the group

Creators and managers of a group have access to a larger range of features in order to administer the community:

- Invitation: invite new members
- Members management: validate and delete accounts
- Messages management: validation and deletion of messages

[3] `http://fr.groups.yahoo.com`

Name	Domain	Range	Description
hasTopic	Community	Topic	associate a community to a topic
hasAP	Roles	AP	associates actions and protocol to a role
hasAction	AP	CommunityAction	associates an AP to an action
hasProtocol	AP	Protocols	associates an AP to a protocol
hasMember	Community	Agent	defines members of the community
memberOf	Agent	Community	inverse of hasMember
hasRole	Community	Role	associates roles to community
roleOf	Role	Community	inverse of hasRole
memberHasRole	Agent	Roles	associates a role to an agent

Table 2: List of the relations of the ontology

- Options: change the welcome message. ...

Our ontology can be used to represent Yahoo! Groups as virtual knowledge communities. We first identify the different roles: regular member, manager and creator. In addition to these three roles, the role of Yahoo! must be added. It serves as a service provider for community hosting and community searching. The number of community creators and community hosts is limited to one by Yahoo! The number of members and of managers is not limited.

The features previously described lead us to describe 29 actions for members, managers and creators:

- Messages: {*SendMessage, DeleteMessage*}
- Photo album: {*CreateAlbum, DeleteAlbum, AddPicture, RemovePicture*}
- File Storage: {*CreateFolder, DeleteFolder, AddFile, DeleteFile*}
- Links directory: {*CreateFolderDirectory, DeleteFolderDirectory, AddLink, DeleteLink, EditerLink*}
- Polls: {*CreatePoll, DeletePoll, AddRéponse, FinVote*}
- Databases: {*CreateTable, DeleteTable, AddValue, DeleteValue*}
- Members management: {*AfficherListe, ViewProfil*}
- Calendar: {*AddEvent, DeleteEvent, EditerEvent*}
- Advertising: {*CreateBoîte*}

6 additionnal actions restricted to manager and creator:

- Invitation: {*InviteMember*}
- Members management: {*ValidateMember, DeleteMember*}
- Messages management: {*ValidateMessage, DeleteMessage*}
- Options: {*SetWelcomeMessage*}

In addition, the action *DeleteCommunity* that definitely shut the community down is restricted to the creator and to the host. The Yahoo! Groups ontology contains 29+6+1 =36 actions. There are 101 relations roles/actions: $29 \times |\{Member, creator, manager\}| + 6 \times |\{manager, creator\}| + 1 \times |\{creator, host\}|$. Each action has protocols for both sender and receiver. In this case, the host (i.e. Yahoo! Groups) is the only receiver of the actions.

This example has shown the use of VKC ontology in a real-world situation for a well-known and widely used community network. The whole implementation of this ontology in OWL+SWRL would take some time, however our approach would then make it technically possible for SAM agents to acquire some static knowledge as well as some behavior in order to participate in Yahoo! Groups, even if they were not initially programmed for this purpose.

4. RELATED WORK

To our knowledge, the integration of Virtual Knowledge Communities with Semantic Agents has not been addressed until now. The presented ontology is specifically designed for the Semantic Agent Model. The concept of protocol has been introduced in the ontology to allow SAM agents to execute VKCs. However the set-theoretic formalization of the VKCs and the principal components of the VKC ontology (except the specific design of protocols, dedicated to the Semantic Agent Model) are well suited to be used with other Semantic Agent architectures such as S-APL [9] or NUIN [6]. The ability of agents to exchange community related concepts and behavior decribed with semantic web models (OWL+SWRL) is, to our knowledge, an original proposal.

The SIOC (Semantically-Interlinked Online Communities) ontology is an attempt to link online community sites. Also it uses Semantic Web technologies to describe the structural and content information of communities, and to find related information and new connections between content items and other community items. SIOC is based around the use of machine-readable information which is found on the online community sites [4]. Whereas our formalization addresses the way a community works, SIOC provides mechanisms to interlink static data provided by different communities. The VKC formalization and the SIOC ontology can thus be seen as complementary projects of semantic communities.

5. CONCLUSION

In this paper we have presented an operational ontology for integrating Virtual Knowledge Communities into the world of semantic agents. Using the specificity of the SAM architecture that allows SWRL agent programming, we have formalized Virtual Knowledge Communities as an ontology that contains not only a description of VKCs, but also their related actions to be executed. The link between

[4] `http://rdfs.org/sioc/spec/`

the descriptive part of the ontology and the implementation is made through the concept of protocols. Protocols are semantic executable code that SAM agents can execute.

In order to formalize VKCs, we first took an abstract approach using a set-theoretic formalization. The process of refining the set-theoretic formalization into an OWL ontology requires few modifications. Since both formalisms are closely coupled (logical foundation of OWL), only the ternary relation requires some modification to be implemented in the OWL ontology. The specificity of OWL, such as inverse relations and restrictions over relations allows to fully describe our formal approach of VKC in an ontology that semantic agents can handle. The decentralized aspect of the community is taken into account with the concept of views of the community. Agents share the domain ontology of VKC, but each agent owns a different view of a community. We illustrated this concept through the example of the minimal commmunity called zero-community. Finally, the Yahoo! Groups helped us to illustrate the generic nature of the ontology. Indeed our ontology is not only designed for VKC but can also be applied to real-world working examples.

Our research has shown that it is now possible for agents to exchange community related concepts and behavior. For that, we use semantic web models (OWL+SWRL). The full convergence of these two models makes the agent more flexible and powerful: it can reason and act with its knowledge without any translation effort (synonym of expressivity loss) and it can learn and adapt to new behavior, which is useful when changing the role and the environment in a community context. Further research will focus on the application of the ontology to different types of communities as well as community benchmarking using our efficiency measure.

6. REFERENCES

[1] F. Bellifemine, A. Poggi, and G. Rimassa. JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, pages 97–108. Citeseer, 1999.

[2] R. Bordini, J. H ”ubner, and R. Vieira. Jason and the Golden Fleece of agent-oriented programming. *Multi-Agent Programming*, pages 3–37, 2005.

[3] Jacques Calmet, Julien Subercaze, and Pierre Maret. Simulation in virtual communities. In *Proceeding of the Social and Organizational Informatics and Cybernetics (SOIC) conference 2006, Orlando, Florida, USA*, pages 238–242, 2006.

[4] K.T. Cheng and AS Krishnakumar. Automatic functional test generation using the extended finite state machine model. In *Proceedings of the 30th international conference on Design automation*, pages 86–91. ACM New York, NY, USA, 1993.

[5] C.V. Damasio, A. Analyti, G. Antoniou, and G. Wagner. Supporting open and closed world reasoning on the web. *Lecture notes in computer science*, 4187:149, 2006.

[6] I. Dickinson and M. Wooldridge. Towards practical reasoning agents for the semantic web. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 827–834. ACM New York, NY, USA, 2003.

[7] Christo El Morr, Julien Subercaze, Marcia Rioux, and Mihaela Dinca-Panaitescu. Virtual Knowledge Network for Human Rights Monitoring. In *Workshop on Web Intelligence and Virtual Enterprises (WIVE'09), 10th IFIP Working Conference on Virtual Entreprises (PRO-VE'09)*, October 2009.

[8] Mark Hammond. Virtual knowledge communities for distributed knowledge management: A multi-agent-based approach using jade. Master's thesis, University of Karlsruhe, 2004.

[9] A. Katasonov and V. Terziyan. Semantic agent programming language (S-APL): A middleware platform for the Semantic web. In *Proc. 2nd IEEE International Conference on Semantic Computing*, pages 504–511, 2008.

[10] Pierre Maret, Mark Hammond, and Jacques Calmet. Virtual knowledge communities for corporate knowledge issues. In Marie Pierre Gleizes, Andrea Omicini, and Franco Zambonelli, editors, *ESAW*, volume 3451 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2004.

[11] Á. Moreira, R. Vieira, R. Bordini, and J. H ”ubner. Agent-oriented programming with underlying ontological reasoning. *Declarative Agent Languages and Technologies III*, pages 155–170, 2006.

[12] Julien Subercaze and Pierre Maret. SAM : Semantic Agent Model for SWRL rule-based agents. In *ICAART 2010 - International Conference on Agents and Artificial Intelligence*, January 2010.

[13] Julien Subercaze and Pierre Maret. *Semantic Agent Systems - Foundations and Applications*, chapter Programming Semantic Agent for Distributed Knowledge Management. Studies in Computational Intelligence. Springer Verlag, November 2010.

[14] Julien Subercaze, Pierre Maret, Ngoc Minh Dang, and

Ken Sasaki. Context-aware applications using personal sensors. In Social-Informatics ICST (Institute for Computer Sciences and Telecommunications Engineering), editors, *BodyNets '07: Proceedings of the ICST 2nd international conference on Body area networks*, pages 1–5, July 2007.

[15] Julien Subercaze, Pravin Pawar, Pierre Maret, and Jacques Calmet. A Service Oriented Framework for Mobile Business Virtual Communities . In Willy Picard Luis M. Camarinha-Matos, editor, *Virtual Enterprises and Collaborative Networks*, pages 493–500. Springer, September 2008.

[16] Jurriaan van Diggelen, Virginia Dignum, Ludger van Elst, and Andreas Abecker, editors. *Proceedings of the AAMAS 2005 Workshop on Agent-mediated Knowledge Management (AMKM-2005)*, 2005.

[17] Ludger van Elst, Virginia Dignum, and Andreas Abecker, editors. *Agent Mediated Knowledge Management, International Symposium AMKM 2003, Stanford, CA, USA, March 24-26, 2003, Revised and Invited Papers*, volume 2926 of *Lecture Notes in Computer Science*. Springer, 2004.

[18] Andreas Abecker Virginia Dignum, Ludger van Elst, editor. *Proceedings of the ECAI-2004 Workshop on Agent-mediated Knowledge Management (AMKM-2004*, 2004.