Knowledge Management in Role Based Agents

Hüseyin Kır, Erdem Eser Ekinci, and Oguz Dikenelli

Ege University, Department of Computer Engineering,
35100 Bornova, Izmir, Turkey
{huseyinkir,erdemeserekinci}@gmail.com, oguz.dikenelli@ege.edu.tr

Abstract. In multi-agent system literature, the role concept is getting increasingly researched to provide an abstraction to scope beliefs, norms, goals of agents and to shape relationships of the agents in the organization. In this research, we propose a knowledgebase architecture to increase applicability of roles in MAS domain by drawing inspiration from the *self* concept in the role theory of sociology. The proposed knowledgebase architecture has granulated structure that is dynamically organized according to the agent's identification in a social environment. Thanks to this dynamic structure, agents are enabled to work on consistent knowledge in spite of inevitable conflicts between roles and the agent. The knowledgebase architecture is also implemented and incorporated into the SEAGENT multi-agent system development framework.

1 Introduction

In multi-agent system research domain, role concept is getting more popular from both methodology and infrastructure perspectives in last years. The main purpose behind using roles in MAS research domain is to provide an abstraction to define beliefs, norms, goals of agents and to shape relationships between the agents in the organization. But, the role concept is not new, and not firstly used by the MAS domain researchers for modeling organizations.

The role concept was introduced within the theater in antique ages to define behaviors and scripts of an actor in scenes of a part. But in the previous century, sociologists converted this simple concept to a deep theory which is comprised of systematic methods of empirical investments to predict activities of individuals, to resolve human social relationships, stratification and interactions by analyzing body of knowledge about human organizations[1]. As a consequence of these researches, the role concept is explicated as designated social part to be played, characteristic behaviors and scripts for social conduct. Albeit at a first glance it appears like that there is not so much change in the general definition of role through the ages, depths of the concept are theorized with many sub-theories (role playing, role taking, role transition, role leaving, role conflict, etc.) which do clarify the problems such as which conditions conduct a role, how an individual plays a role and what the dynamics are behind taking or leaving a role.

When the MAS researchers import the role concept to the multi-agent system theory, they pave the way of transferring all related sub-theories of sociological role theory. Already corresponds of many role sub-theories propounded

H. Aldewereld, V. Dignum, and G. Picard (Eds.): ESAW 2009, LNAI 5881, pp. 181–196, 2009.

by sociologists[2,1] can be seen in the prominent MAS researches[3,4]. But we ascertain that the fact of the self concept's importance in the role theory is disregarded in these researches. In this paper, to increase applicability of roles in MAS domain we propose a knowledgebase architecture by drawing inspiration from the *self* concept in the role theory of sociology.

The self concept, basically, denotes the word "I", which is the knowledge of physical properties, beliefs, impulses, characteristic behaviors and talents, and all other referents to the "I" [2]. But in theory, the concept is not a simple, complete structure but granulated net that is expanded and reorganized during life long evolution of the individual[5,6]. In daily life, each individual participates many organizations such as family, school, and business and gains many identities by taking different roles. In sociology, individual's participation in an organization is technically expressed as expansion of the self with the acceptance of all norms, goals and beliefs, which belong to the enacted role[6]. Additionally, when the individual moves from one participated organizations to another, the granulated form of the self is dynamically reorganized according to the identification of the individual with respect to its cognition about its roles[5,7].

The main contribution of the self concept to the role theory is depicted as clarifying the problem of resolving human behaviors in conflicting positions[7]. In detail, an individual may play more than one role in the same organization or different organizations. But, rules, norms and goals of these roles may not be completely consistent with each other, even if they are in the same organization. Furthermore, an individual can play a role, which is not congruent with her/his self, by violating self's rules and norms until reaching self goals, if there is not another way. In such conflicting positions, by the help of granulated and alterable form of the self concept, sociologists become capable of explaining the meaning of individual's behaviors[5].

Analogously to sociological role theory, in a role-based multi-agent system, agents enact various types of roles due to desire of achieving their congenital goals during their life time. Differently from the human case, in the MAS literature conflicts between agents and roles are formally expressed and highlighted as an undesired situation[3,8]. But as emphasized previously, conflicts between the roles of an organization is inevitable for the reason of looking through the different perspectives to the organization.

In the light of this idea, to make role theory completely applicable on MAS, we propose a knowledgebase architecture, which consists of dynamically arranged knowledge blocks, similar to the granulated and alterable form of self concept developed by sociologists. The proposed knowledgebase architecture has two components; active and passive knowledge. The passive knowledge is used as storage unit for all knowledge blocks of the agent. Besides, the active knowledge is a subset of passive knowledge that is arrangement of knowledge blocks which are the most relevant to the agent's active position in the organization. While consistency of the passive knowledge is not cared, active knowledge is obliged to be consistent to plan deterministically at decision making time. This architecture is implemented and incorporated into the SEAGENT multi-agent development

framework[9]. SEAGENT framework is based on the OWL ontologies in a way that all artifacts of the programming paradigms are stored in the ontologies. So, the knowledgebase is implemented to manage the knowledge handled with these ontologies by the direction of the role theory and the self concept.

The remainder of this paper is structured as follows: In Section 2, we detail our knowledgebase semantics on the agent meta-model and formally declare the effects of the role operations on the knowledge. Section 3 presents the implemented architecture of the proposed knowledge approach which is incorporated into the SEAGENT multi-agent development framework. Section 4 elaborates a case study that emphasize the benefits of the proposed knowledge model. Finally in Section 5, we examine the related works before concluding with Section 6.

2 Role-Based Knowledge Semantics

In the decision making phase, the knowledgebase of the agent has the responsibility of providing information about how the agent reacts the events perceived from the environment, or how it achieves proactively its goals. But beyond providing information about plans or goals, for effective planning, the knowledgebase must also exhibit partial projection of the world encompassing the agent. Through this direction, when the world around the agent is organized with roles, the projection of the world in the knowledgebase takes form according to these roles and their relationships. Since, to handle projection of the world around the agent we define three types of knowledge blocks as granules of our knowledgebase; self knowledge, role knowledge and role-instance knowledge blocks. If it is required to introduce these blocks simply; the role knowledge block involves norms, goals, rules and beliefs specific to a role in an organization and the self knowledge block represents the agent itself in the form of congenital behaviors, norms, goals, rules and beliefs. Differently from the other ones, the role instance knowledge block handles the information of inferred rules, norms and beliefs while the agent is performing under a role. Followings of this section declares the formal semantics of these knowledge blocks.

From the point of knowledge view, the knowledge of a role $r \in R$ is a tuple $K_r = \langle G, P, \sigma_r, \pi_r, \mathbb{R} \rangle$ where, G is the set of goals, P is the set of plans that are used to achieve each role goal, σ_r specifies the facts of the role, π_r specifies the rules of the role and finally $\mathbb{R}$ represents the set of possible relations between roles. To detail semantics of $\mathbb{R}$ we use the model proposed in [10] by Kristensen and Osterbye. They ascertain three types of role relations: specialization, aggregation and dependency. Specialization is supported by extension and used to built up role hierarchies. We represent specialization as $\mathbb{R}_{spec}(r_1, r_2)$, where $r_1, r_2 \epsilon R$, and implies that r_1 extends r_2. Similarly, aggregation is used to construct role compositions and represented as $\mathbb{R}_{aggr}(r_1, \{r_x, r_y, ...\})$, where $r_1, r_x, r_y, ... \epsilon R$, and defines that r_1 is aggregated of given role set. Finally the role dependency means one role is dependent on another role for the realization of its goals[11]. In accordance with the Kristensen's work, we examine the reflections of these inter-role relations on the knowledge block relations.

Another knowledge block in our knowledge model is the self knowledge. The self defines the personal attitudes of an agent $a \in A$. Hence, we represent the self knowledge as a tuple $K_{self} = \langle G, P, \sigma_{self}, \pi_{self}, \Omega \rangle$ where, G represents the set of congenital goals of the agent, P is the set of plans that comprise the own capabilities of the agent, σ_{self} specifies the facts that the agent has about the world and himself, π_{self} specifies the rules that constraints the agent behavior and Ω represents the identity of the self.

In our model the last knowledge block is the role instance knowledge. When an agent enacts a role, it creates a unique entity, called role instance $ri \in Ri$, in the organization. We specify the role instance knowledge as $K_{ri} = \langle id, roleType, enactmentType, \sigma_{ri}, \pi_{ri}, \mathbb{A} \rangle$ where *id* is the unique identifier of the role instance, *roleType* represents the role to which the role instance belongs and *enactmentType* determines the degree of dedication of an agent on a role. The semantics of the enactment is detailed in [11,4] and there are four types of enactment; maximally selfish, selfish, social and maximally social, $\mathcal{T} = \{\tau_{MaxSelfish}, \tau_{Selfish}, \tau_{Social}, \tau_{MaxSocial}\}$. If an agent enacts a role in a selfish form this means the agent attaches higher priority to the self's rules, goals and norms. On the contrary, if the enactment is social, the agent prioritizes its role responsibilities. Finally $\mathbb{A}$ represents the set of role instances that the role instance aggregated of. In this article we use the shorter form as $k_{ri_{id}^{roleType}} \in K_{ri}$, to represent the role instance knowledge. When a role is initiated newly, it has an empty fact(σ_{ri}) and rule(π_{ri}) sets. This knowledge is composed along the life cycle of the agent via learning and inference. Also, these two concepts could be considered as an extension of the role knowledge which empower agent to objectify the role.

Consequently, we represent the agent's whole knowledge as K_a which is union of the self knowledge k_{self}, set of the role knowledge $K_r = \{k_{r_1}, k_{r_2}, ..., k_{r_n}\}$ and set of the role instance knowledge $K_{ri} = \{k_{ri_1}, k_{ri_2}, ..., k_{ri_n}\}$, $k_{self} \cup K_r \cup K_{ri} \subseteq K_a$. But, as emphasized in the introduction, consistency of entire knowledgebase K_a cannot be established due to unavoidable conflicts between self of agents and roles and between enacted roles. The granulated structure of knowledgebase enables an agent to construct the most relevant and consistent subset of the agent knowledge in accordance with its active role. Even the agent enacts various roles that are in conflict, decision making must always placed on a consistent knowledge. To provide the consistency, we separate the knowledgebase into two constituents; active knowledge(K_{active}) and passive knowledge($K_{passive}$). While passive knowledge is equivalent to the entire knowledge of the agent $K_{passive} \subseteq K_a$ and its consistency is not expected, the active knowledge is a sub-set of passive knowledge $K_{active} \subset K_{passive}$ that is arranged from the relevant knowledge blocks into a consistent form. We introduce the semantics that formally define how passive and active knowledge are managed according to the life-cycle of role-based agents in the following section.

Operational Knowledge Semantics

Through the life-cycle of a role-based agent, the agent enacts roles and performs under these roles till reaching its goals and then deacts the roles. Concurrently,

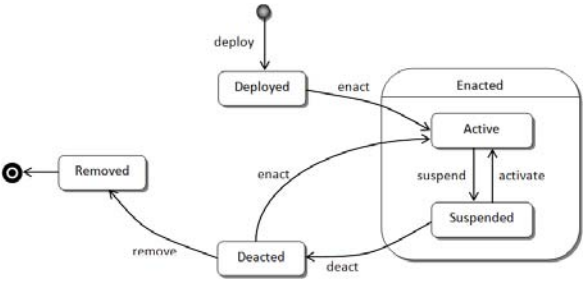

Fig. 1. Role Life Cycle

the knowledgebase also evolves together with its owner agent. In order to detail changes of the knowledgebase during the agent evolution, one must declare operations performed between an agent and a role. These operations are determined informally by Odell et al. in [12]. But Dignum et al. articulate these operations in a formal way[4]. We declare an enhanced version of the role operations in Figure-1 and within the following definitions, we formally declare the effects of these operations on the active and the passive knowledge of the agent.

Definition 1. (Deploy) An agent takes the knowledge of a role by performing deploy operation. This operation makes agent acquainted with the organizational facts, norms and goals from the perspective of a role. By this way, the agent obtains the knowledge about how the role can be performed. After the deployment, the knowledge block of the role is installed into the passive knowledge of the agent. The operation is represented as $O_{deploy}(a_1, r_1) \rightarrow K'_{passive}$ where $r_1 \in R$, $a_1 \in A$ and $K'_{passive}$ is for the evolved agent knowledge after the deployment operation. Let $\langle g_a, p_a, \Sigma_a, \Pi_a)\rangle$ be the agent's passive knowledge before deploy operation and $\langle g_{r_1}, p_{r_1}, \sigma_{r_1}, \pi_{r_1}, \mathbb{R}\rangle$ is the knowledge block of the role r_1, then the deployment operation can be detailed as follows:

$$\frac{K_{passive}{=}\langle g_a{,}p_a{,}\Sigma_a{,}\Pi_a\rangle \quad k_{r_1}{=}\left\langle g_{r_1}{,}p_{r_1}{,}\sigma_{r_1}{,}\pi_{r_1}{,}\mathbb{R}\right\rangle \quad O_{deploy}(a_1{,}r_1)}{K_{passive} \cup k_{r_1} \subseteq K'_{passive} \quad K'_{passive}{=}\left\langle g_a{\cup}g_{r_1}{,}p_a{\cup}p_{r_1}{,}\Sigma_a{\cup}\sigma_{r_1}{,}\Pi_a{\cup}\pi_{r_1}\right\rangle}$$

But, in order to perform deployment operation successfully, some preconditions that comes from the relationships of the deployed role must be satisfied. If the role intended to deploy extends or aggregates from other roles, then the parent or aggregated roles should be deployed as well:

$$O_{deploy}(a_1{,}r_1) \Rightarrow \forall r_x{\in}\, R\ ((\mathbb{R}_{aggr}(r_1{,}r_x) \vee \mathbb{R}_{spec}(r_1{,}r_x)) \wedge O_{deploy}(a_1{,}r_x))$$

Definition 2. (Enact) By performing this operation, an agent gains a new identity and a unique instance of the role is declared to the organization. Instantiation of a role is represented as $O_{enact}(a_1, r_1, \tau){\rightarrow}k'_{self}$, where $r_1 \in R$, $a_1 \in A$, τ specifies the enactment type and k'_{self} specifies the evolved knowledge block of the self after the enactment operation. This operation ingenerates a new

instance of the role r_1 as $ri_x^{r1} \in Ri$ and adds this new identity to the agent's self knowledge block $k_{self} = \langle g_{self}, p_{self}, \sigma_{self}, \pi_{self}, \Omega \rangle$.

$$\frac{k_{self}=\langle g_{self},p_{self},\sigma_{self},\pi_{self},\Omega\rangle \quad O_{enact}(a_1,r_1,\tau)}{k'_{self}=\langle g_{self},p_{self},\sigma_{self},\pi_{self},\Omega'\rangle \quad \Omega \cup \{ri_x^{r1}\} \subseteq \Omega'}$$

Similar to the deployment operation, in order to perform the enactment successfully the agent follows the suit by enacting the aggregated roles:

$$O_{enact}(a_1,r_1) \Rightarrow \forall r_x \in R\ (\mathbb{R}_{aggr}(r_1,r_x) \wedge O_{enact}(a_1,r_x))$$

Besides, enactment does not cause only extending the identities in the self knowledge block. In addition, a role instance knowledge $k_{ri_x^{r1}} = \langle x, r_1, \tau, \sigma_{ri}, \pi_{ri}, \mathbb{A} \rangle$ is also created and inserted into the passive knowledge, $K_{a_1} \cup k_{r_1} \subseteq K'_{a_1}$. But the newly created role instance has an empty fact (σ_{ri}) and rule (π_{ri}) sets. These rules and facts are directly inserted into the role instance knowledge block or generated by inference along the activation and execution of the role instance.

Definition 3. (Activate) An important point is that, an agent may enact various roles to achieve numerous goals concurrently. Hence, a role is not played continuously. Due to the position of the agent in the organization, a role is activated at a time. The activation operation puts forward an identity of an agent to prepare it performing under a role instance. From the point of knowledge view, each activation operation causes a context switch that updates the active knowledge of the agent with respect to the activated role. The active knowledge is constructed from the most relevant knowledge blocks stored in the passive knowledge that literally reflects the activated role instance's conception of the world. Additionally**,** relations between roles and enactment type of the role instance significantly effects the content of the active knowledge.

Before declaring semantics of the active knowledge construction in terms of the enactment type, an important function *Construct Perspective (CP)* and logical operator $\oslash$, which are intensely used in our semantics, are required to be defined. First one is conflict resolver operator "$\oslash$", which designates the maximal subset of the former operand that does not conflict with the latter operand. For instance, let $s_1 = \langle x_0, x_1, x_2, x_3, \neg x_5 \rangle$ and $s_2 = \langle \neg x_1, x_2, x_4, x_5 \rangle$ are set of premises, then $s_1 \oslash s_2 = \langle x_0, x_2, x_3 \rangle$ and $s_2 \oslash s_1 = \langle x_2, x_4 \rangle$. In order to define and eliminate conflicts between entities, we utilize goal and rule conflict definitions made by Dastani et al.[4] and the semantic relations between these entities.

Construct Perspective function, on the other hand, is the root function for arranging the active knowledge from the related knowledge blocks. The function is called when the activation operation is triggered, $O_{activate}(ri_x^{r1}) \rightarrow CP(ri_x^{r1})$ where $ri_x^{r1} \in Ri$. We define $CP(ri_y^{rx})$ function for the role $r_x \in$ R and role instance $ri_y^{rx} \in Ri$, whose type is r_x, as:

$$\frac{CP(ri_y^{rx})}{K_{active} = k_{ri_y^{rx}} \cup CP^{\tau}(r_x) \cup \forall ri_z^{rk} \in \mathbb{A}\ (CP(ri_z^{rk}))}$$

where $k_{ri_y^{rx}} = \langle y, r_x, \tau, \sigma_{ri}, \pi_{ri}, \mathbb{A} \rangle$.

In order to construct the knowledge perspective of a role instance, three groups of knowledge block are added to the active knowledge. The knowledge block of the activated role instance($k_{ri_y^{r_x}}$) is added to the active knowledge without any hesitation of a conflict. Because, when a role is initiated for the first time, it has empty fact (σ_{ri}) and rule (π_{ri}) sets and the efforts on keeping active knowledge consistent prevents expansion of the role instance knowledge in a conflicting way. Also, the active knowledge constructor function recursively adds all aggregated role instances' knowledge, which are defined in $\mathbb{A}$, to the active knowledge. And finally, the knowledge block of the role is added to the active knowledge with the operation $CP^{\tau}(r_x)$. Dignum[11] defines the evaluation of the goals of an agent in accordance with the role enactment type (τ), but goals are not the only determinant on the behavior of an agent. Hence, we extend this definitions in a way that comprises every entity in the knowledge.

The function $CP^{\tau}(r_x)$ process in a different way for each enactment type τ and the active knowledge varies as follows:

Case 1. (*Maximally selfish enactment*) The agent only uses its own goals, and ignores any objectives of the role. But, the role enacting agent cannot try to achieve pure self goals while playing the game by the organizational rules. Hence, the agent should disregard the organizational rules. But, it still has to know at least the shared organizational vocabulary in order to communicate with the other agents. $CP^{\tau}(r_x)$ function for this type of enactment is:

$$CP^{MaxSelfish}(r_x) = k_{self} \cup p_{r_x} \cup (\sigma_{r_x} \oslash \sigma_{self}) \cup \forall\, r_m \in \mathbb{S}\ (CP^{MaxSelfish}(r_m))$$

where $k_{r_x} = \langle g_{r_x}, p_{r_x}, \sigma_{r_x}, \pi_{r_x}, \mathbb{R} \rangle$, $\mathbb{S}$ defines the list of specialized roles that is inferred from the role relations $\mathbb{R}$ and as mentioned before $\sigma_{r_x} \oslash \sigma_{self}$ designates the role facts that does not conflict with facts of the self knowledge block.

Case 2. (*Selfish enactment*) Different from maximal selfish enactment, the enacting agent takes as many of its role's goals and rules as possible into consideration. But self rules still have the priority, so rules of the role that conflict with the self rules are disregarded.

$$CP^{Selfish}(r_x) = k_{self} \cup p_{r_x} \cup (\sigma_{r_x} \oslash \sigma_{self}) \cup (\pi_{r_x} \oslash \pi_{self}) \cup (g_{r_x} \oslash g_{self}) \cup \forall r_m \in \mathbb{S}\ \ (CP^{Selfish}(r_m))$$

where $k_{r_x} = \langle g_{r_x}, p_{r_x}, \sigma_{r_x}, \pi_{r_x}, \mathbb{R} \rangle$ and $\mathbb{S}$ defines the list of specialized roles that is inferred from the role relations $\mathbb{R}$.

Case 3. (*Maximally social enactment*) Agent, even though it does not firmly believe, may act for the sake of the organization's benefits. For this reason, the agent only uses objectives from the role and precisely obeys the organizational rules. Agent ignores its self goals and rules, for the duration of the role enactment.

$$CP^{MaxSocial}(r_x) = k_{r_x} \cup p_{self} \cup (\sigma_{self} \oslash \sigma_{r_x}) \cup \forall\, r_m \in \mathbb{S}\ (CP^{MaxSocial}(r_m))$$

where $k_{r_x} = \langle g_{r_x}, p_{r_x}, \sigma_{r_x}, \pi_{r_x}, \mathbb{R} \rangle$ and $\mathbb{S}$ defines the list of specialized roles that is inferred from the role relations $\mathbb{R}$.

Case 4. (Social enactment) This enactment type is similar to the selfish enactment, but herein the agent includes as many of its own goals and rules as possible but the organizational rules and goals still have the priority.

$$CP^{Social}(r_x)=k_{r_x} \cup p_{self} \cup (\sigma_{self} \oslash \sigma_{r_x}) \cup (\pi_{self} \oslash \pi_{r_x}) \cup (\ g_{self} \oslash g_{r_x}) \cup \forall\ r_m \in \mathbb{S}\ \ (CP^{Social}(r_m))$$

where $k_{r_x}=\langle g_{r_x}, p_{r_x}, \sigma_{r_x}, \pi_{r_x}, \mathbb{R}\rangle$ and $\mathbb{S}$ defines the list of specialized roles that is inferred from the role relations $\mathbb{R}$.

As a special case, if there are no conflicting goals or beliefs between the self and the activated role, then the constructed active knowledge of social enactment is same with the selfish enacted role's active knowledge. In other words, if there are no conflicting rules then, $\pi_{self} \oslash\ \pi_{r_1} = \pi_{self}$, $\pi_{r_1} \oslash\ \pi_{self} = \pi_{r_1}$ and the same for the facts and goals. Hence;

$$O_{activate}(ri_x^{r1}) \rightarrow (k_{self} \cup\ k_{r_1} \cup\ k_{ri_x^{r1}}) \subseteq K_{active}$$

whether $k_{ri_X^{r1}}=\langle x, r_1, \tau_{Social}, \sigma_{ri}, \pi_{ri}, \mathbb{A}\rangle$ or $k_{ri_x^{r1}}=\langle x, r_1\ \tau_{Selfish}, \sigma_{ri}, \pi_{ri}, \mathbb{A}\ \rangle$.

While active knowledge construction ensures that the agent makes deterministic decisions, occasionally agent may need to share a role specific knowledge or a know-how with another role that does not have direct relation. In such a case, agent adopts that knowledge as experience and adds the knowledge to its self knowledge. Thereby, agent can use that knowledge independent from the life-cycle and knowledge scope of the role.

Definition 4. (Suspend) Role instances goes into suspending state when none of its goals and eventually tasks are in execution and there is nothing more to do in the scope of instance. In this case, the entire active knowledge is committed back to the agent knowledge and a new active knowledge is constructed with the subsequent activation operation.

Definition 5. (Deact) Agent deacts a role instance in the case of withdrawing from a social position. Deact operation could be triggered through the achievement of all role goals or the inexpediency of the enacted role instance. This operation deallocates the role instance from the agent and represented as: $O_{deact}(a_1, ri_1^{r1}) \rightarrow k_{self}^{'}$, where $ri_1^{r1} \epsilon Ri$ and $a_1 \epsilon A$. When the role instance's main goal is achieved, the instance is deallocated from the agent.

$$\frac{k_{self}=\left\langle g_{self},p_{self},\sigma_{self},\pi_{self},\Omega\right\rangle \quad O_{deact}(a_1,ri_1^{r1})}{k'_{self}=\left\langle g_{self},p_{self},\sigma_{self},\pi_{self},\Omega'\right\rangle \quad \Omega' = \Omega \setminus ri_x^{r1}}$$

Enacting a role and deacting the same role doesn't always end up with the same agent knowledge, because through the execution of the role instance, agent may gain experience and its self knowledge may evolve. Also, as a rule, if there is an aggregate role which is enacted by the agent, its participant roles could not be deacted.

$$O_{deact}(a_1,ri_1^{r1}) \Rightarrow \forall r_x \in R\ (\mathbb{R}_{aggr}(r_x,r_1) \wedge O_{deact}(a_1,ri^{r_x}))$$

Definition 6. (Remove) Finally, with remove operation, the agent also loses the capability of playing the role. This operation removes the resources of a role and every instances of it from the agent. Role remove is represented as $O_{remove}(a_1, r_1) \rightarrow K'_{passive}$ where $r_1 \epsilon R$ and $a_1 \epsilon A$.

$$\frac{K_{passive}=\langle g_a,p_a,\Sigma_a,\Pi_a\rangle \quad k_{r_1}=\left\langle g_{r_1},p_{r_1},\sigma_{r_1},\pi_{r_1},\mathbb{R}\right\rangle \quad O_{remove}(a_1,r_1)}{K'_{passive} = K_{passive} \setminus (\ k_{r_1} \cup \forall ri_x^{r1}(k_{ri_x^{r1}})\ )}$$

that is $K'_{passive}=\langle g_a \setminus g_{r_1}, p_a \setminus p_{r_1}, \Sigma'_a, \Pi'_a\rangle$ where $\Sigma'_a = \Sigma_a \setminus (\sigma_{r_1} \cup \forall ri_x^{r1}(\sigma_{ri_x^{r1}}))$ and $\Pi'_a = \Pi_a \setminus (\pi_{r_1} \cup \forall ri_x^{r1}(\pi_{ri_x^{r1}}))$.

If there is an instance of the role running on the agent, the role cannot be removed, $O_{remove}(a_1, r_1) \Rightarrow \forall ri_x^{r1} \in Ri\ (O_{deact}(a_1, ri_x^{r1}))$. Also, in order to remove a role which is depended on by another role with the relationship types of specification and aggregation must be removed, $O_{remove}(a_1, r_1) \Rightarrow \forall r_x \in R\ ((\mathbb{R}_{spec}(r_x,r_1) \vee \mathbb{R}_{aggr}(r_x,r_1)) \wedge O_{remove}(a_1, r_x))$.

3 The Architecture

We implemented a knowledgebase architecture, which supports the semantics detailed in the previous section, and incorporated it into the SEAGENT semantic web enabled multi-agent framework[9]. The SEAGENT framework handles all its artifacts such as roles, goals and plans within OWL ontologies. Since OWL is a description logic based knowledge representation language, the SEAGENT framework provides great facilities for implementing mentioned role-based knowledge semantics. In this section, we focus on our knowledgebase implementation and position it in SEAGENT framework.

Figure-2 represents the implementation of the proposed knowledgebase architecture within the SEAGENT framework. As indicated in the figure, two vertical layers compose the overall architecture. The first layer, called Execution Layer, provides planning and execution infrastructure for the SEAGENT agents. The second one, the Semantic Web Layer encapsulates the ontologies that represents the roles, goals, rules and domain specific concepts. This layer also represents the semantic web together with the local ontologies which imports the shared concepts published in the internet. The proposed knowledgebase architecture is composed of three sub-modules: Knowledge Manager, Active Knowledge and Passive Knowledge. The Passive Knowledge module is responsible for storing all local ontologies that are used by the agent. We provide an extensible support for different physical passive knowledge persistence solutions and there are two existing implementations as TDB and RDB light weight ontology storage based on Jena API[1]. On the other hand, Active Knowledge module serves a complete and consistent knowledge subset that is arranged from the passive knowledge in accordance with the active role instance and its dependencies. In fact, this knowledge is an in-memory ontology model that imports all active role instance related

[1] http://jena.sourceforge.net/DB/index.html

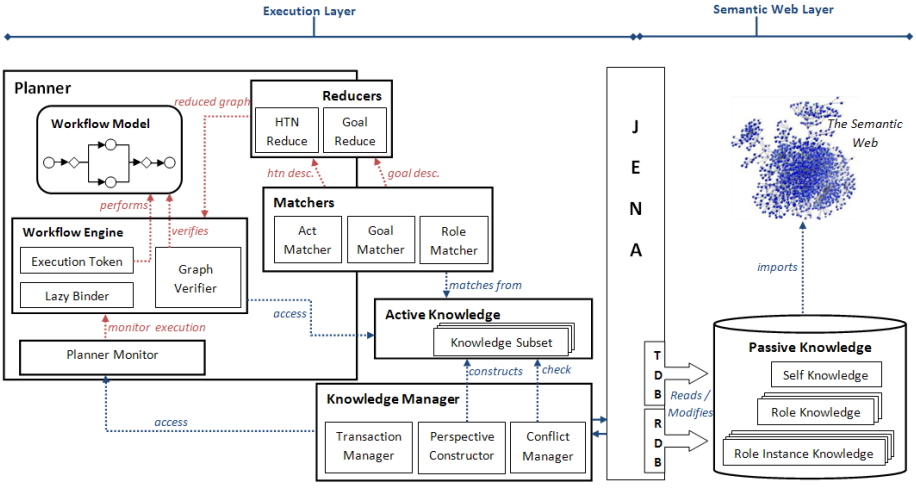

Fig. 2. SEAGENT Knowledgebase Architecture

ontologies and accessible from the tasks of the active role. Lastly, the Knowledge Manager module acts as a bridge between active and passive knowledge modules and has critical responsibilities such as, construction of active knowledge, ensuring the consistency of it and synchronization of the active and passive knowledge. To fulfill these responsibilities, the Knowledge Manager module is composed of three sub-modules that are Perspective Constructor, Transaction Manager and Conflict Manager. Following paragraphs detail the functionality of the Knowledge Manager module.

The Perspective Constructor module is in the charge of the construction of active knowledge. Figure-3 represents the algorithm that constructs the active knowledge. In accordance with the formerly proposed knowledge semantics, this algorithm takes the activated role instance as parameter and simply identifies a maximal related sub graph from a net of role and role instance relations. After the specification of role and role instances that are related with the activated role instance, their knowledge are read into active knowledge in accordance with their enactment type. This provides a consistent and well defined knowledge context, even agent enacts conflicting roles and participates in non-adaptive organizations. This knowledge serves agent a conception of world from the perspective of the active role instance.

Another important function of the Knowledge Manager is the concurrency management. Since selected storage infrastructure is a non-transactional media and we have different atomicity requirements, we implemented our own transaction management sub-system and this functionality is provided by the Transaction Manager sub-module. As emphasized previously, an agent may enact several roles concurrently (one actively at a time), and each role instance may execute several tasks in order to achieve several goals synchronously. However, the Transaction Manager module ensures that only one task can modify an ontology at a

```
function CONSTRUCT_ACTIVE_KNOWLEDGE (role instance)
returns active knowledge
    ak ← empty knowledge
    ak ← CONSTRUCT_PERSPECTIVE (role instance)
    return ak

function CONSTRUCT_PERSPECTIVE (role instance)
returns active knowledge
    role ← role instance type
    enactment type ← role instance enactment type
    ak ← READ_KNOWLEDGE(role instance)
    ak ← CONSTRUCT_PERSPECTIVE (role, enactment type)
    for each aggregate role instance of role instance do
        ak ← CONSTRUCT_PERSPECTIVE (aggregate role instance)
    return ak

function CONSTRUCT_PERSPECTIVE (role, enactment type)
returns active knowledge
    ak ← empty knowledge
    if enactment is MAXSelfish then
        ak ← READ_KNOWLEDGE(self)
        ak ← READ_PLANS(role)
        ak ← role facts not conflicting with self facts
    else if enactment is Selfish then
        ak ← READ_KNOWLEDGE(self)
        ak ← READ_PLANS(role)
        ak ← role fact, rule and goals not conflicting with self knowledge
    else if enactment is MAXSocial then
        ak ← READ_PLANS(self)
        ak ← READ_KNOWLEDGE(role)
        ak ← self facts not conflicting with role facts
    else if enactment is Social then
        ak ← READ_PLANS(self)
        ak ← READ_KNOWLEDGE(role)
        ak ← self fact,rule and goals not conflicting with role knowledge
    for each parent role of role do
        ak ← CONSTRUCT_PERSPECTIVE (parent role, enactment type)
    return ak
```

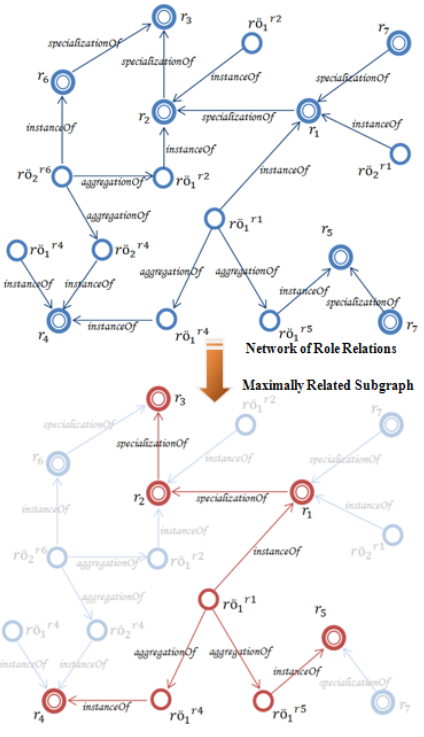

Fig. 3. Active Knowledge Construction Algorithm

time via transaction. In the implementation of our concurrency control infrastructure, two-phase locking algorithm is adopted because of its simplicity and success in practice[13]. Also with the construction of our own transaction mechanism, we are enabled to redefine the atomicity of the knowledge transactions and agent goals are the best guide for this purpose. Transaction Manager commits all un-committed modifications, that are triggered by the task, in the case of achievement of the goal that the task contributes. Or, in the case of an exception that occurred during the execution of the task, all un-committed modifications are aborted. In the case of long-term goals, this approach may cause long-lived transactions that access large number of resources and this may cause deadlocks. To cope with this problem transactions can be prematurely committed within the tasks of the agent.

As a result of the agent's cognitive structure, agent learns and gains experience from its former actions. Besides, consistency of the agent knowledge shouldn't decay with this newly added or inferred knowledge. For each knowledge modification, Conflict Manager checks the consistency of the active knowledge and all suspending role instances' knowledge perspectives. Since the SEAGENT framework represents all knowledge artifacts with OWL ontologies, this consistency check is basically an ontology validation. But from the point of agent view, a

concept's validity is only dependent on the agent's perception on that concept. Hence, open world assumption based inference engines are not sufficient for our requirements. We extended Eyeball[2] tool with new inspectors in order to detect and report goal and rule conflicts. When Conflict Manager distinguishes a knowledge modification that threaten the consistency of the active knowledge, it throws an exception and notifies the SEAGENT exception handler[14].

4 Case Study

This section presents a multi-organization application that is implemented with SEAGENT Multi-Agent Development Framework as a case study to emphasize and observe the benefits of the proposed knowledge architecture. In order to provide conflicting but not totally irrelevant roles to our agents, we preferred the barter domain. Figure-4 represents Corporate Barter and Barter Trade Exchange organizations' design model. The corporate barter organization, or peer to peer barter, is the most plain type of barter which is arranged between two companies who have mutually agreeable goods or services to swap evenly. Participants can openly determine the price of the goods being exchanged and offer discounts without inviting retaliation by competitors or existing customers. This type of organization is a free competition market, where every customer tries to be monopoly in its area of expertise. Hence, companies aim to dominate the market with creating competition via reducing prices and increasing product quality. Explicit representation of the corporate barter organizational rules that are applicable to the customer role are, *(ruleSet1)*:

PERMITTED(*customer* DO *overpricing*(*product*))
PERMITTED(*customer* DO *underpricing*(*product*))
PERMITTED(*customer* DO *offer_discount*(*product*))
OBLIGED((*customer*$_i$ DO *make_payment*(*customer*$_j$, *product*$_i$)
IF(*customer*$_j$ DO *make_payment*(*customer*$_i$, *product*$_j$)
AND (*customer*$_i$, *customer*$_j$ DO *accept_trade*(*product*$_i$, *product*$_j$)))

Similarly, barter trade exchange organization is a collection of businesses that trade their goods and services, managed by an intermediator. In this organization, businesses do not exchange goods directly as in the bilateral form of the corporate barter. Rather, barter trade exchanges is multilateral and participants only interacts with an intermediator. Another difference of barter trade exchange is the use of a form of private label currency. The customer role of this organization stimulates enactor agents to maximize their profit via increasing prices within the legal limits or reducing transportation cost by preferring nearby barter options. The rules and norms in the role description of the Customer role of barter trade exchange organization are, *(ruleSet2)*:

FORBIDDEN((*customer* DO *overpricing*(*product*)
AND(*customer* DO *underpricing*(*product*) AND(*customer* DO *offer_discount*(*product*)
IF(*customer* DO *violate_competition_legislation*(*product_price*)))),
OBLIGED((*customer* DO *make_payment*(*barter_manager*, *product*$_i$)
OR (*make_payment*(*barter_manager*, *barter_dollar*))
IF(*customer* DO *accept_trade*(*product*$_i$, *product*$_j$)
OR *accept_trade*(*product*$_i$, *barter_dollar*))

[2] http://jena.sourceforge.net/Eyeball/

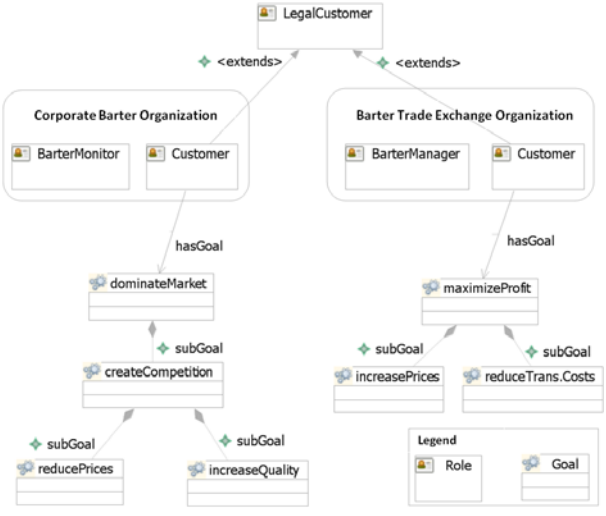

Fig. 4. Organization model

Also, both Customer roles extend from a more generic *Legal Customer* role that is supplied by a competent authority in order to ensure, without considering barter type, all customer roles respect the legislation. Such as, prohibiting barter of unmarketable goods like weapons, drugs and prostitution, or obligating companies to have appropriate license to sell goods that are permitted to be sold only by licensed sellers(such as alcohol), *(ruleSet3)*:

FORBIDDEN(*customer* DO *barter*(*weapon, drug, ...*)),
OBLIGED(*customer* DO *have*(*license*) IF(*customer* DO *barter*(*product*)
AND(*product* DO *require_license*(*license*)))

In order to affirm the expediency of our approach, we will examine a case where a company agent participates *Corporate Barter Organization* and *Barter Trade Exchange Organization* by socially enacting both *Customer Roles* in order to achieve his self goal of consuming the company's unused stocks. Also, the self knowledge has a rule in sales as applying a fixed price to all goods and not making discounts regardless of the amount of the purchase. Similarly, he has a self rule in buying as only trading the indispensable goods, *(ruleSet4)*:

PERMITTED(*self* DO *barter*(*product*) IF(*self* DO *demand*(product))),
FORBIDDEN(*self* DO *discount*(*product*))

While existing agent cognitive models refuse the simultaneous enactment of both customer roles because of their conflicting rules on pricing policies, it could be easily seen that this is not an uncommon and avoidable case. As a part of the solution that we proposed for this problem, SEAGENT agents stores both rules, as SWRL rules, and goals, as goal ontologies, into his passive knowledge. In a case where another agent, which is also participated in *Corporate Barter Organization,* makes a barter offer with a very low profit margin and even quite

under the product's fair market value, our agent activates the appropriate role instance and evaluates his knowledge in order to make a decision.

With a traditional knowledgebase, both rule sets would be included in an aggregate agent knowledge and agent would query whole related and unrelated knowledge in order to make a decision. Correspondingly, while this very low profited barter is permitted by the *Corporate Barter Organization*, underpricing which violates competition legislation are banned in the *Barter Trade Exchange Organization.* Also, while *Corporate Barter Organization* permits offering discount, self of the agent adopts one-price policy:

PERMITTED(*customer* DO *underpricing*(*product*) - **from ruleSet1**
FORBIDDEN((*customer* DO *underpricing*(*product*) - **from ruleSet2**
IF(*customer* DO *violate_ competition_ legislation*(*product_price*))))
PERMITTED(*customer* DO *offer_ discount*(*product*)) - **from ruleSet1**
FORBIDDEN(*self* DO *discount*(*product*)) - **from ruleSet4**

On the other hand, in our approach with the activation of the customer role instance of the *Corporate Barter Organization,* the knowledge manager of the agent reconstitutes the active knowledge. In accordance with active knowledge construction semantics defined in previous section, this knowledge contains the active role instance's knowledge and since the enactment type is social, *Corporate Barter Organization*'s Customer role's knowledge, Legal Customer's knowledge and the subset of the self knowledge that does not conflict with the role knowledge. In the following rule-set, content of the active knowledge is shown.

PERMITTED(*customer* DO *overpricing*(*product*)),
PERMITTED(*customer* DO *underpricing*(*product*)),
PERMITTED(*customer* DO *offer_ discount*(*product*)),
OBLIGED((*customer*$_i$ DO *make_ payment*(*customer*$_j$, *product*$_i$)
IF(*customer*$_j$ DO *make_ payment*(*customer*$_i$, *product*$_j$)
AND (*customer*$_i$, *customer*$_j$ DO *accept_ trade*(*product*$_i$, *product*$_j$))),
FORBIDDEN(*customer* DO *barter*(*weapon, drug, ...*)),
OBLIGED(*customer* DO *have*(*license*)
IF((*customer* DO *barter*(*product*)) AND(*product* DO *require_ license*(*license*)))),
PERMITTED(*self* DO *barter*(*product*) IF(*self* DO *demand*(product)))

The constituted active knowledge that is clarified from conflicting and irrelevant knowledge represents the exact perception of the world of the activated role instance. Hence, this approach makes enactment of the roles which have conflicting knowledge possible and ensures the integrity of the global identity.

5 Related Work

In the agent research domain, there are some approaches which are motivated to handle inconsistent knowledge of agent by the approach of clustering[15,16]. In [16], Halpern and Fagin presents a model of local reasoning, where an agent's knowledge is handled as a "society of minds", each with its own cluster of beliefs, which may contradict with each other. In each frame of mind, the agent beliefs are consistent, but the conclusion that the agent draws in different frames of mind may be inconsistent. Differently from our role-based approach, in the model of this research, knowledge clusters are defined as non-interacting and agent can utilize only one cluster at a time. So, they develop a new modal operator to be able to define axioms that are true in all frames of the mind.

On the other hand Wassermann[15], similarly, proposes a belief base structure that simulate commonsense reasoning in an psychologically way, that amplify the RABIT system. This system consists of four modules: LTM(Long-Term Memory) that all agent beliefs are stored, STM(Short-Term Memory) which is a small subset of LTM where the reasoning (decision making) takes place, ITM(Intermediate-Term Memory) that stores the history of the reasoning process and finally RTM(Relevant-Term Memory) which is a kind of context, storing the relevant concepts. Also, an heuristic algorithm to retrieve the most relevant beliefs from a structured belief base(LTM) into the STM is introduced. But, because of the absence of a well defined context scope, we cannot affirm that the constructed STM embody the exact mental state of the agent.

Another research, that must be touched on here due to its conceptually similarity with our approach, is the VIKEF project[17]. The VIKEF project aims to create large-scale information systems that base on Semantic Web Technology. At the center of the system there is an RDF knowledge base which contains a large amount of information about documents and their contents. But, as a result of the nature of web, it is obvious that conflicting information will arise. To cope with this problem, they propose a context-based system to store semantically contradictory statements in the knowledge base. But they use relations between contexts, called compatibility relations, to gather related contexts differently from our role-based approach.

6 Conclusion

In this paper we propose a knowledgebase approach to increase applicability of role concept in the MAS research domain. While developing this role-based agent knowledgebase architecture, we dive into the depths of the role theory which is developed in consequence of sociological empirical researches. By the help of role theory, sociologists also use roles to predict human behaviors according to the body of knowledge about human organizations[1]. In the development of our knowledgebase approach, we inspire from the self concept that has an important place in the sociological role theory to resolve human behaviors in conflicting social positions.

We also implement this knowledge approach and integrate it into the SEAGENT multi-agent development framework[9]. The proposed knowledgebase architecture is separated into two parts; active and passive knowledge. Both knowledge parts consist of knowledge blocks each of which contains consistent information about agent, role or role instance. While in the passive knowledge persistence of knowledge blocks are provided, on the other hand, in the active knowledge related knowledge blocks are arranged in a consistent form to provide consistent knowledge to the agent for deterministic decision making. In Section 2, we declare the semantics of these knowledge blocks and how active and passive knowledge evolve in the life-cycle of role-based agent. To measure applicability of our approach and ensure the implemented architecture, we also develop a case study in electronic barter domain application in Section 4. Within the case

study, we observe that an agent can play roles which have conflicting knowledge. In the further steps of this research, we aim to cope with management of the knowledge from the organizational perspective.

References

1. Biddle, B.J.: Recent developments in role theory. Annual Review of Sociology 12, 67–92 (1986)
2. Allen, V.L., van de Vliert, E.: A role theoretical perspective on transitional processes. Role Transitions: Explorations and Explanations, 3–18 (1984)
3. Dastani, M.M., van Riemsdijk, M.B., Hulstijn, J., Dignum, F.P.M., Meyer, J.-J.C.: Enacting and deacting roles in agent programming. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 189–204. Springer, Heidelberg (2005)
4. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: Autonomous agents and multiagent systems (AAMAS), pp. 489–496. ACM, New York (2003)
5. Marks, M.S.M., Stephen, R.: Multiple roles and the self: a theory of role balance. Journal of Marriage and the Family (1996)
6. Goffman, E.: The Presentation of Self in Everyday Life. Anchor (June 1959)
7. Stryker, S., Serpe, R.T.: Identity salience and psychological centrality: Equivalent, overlapping, or complementary concepts? Social Psychology Quarterly 57, 16–35 (1994)
8. Zhang, X., Xu, H., Shrestha, B.: An integrated role-based approach for modeling, designing and implementing multi-agent systems. Journal of the Brazilian Computer Society (JBCS) 13, 45–60 (2007)
9. Dikenelli, O.: Seagent mas platform development environment. In: Autonomous Agents and Multiagent Systems, pp. 1671–1672 (2008)
10. Kristensen, B.B., Osterbye, K.: Roles: conceptual abstraction theory and practical language issues. Theor. Pract. Object Syst. 2(3), 143–160 (1996)
11. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
12. Giorgini, P., Müller, J.P., Odell, J.J. (eds.): AOSE 2003. LNCS, vol. 2935. Springer, Heidelberg (2004)
13. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency control and recovery in database systems. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1987)
14. Cakirlar, I., Ekinci, E.E., Dikenelli, O.: Exception handling in goal-oriented multiagent systems. In: Proceedings of the workshop Engineering Societies in the Agents World IX. Springer, Heidelberg (2009)
15. Wassermann, R.: On structured belief bases. In: Frontiers in Belief Revision. Applied Logic Series, vol. 22, pp. 349–369. Kluwer Academic, Dordrecht (2001)
16. Fagin, R., Halpern, J.Y.: Belief, awareness, and limited reasoning. Artificial Intelligence 34(1), 39–76 (1987)
17. Stoermer, H., Palmisano, I., Redavid, D., Iannone, L., Bouquet, P., Semeraro, G.: Contextualization of a RDF knowledge base in the VIKEF project. In: Sugimoto, S., Hunter, J., Rauber, A., Morishima, A. (eds.) ICADL 2006. LNCS, vol. 4312, pp. 101–110. Springer, Heidelberg (2006)