IJE TRANSACTIONS B: Applications Vol. 25, No. 3, (August 2012) 233-248

International Journal of Engineering

Journal Homepage: www.ije.ir

A Context-aware Architecture for Mental Model Sharing through Semantic Movement in Intelligent Agents

S. Salehi [a], F. Taghiyareh [a], M. Saffar [a,*], K. Badie [b]

[a] Department of ECE, University of Tehran, Postal Code: 14395-515, Tehran, Iran
[b] Department of IT, Iran Telecommunication Research Centre, Tehran, Iran

PAPER INFO

Paper history:
Received 12 October 2011
Received in revised form 05 February 2012
Accepted 17 May 2012

Keywords:
Autonomous Agent
Semantic Movement
Shared Mental Model
Mental Model
Context-aware
Architecture
Intelligent Agent

ABSTRACT

Recent studies in multi-agent systems are paying increasingly more attention to the paradigm of designing intelligent agents with human inspired concepts. One of the main cognitive concepts driving the core of many recent approaches in multi agent systems is shared mental models. In this paper, we propose an architecture for sharing mental models based on a new concept called semantic movement. This architecture has been inspired by a variety of mental models in humans and supports agents working in different simultaneous contexts and it uses semantic movement as a conflict resolution mechanism. Semantic movement is a kind of transition from the present mental states to some new states in order to resolve harming conflicts between mental models of participants. We formalized the semantic movement process in order to use it in our proposed architecture. Our test bed to evaluate this architecture is a set of complex scenarios which are likely impossible to be solved by individual agents without sharing. Our architecture exhibits better performance than other alternative methods that have sharing capabilities. We believe that the proposed architecture would be able to provide agents with the ability of generating more consistent behaviors between agents. Moreover, this architecture can become a suitable platform for the negotiation of self interested agents.

doi: *10.5829/idosi.ije.2012.25.03b.10*

1. INTRODUCTION

The concept of mental model was first introduced in 1943 by Craik [1]. He suggested that human's mind creates small scale models from reality to anticipate events. Several definitions for mental model have been proposed in the literature. Definitions usually point out that a mental model consists of knowledge about a situation or a physical system that can be used to reason about statements or predict the possible outcomes of executing actions. One of the mostly cited functional definitions of mental model that we use in this work is as follows as described by Baldauf et al. [2]:

"Mental models are the mechanisms whereby humans are able to generate descriptions of system purpose and form, explanations of system functioning and observed system states, and predictions of future system states."

It should be noted that mental models with cognitive flavors have the capability to show up better in knowledge based applications such as multi-agent systems, web intelligence and human-computer interactions [3- 10]. Although most of the research in this area is concerned with using mental models to enhance teamwork performance in a specific task, the process of obtaining the mental models and sharing them is not elaborated enough. Also, researchers spend little effort on analyzing the application of mental models in other domains such as web services and their discovering and composition. Let's say despite the fact that some researchers have considered the role of agents and their organizational properties but only a few of them have tackled the problem of conflict resolution within the entire process of knowledge transference between agents.

Agents can work on the provider or user side or even in the middle of the links between providers and users. These agents can discover or compose web services with the help of common agent abilities like negotiation and communication. There are several known methods to discover and compose web service and to offer selected web services to users. In most of them, the agents are operating in an isolated environment with a minimum capability for communication which makes

* Corresponding Author Email: saffar@ut.ac.ir (M. Saffar)

cooperation almost impossible. For example, one agent is at the user side and one agent is at the provider side and these agents negotiate to offer service to user. We can improve the performance of these agents if we replace them with a multi-agent system in which each agent tries to capture the user's interests from a specific point of view. It makes the system more organized and easy to change or extend by simply adding new agents or changing agents that are surely more cohesive than the single agents in use before. These agents can share their mental models to reach a compatible and complete image of the user. This idea can also be practical for agents who work on provider's side. Middle agents also can get replaced with multi-agent systems. Each agent in these systems can have a different view of knowledge and they can share their mental models to select more suitable services for user's needs.

A notable amount of work related to mental models is about predicting a human mental model by a multi-agent system. The previous behaviors of the human which were collected by the system are analyzed periodically in order to improve the approximation of the mental model of human stored in the system and to suggest the person more suitable behaviors in the future. These human assistant agents are influential in different organizations. For example in a human–agent team, because the computer agents can predict the human mental model, they can work better together with human and the teamwork performance increases.

As was stated, in most of the related works in mental models, the agent's mind structure is not defined. In the other words, they assume that the mental models are already present and they try to introduce the appropriate algorithm to share them. These algorithms consider the role of agents and share agents' mental models based on their roles. In fact, they are dependent on the role of agents. Furthermore, all of these works don't consider the context as a section of mental model. In order to rectify these shortcomings, in this paper, we want to introduce a context-aware mental model to enable the process of sharing them with the use of semantic movement. Not all of the knowledge should be shared in order to improve the performance of the entire multi-agent system. Sharing the correct subset of knowledge can be done with respect to the context. A context is based on the task that agents should perform or the environment they operate in. For example, in soccer simulation, contexts can be defined based on tasks such as attack, defense, etc. In the application of intelligent building, contexts can be defined based on shared environments such as rooms, halls, etc. In some other applications, contexts can be defined with respect to both tasks and environments.

Sharing mental models is the process of identifying and eliminating conflicts between mental models in different agents working together in the same context. Semantic movement is the process of transforming, transferring and adapting the implicit knowledge stored in the mental model of an agent in order to make it more consistent with the knowledge of the group.

The rest of this paper is organized as follows. In the next section, a brief explanation of previous works is presented. The section 3 discusses the agent role in semantic web. Next section explains the context structure and the new concept of semantic movement are introduced in section 5. The section after that defines the overall proposed architecture for mental model sharing. The properties and features of this architecture are explained in section 7 followed by the sharing strategies and their properties. Section 8 also explains how the sharing strategies are adapted to be suitable for using in our architecture. The experiment platform is introduced in the next session, after that some practical scenarios of applying our architecture in the experiment platform are explained in section 10 and section 11 discusses the experimental results. Finally the conclusion and future works are presented.

2. PREVIOUS WORK

Multi-agent systems can be one of the main tools in composing and selecting web services. A large number of researches in the semantic web and web service domains use some kind of multi-agent platform to enhance different parts of their solutions. Although in lots of these works, they have little considerations about issues like team working and knowledge sharing which can potentially improve the performance of these systems.

Hendler [11] discussed how a semantic web infrastructure can be augmented by the help of ontology to have more powerful agent based approaches in the semantic web domain. The design and construction of web services to support situational awareness is described by Gibbins et al. [12]. It also mentioned that some properties or known methods in multi-agent systems like separation of message contents from application domain can get easily applied in web services. Coalition et al. [13] has described the web service capabilities and properties. Mc Ilraith and Son [14] used agent technologies to provide generic procedures and to customize user constraints. They applied Golog to compose web services and specify the sequence of them. S. Narayanan and Mc Ilraith [15] used semantic for web service simulation, verification and composition. Buhler and Vidal [16] discussed the application of agents in the web service description tasks. An architecture to automatically connect agents and web services is introduced by Greenwood and Calisti [17]. In this method, web services invoke agents' services and agents offer appropriate

suggestions. Maximilien and Singh [18] applied middle agents to select appropriate web services for different applications. Maximilien and Singh [19] implied a dynamic service selection routine by using an agent framework. They used agents to simulate providers and consumers in a service configuration environment. A context aware agent based approach for web service composition was introduced by Maamar et al. [20]. They applied agents and context properties efficiently to compose web services.

Another major concept according to this work is mental models. The work related to mental models can be categorized into two major subsets. Researches in the first category concentrate on mental models and share mental models. A mental model ontology is presented in literature [21]. It propounds the appropriate description for the concept of mental model. A role-based shared mental model was introduced by Yu Zhang [6]. A shared mental model in this paper is a graph, representing a complete picture of the team plan that each part of the plan is assigned to a specific role. In this method, each agent extracts its local mental model from the shared mental model. It is left unmentioned how this shared mental model can be formed. Finally, we believe that the idea of obtaining local mental models from a shared mental model should be reversed. In cognitive theories for shared mental models a shared mental model should be formed from local mental models while agents interact and collaborate with others.

Kaivan Kamali et al. [22] proposed multi-part proactive communication method. This kind of communication was used as a base for creating a shared mental model between agents in the system. Daniel Fuller et al. [10] introduced shared mental model for improvisational performance. It is claimed that this model can show good performance under situations where users show unpredictable behaviors. Conflicts between mental models are detected by cognitive divergence. Resolving conflicts are done by cognitive convergence.

Schmitt et al. [23] introduced a framework for the engineering of context aware systems. It discussed that context awareness is relevant to ambient intelligent systems in order to provide adequate services for current situation and their users to improve satisfying interaction with systems. This framework determines user's mental model and the manner that they build, refine, change and discard it. In other words, it presents a framework for context-aware systems and in this framework it determines user's mental model. But this work doesn't consider mental model's structure and sharing algorithm to form shared mental model while we introduce a context-aware architecture for agents' mental model and describe a method to share them.

CAST is a team model for predicting others information requirements and proactive information sharing between agents or humans in the system. It is based on a Petri-net model that contains information about agents' responsibilities and the team's current status. John Yen et al. [7] added a decision making component to CAST which optimizes the proactive information sharing behavior by deciding when others need particular information or by summarizing information before sending it to others. Phillips et al. [24] considered mental model of robots and figure on elements that influence it.

In the second category related to mental models, Brigitte Burgemeestre et al. [25], Xiaocong Fan et al. [8- 9] and Kennedy et al. [26] considered the problem of making agents that can work with humans. In teams with both humans and artificial agents, two categories are formed. Agents can represent or simulate a human or agents can be as teammates for humans. In both categories, it is desirable that agents collect information about humans' mental models in order to better cope with them.

3. WEB SERVICE, MULTI-AGENT SYSTEM

Multi-agent systems have lots of applications in semantic web and web services domains. In this section, few of these applications are named and explained briefly. There are lots of organizations with a number of departments that each of them has a web application for its own. These web applications usually work on contents that are either stored locally or are stored on the side of another department's web application. Multi-agent systems can work on connecting different web applications for different departments. For example, Hendler [11] used this method to find possible ways for satisfying user needs and suggesting them to the users.
The process of using a web service can be seen as an agent that advertises its functionalities and other agents requesting those available services from the serving agent [16]. Using different ontologies for communication on the provider side and the user side, makes using of the services hard or impossible [27]. An agent can also be embedded in a provider or user of a web service. Middle agents are also playing an important role in managing and enhancing web service structure [28].

Web services are passive. They are only waiting for a request to receive and provide the service. Using agents inside a web service provider can add proactive behaviors to the previously simple passive web service. For example, an agent inside a web service can proactively monitor other possible web services to track any changes in their status or API and analyze these changes to make an approximation of the shift in users' interests. It can finally help web service providers to

adapt their system according to users' needs. The agent inside a web service provider also enables the web services to form a team and provide service with better quality [26].

The main challenge of selecting the right web service for a user with special needs is still under investigation by researchers in the field of Semantic Web. Multi-agent systems, if used properly can be the perfect tool for this task. Different agents in the system are responsible for modeling different interests of the individuals. They should merge their models to create a consistent complete model of the user. This process can be done with mental model sharing introduced in this work.

Middle agents in a web service environment are responsible for discovering and composing possible web services based on user's needs. Just like multi-agent systems operating in a provider or user side, the agents in the middle can form a multi-agent system. Connecting these middle agents can significantly improve their performance.

There may be lots of other applications of multi-agent systems in the semantic web and web service domains. We only highlighted some main ideas according to this approach. In almost all of these ideas, enabling agents in the system with the capability of mental model sharing may improve the selection and composition of services leading to more users' satisfaction.

4. CONTEXT STRUCTURE

There are two major trends in multi-agent research, context-aware and context-free systems. The work related to context-aware system can be categorized into two classes. The first class has concentrated on basic issues in this trend and its properties like reports by Baldauf et al. [2], Hong et al. [29], Payton [30] and several other researchers, while the second has investigated the usage of properties of context-aware systems in other applications especially in services like Arabshian and Schulzrinne [31], Sadeh et al. [32], Hattori et al. [33] and several other researchers.

Common architecture principles of context-aware systems are introduced by Baldauf et al. [2]. It recommends a layered conceptual design framework to explain elements of context-aware architecture. Hong et al. [29] suggested a new classification framework of context-aware systems based on the architecture and explored its features. J. Payton [30] introduced a new context-aware system that aimed to reduce the programming efforts by hiding the details of agent coordination in the process of producing the context-aware system. H. J. Lee et al. [34] described the overall architecture of an agent based context-aware system and used it for generating high level context to the application and services. A multi-agent based architecture for context-aware system was presented by Chun-Dong and Xiu-Feng [35]. The architecture here is based on multi-agent systems and the concept of role, which is used to assign responsibilities to agents. Lim et al. [37- 38] investigated the effectiveness of some types of explanations with the goal of increasing user trust and acceptance of context-aware systems. They introduced ten question type and studied users' understanding of the system. Another view of context-aware systems, is applying context to other applications in order to improve their performance. Arabshian and Schulzrinne [31] introduced a distributed context-aware service discovery by using context ontology and recording context history. An agent-based environment for context-aware mobile services was presented by Sadeh et al. [32] such that it had a set of ontologies for describing personal resources, contextual attributes, user preferences and web services. Hattori et al. [33] used a context reasoning agent for automatic recognition of data and events by realization of environment and user context. Kwon et al. [39] proposed a proactive need identification mechanism for a personalized reminder system that identified the user's current needs based on user's current context. A framework that is composed of a set of policy based agents for providing a context-aware mobile working environment has been introduced by Harroud and Karmouch [40]. Agents' policies get updated to new policies based on the current context. Burkle et al. [41] presented an agent based architecture that aimed to help the collaboration of context-aware services.

In this paper we used the properties of context-aware systems in order to improve our mental architecture and sharing strategy. We improvise context as a layer in our architecture to benefit from its advantages. Section 6.1 explains this layer.

5. NEW CONCEPT OF SEMANTIC MOVEMENT

The existing approaches for mental model sharing are such that they mostly rely on the role of agents (with regard to their assigned tasks) with no particular attention to the very least requirements which are necessary to reach a collective commitment. This leads to deferring any sort of convergence with regard to the desired process. Let's say much amount of time and cost maybe spend on resolving irrelevant kinds of inconsistencies or conflicts between the participating agents which are not of particular sense to the desired context. To circumvent this problem, a context-aware architecture is required that can be able to respond to mental model sharing only at the stages where conflicts under study are relevant to the current context.

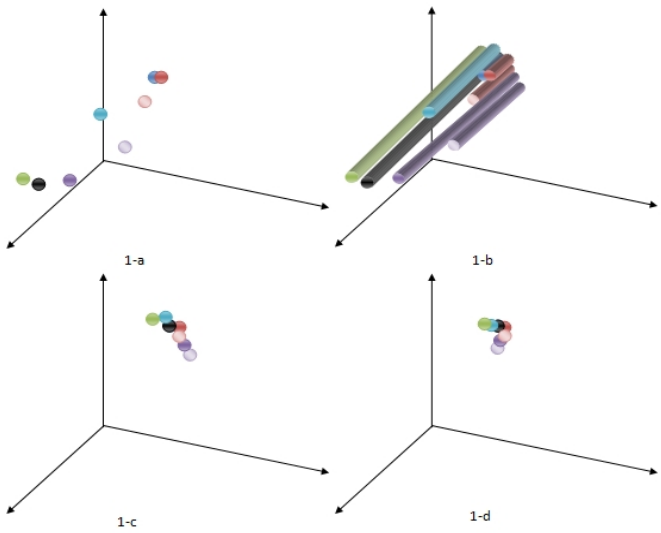

Figure 1. (a) The agents' mental models, **(b)** The projection process, **(c)** Projected mental models before conflict resolution and convergence, **(d)** Projected mental models without destructive conflict

This leads to a promising saving in time and cost essential to sharing process.

To achieve this, in this paper we introduce the concept of semantic movement according to which the propositions belonging to agents' beliefs are changed in such a way that they may finally end up with optimal positions for which no further conflict may be seen between the mental models of the participating agents. The configuration to be obtained under such conditions can be regarded as a safe zone within which no further conflict would be expected between the agents' mental models for the corresponding context. Taking this point into account, one may expect the agents to communicate safely with each other as far as no change has accrued with regard to the problem context. Figure 1 shows schematic representation for semantic movement of the agents' mental models.

Such an approach to partial checking of conflicts between agents has the ability to realize time efficient sharing for real time tasks such as soccer agents. The proposed procedure for semantic movement is:

While (true)

If (sharing time)

Select suitable method to mask agents' mental models

Project each agent's mental model by selected method

While (if exist any destructive conflict in projected mental models)

Resolve detected conflicts

End

End

Else

.

.

.

End

End

6. THE PROPOSED ARCHITECTURE

In this paper, we propose an architecture for a single agent that provides the agent with a shareable mental model. This mental model is designed in a way that significantly improves the sharing procedure in a context-aware multi-agent system. Also, it is capable of storing any kind of knowledge including knowledge about the facts, rules, procedures, strategies, team plans, etc. In this section, we want to describe this architecture. This architecture is built with three layers called "context layer", "agent layer" and "mental model layer". It also has a cross layer part named "background" that is accessible from all three layers. In Figure 2 a simple schema is shown for architecture.

6. 1. Context Layer The top layer of architecture is the context layer. All possible contexts that the system can work in are defined here. When agents sense the surrounding environment, they might derive new facts about it. The context layer gets these facts as inputs.

Also, this layer has a set of rules for context selection. These rules fire according to perceived facts that agents believe them. A context is defined as a subset of properties that can be saved in the mental model. Since we use this architecture to share agents' mental models, this definition is appropriate to clarify which property belongs to which context and specify active context.

With this definition, a context is basically defined by a criteria function, deciding which fields of the mental model are important for reasoning or sharing in that context and which fields are not. Indeed, the output of this layer is the masks that specify which fields are relevant to which context. This layer determines the active contexts and their masks and passes them to agent layer. While working in a context, only the fields relevant to that context are available to processes like reasoning or sharing. This clearly simplifies these processes because they have to deal with smaller size mental models. Properties of context and selection rules are determined by domain experts now, but automatically specifying this properties and rules is the next step of our work.

Context layer is the only layer among others that has some information about the whole domain of the problem rather than individual agents. All agents in the system are sharing this layer. This implies that the definition of contexts is shared between all agents in the system and there is no agent with individual opinion about contexts.

One of the main features of this layer is the flexibility of defining new contexts or editing previously defined ones whenever needed. This model doesn't restrict the definition of contexts to be fixed before the execution of the system and with a good implementation of this model contexts can change at run-time. Another feature of this layer is that, contexts can have shared properties. With shared properties, one can model the changes that might occur in other contexts while the system is working in the active context. The more shared properties two contexts have, the more it is possible that working in one of them changes the mental model state in the other one.

6. 2. Agent Layer The second layer in our architecture is the agent layer. Every agent in the system has a record in this layer. A record is simply an instance of the mental model schema defined for agents. The mental model schema contains all attributes of all contexts with the lowest level of abstraction. This enables the mental model to get refined to the appropriate mental model for the active context with the selected level of abstraction by applying generalization /specification defined in the background layer and context projection mechanisms defined in the context layer. To move the mental model to a higher level of abstraction, it is needed to aggregate data in the lower level to compute values of higher level properties of mental model.

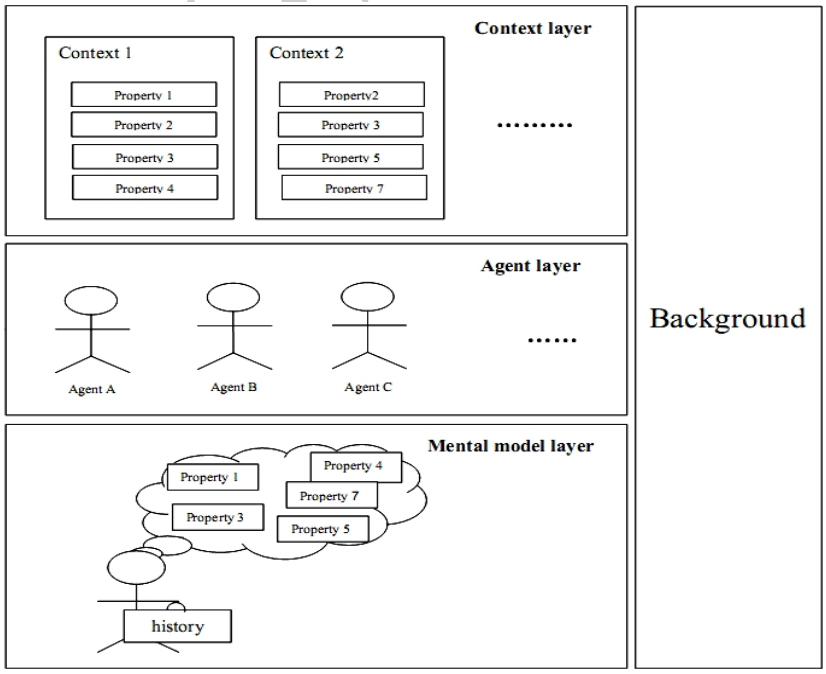

Figure 2. Our proposed architecture for mental model sharing

The masks of active context are passed to this layer as input. The agent layer contains a set of data structure that show mental model of agents in the system. Indeed, this layer contains the metaphor of each agent. It should be pointed out that this conceptual grouping of data in a single layer does not imply that storing them should also be centralized. In fact, this layer is an intermediate layer and is used to manage agents in the system. In other words this layer is responsible for managing the specific properties of agents, arrival of new agents, exit of existing agents and etc. Each agent's metaphor in this layer has been linked to agent's data structures in mental model layer and in that layer every agent should store and update its own mental model and it is responsible for the processing needed for moving the mental model to a different level of abstraction and for masking it with a different context.

The number of fields and the specific type of data for each field are application dependent so the designer of the system should also design the mental model suitable for that application. It is worthy to mention that the model does not put any restrictions on the number and the types of fields.

6. 3. Mental Model Layer The lowest layer of our architecture is the mental model layer. In this layer each agent has a history of mental model changes for each context it has worked in. All of the properties in agents and their values over time are stored in this layer. This layer receive projected mental model as input from agent layer. Then, agents sense the environment and store perceived value in the pertaining data structure. These values use to infer high level information and to detect harmful conflicts. Indeed the output of this layer is the knowledge that results in an action or a conflict with other agents. Storing the whole history from the time of the system startup can be unrealistic in some applications because of storage capacity limitations and restrict ions in analyze or retrieval time of the system. To put an upper bound on the size of history kept for each agent we defined a window. Each snapshot of the mental model that is outside of this window is getting thrown away from this layer. The window size is a parameter that the designer of the system should tune and our architecture doesn't put any limitations on it. It is important to note that working in a context does not push out the history of other contexts and it only changes the history of the active context. Each agent is responsible for storing and updating its own layer of mental models and the grouping of these histories of mental models are only conceptual and not physical.

Each agent can refer to its history in the mental model layer in order to analyze its trend of semantic movement. Any kind of data mining techniques can be done on the mental models in this layer and the results can be used in adjusting agent's behaviors. It may also provide training data for a learning algorithm like reinforcement learning or case-based learning. Designers of the systems using our architecture can implement their own algorithms for using the mental model layer, but in the architecture it is mainly used as a source of data for analyzing and adjusting semantic movement strategies.

6. 4. Background Background is another shared part of the architecture like context layer. It is placed in the model to describe the knowledge that agents might have. It can be described as a shared ontology that defines the hierarchy of abstraction levels, the name and meaning and relation of properties in the mental model, the methods for aggregating data to generate higher level abstractions of mental model and the semantic movement strategies that one agent might use. The background part contains the information about relations between entities and one of the relations especially important in our work is generalization/ specification relation. This kind of relation can be used to switch the level of abstraction of the mental models between different levels. Higher levels of abstraction make the mental models to filter unnecessary fields or aggregate detailed fields to be in a more general and compact state. Lower levels of abstraction result in more detailed mental models. Working on the correct level of abstraction can help the reasoning and sharing mechanisms to work efficiently and as agile as possible. As an example, consider our experiment environment, the wumpus word introduced in section 9. In this environment, the sense of breeze and glow at a cell can be the sign of hole at neighbor cells. This that breeze and glow must have a relation with the concept of hole in the background.

The knowledge stored in this part is application dependent, so it is the designer's duty to properly define the background part so that it correctly reflects the properties of the domain of interest and gives the system the ability of working in different levels of abstraction. Also, this part provides three other layers with all information that each agent may require. In other words this information is output of this layer.

7. PROPERTIES OF ARCHITECTURE

We can classify main properties of architecture in four categories:

1. The properties that context layer add to system
2. The advantages of background
3. Hierarchical structure of mental model
4. The properties of history in mental model layer

7. 1. Being Context Aware Being context-aware is a property of the agents using our architecture. In lots of

applications, it is desirable that agents perform tasks in different contexts simultaneously. In real-world applications like intelligent buildings or complicated control systems, each agent should consider several aspects of the environment in order to successfully make a plan for its work. While acting in a context, the reasoning process should only act on beliefs and knowledge relevant to that context. In other words, mental models should get projected based on the current context to rule out unnecessary information about other contexts. Another important effect of using a context-aware architecture is that it boosts the sharing process by filtering possible non relevant conflicts that agents might have that have no effect on the active context of both agents. While being context-aware is beneficial, it also proposes its own set of problems. For example, it is necessary to have a method for identifying the current working context of other agents in the system. The problem of identifying the active context for each agent is simplified in this work with the synchronized change in context for all participating agents.

7. 2. Advantages of Background Another important property of our architecture is that is provides the system with a shared ontology. This ontology represents the hierarchical structure of knowledge in all contexts together along with the information about conflicting values. It is vital to have a shared ontology between agents who want to share their mental models because it makes them possible to understand and use others' knowledge. It also helps in the conflict detection phase, because an agent can refer to this ontology to judge about presence of conflict between its mental model and others. It also helps agents to narrow down to the root of a conflict by moving down in the hierarchy of abstraction levels in shared ontology. This can help in conflict resolution by isolating and pinning down the root of conflict in mental models of agents.

7. 3. Hierarchical Structure Our architecture is capable of showing mental models in a hierarchical way. This can help agents to store their mental model in different levels of abstraction, enabling them to operate on the appropriate level of abstraction that fits the best for the active task. Omitting unnecessary details while reasoning, can significantly reduce the running time and help to reach the desired goals faster. Another beneficial property of hierarchical mental models is that conflict detection can be done in a systematical manner, by starting from the most abstract level of mental model and narrowing down in the conflicting parts in order to find the cause of conflicts in lower levels and finally resolution of them in the lowest possible level.

7. 4. History Our architecture also incorporates a history of mental models that saves the recent changes in different contexts. A recent history of the changes in mental model can show the semantic movement of agents in the near past. An agent can periodically evaluate its performance to decide whether its semantic movement policy is efficient or not. An agent can determine its trend of semantic movement by analyzing its history of mental model changes. Based on the current performance and the current semantic movement strategy, an agent can decide whether a change in the strategy is needed in order to improve its performance.

8. SHARING STRATEGY

In lots of applications that embody mechanisms like teamwork or negotiations, the sharing process introduced in this section can be done prior to any other step. Even it can help in the task of dynamic team formation in which agents are reasoning about the team and are deciding about the formation of the team. There are several issues one should take into account when he/she wants to use the proposed architecture. Issues like when to share the mental models and who has to participate in the sharing process are the most important decisions that should be made by taking into account the inherent properties of the application domain. It is worth mentioning that two basic but effective methods for deciding when the sharing is effective are, using a timer or delegating an agent to decide about sharing time by monitoring a performance measurement and see when it gets below a threshold.

The sharing strategy has three stages, conflict detection, mental model sharing and conflict resolution. To keep the model general enough, the conflict detection criteria should be defined by the application domain expert. To show that conflict detection is purely domain dependent the following example can be mentioned. Imagine that several agents are responsible to monitor vital signs of a patient. One might report the temperature of a patient is 37°C and the other senses it as 38°C. This is clearly a conflict in this domain. Now imagine another task involving the control of the room temperature. One controller thinks the room temperature is 20°C and another one thinks it is 21°C. In the domain of room temperature this difference can be safely ignored so it is not a conflict. The conflict detection criteria is a function that accepts two values of the same property and returns a Boolean value showing that if a conflict exists between these values or not.

To share the mental models, context layer and background structure should be used. Context structure is used in two ways. The first usage is to determine which conflicting agents should anticipate in the sharing process. Each agent in the system has an active context at each time interval. The agent can switch its context but it can't work in different contexts at the same time.

After conflict detection, if the conflicting agents are working in the same context they should share their mental models. Also, if the agents are in different contexts, the domain expert should provide a suitable guideline for the system, to decide whether sharing should be done between the agents in those contexts or not. The second usage of context layer in the sharing process is that context provides a mask for the mental models which are about to get shared. Each agent should only send the part of its mental model which is relevant to its active context. When the agents are working in different contexts, the masked mental models don't have the same properties. The agents in this situation receive masked mental models of other agents in different contexts and use the whole masked received mental models and not only the parts related to its own context for reasoning.

The main application of background structure in the conflict detection phase is to work on the correct level of abstraction for sharing mental models. It starts with the highest level of abstraction and each conflicting agent is giving out its masked mental model in the highest abstraction level. The masked mental models are then analyzed for conflict detection. If there is any conflict between values of some fields, the agents should move down in the abstraction level by one step with the help of background structure. It is worth mentioning that each field is related to the contexts when it has a child field related to those contexts or if it is a leaf field and the domain expert assigned it to the proper contexts. This structure of the background part implies that moving down in the abstraction level to find the desired fields needs to get coupled with context projection by the active context because all of the children fields may not be relevant to the active context. The children fields are then analyzed to find conflicts between them and the moving down in the abstraction level continues for only conflicting fields until the root of the conflict gets detected.

The final stage of the sharing strategy is conflict resolution. There have been several proposed models for this task, but they are mainly discussed in the human communication domains. We selected a model called Interest, Right, Power introduced by Furlong [42] for conflict resolution. This model has three layers. The innermost layer is the interest layer. The process of conflict resolution in this layer, concerns interests or desires and fears of the participants. Agents should begin the conflict resolution step, trying to resolve all they can in this layer. The outcome of this step if it results in a resolution of conflicts is a win/win situation. Each one of the participants, gain at least a portion of his/her interests. Some methods according to this layer are distributed problem solving, mediation and brainstorming.

If agents can't find a way to agree upon a point that resolves their conflicts, it is time to move to the upper level which is the Rights level. The main part of this layer is that it takes into account the rights and laws of the participants and it usually involves a superior role which decides about whose rights should get de-legitimized. Some proposed types of rights that should be considered here are laws, statues, conventions, past practices, policies and contracts. Because of the superiority nature of the decider role, the conflict resolution in this layers ends in favor of a party and the other party loses, so it makes a win/loss situation at the end. Some examples of processes in this layer are litigation, tribunal decisions and neutral evaluations.

In the highest layer, participants try to convince others by force. Usually all parties use all possible resources at their disposal to act on a way to make other parties lose. Because of the competitive nature of this layer, each party in this task tries to sacrifice some of his/her interest in order not to lose most of the interests. This makes it to generate a lose/lose situation. Parties try to minimize their loss by forcing the other parties to lose more. Some examples according to this layer are threatening, intimidation, physical force and strikes.

The conflict resolution method that we used in this method was inspired from the interest/right/power (I/R/P) framework. Some conflicts can resolve in the interest layer when in the middle of sharing mental model step, some missing or unknown values are getting filled that makes the agent to change its mind after receiving this new type of data which was unknown before. This is a win/win situation because all agents involved in the sharing process gain a little more perspective from the world or other members of the team, and the conflict gets resolved as a side effect of the sharing stage. This can get mapped to the interest layer of the I/R/P framework.

There are some situations that after moving down the background hierarchy to its deepest level, the conflicts still exist. This is the time to move to the next layer, Rights layer. The method we used to mimic the properties of rights layer is based on the judgments of a superior agent called coach. The coach agent has some judgment principles that are application sensitive. It asks for the projected mental models according to the active context of parties and evaluates each one of them with respect to its principles.

The power layer introduced in I/R/P framework is not applicable in our architecture because it is referring to a situation that is meaningful in human's societies where people use their power to dominate other possibly weak persons. Figure 3 presents the sharing process that empowered with semantic movement.

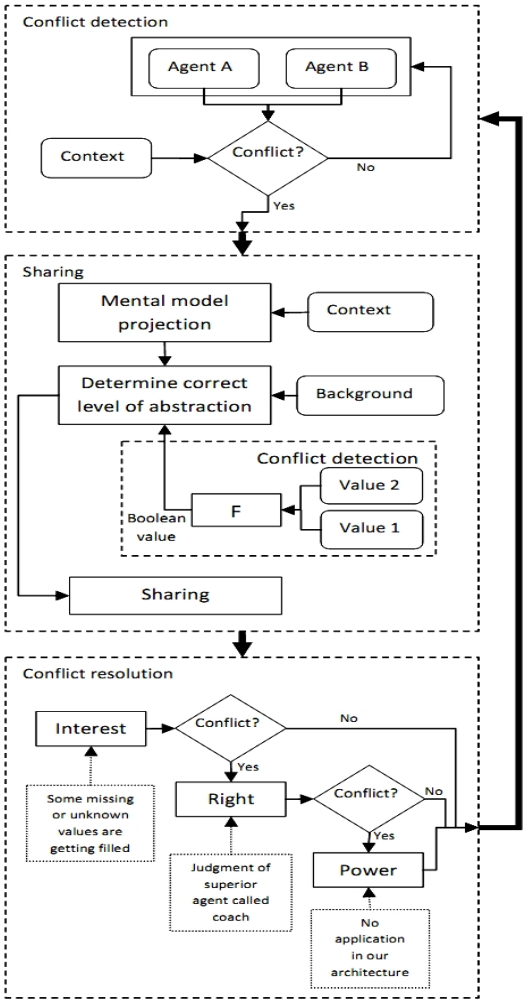

Figure 3. Sharing strategy empowered with semantic movement

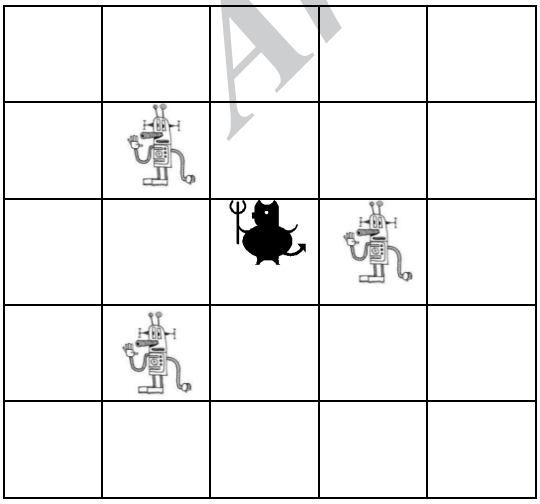

Figure 4. A sample situation in test environment

9. EXPERIMENT ENVIRONMENT

We need a complex multi-agent environment with multiple contexts in which data and information sharing is possible and beneficial. In lots of domains in multi-agent systems, using a form of data sharing strategy is essential. In some domains with complex environments and limited sensing powers for agents, each agent needs to have a complete and updated knowledge about the world and about the situation of other agents. Making a complete and updated world model only pays off the efforts of creating it by intensive communication between agents, when it is coupled with a method to use this world model for coordination and cooperation and generally acting on the environment. The issues of action selection, coordination and cooperation and all other action related problems are not in the domain of this research. We only presented architecture for sharing mental models and resolving possible conflicts, and no action execution is involved in our model. The experiment environment should be limited to gathering data and then sharing mental models. We worked on a test bench to practically implement and use our architecture for sharing mental models. This test bench has lots of uncertainties that make the need of data sharing for agents to correctly reason about the state of the environment. It also has different contexts and mental models are projected and shared in each of the contexts.

The experiment environment we developed is a multi-context version of the famous wumpus world game but we changed it in different ways. There are four types of individuals in this game. The first type of individual is the agent. The agents are the ones who can move in the map and sense four different senses that are smells, glows, breezes and sounds. The other individual is a wumpus that eats the agents if they move in their cell. They also make their adjacent cells to smell and sound. The next item in the map is a hole. When there is a hole in a cell it kills all agents that move inside that cell. It also makes the neighbor cells to have a glow and breeze. Some cells also contain gold pieces. When an agent moves in a cell with gold, it is rewarded by taking that gold for itself. Gold in a cell makes the adjacent cells to have sound and glow in them.

To make the test environment suitable for sharing purposes we decided to have three types of agents in a way that they need to share their knowledge about the world map to make a clear and unambiguous world model. A type of agent has the ability to sense smells and glows. The other one senses breezes and smells, and the last one senses breezes and hears sounds. In this environment with no sharing a single agent can't be sure about the state of a cell whether it has a wumpus in it or it is a hole or it has gold. But if they share their mental models each of them can reason with more confidence

about the situation of a cell. We also made the sensing of agent faulty so that with a configurable probability an agent can't sense correctly. For example a hearing agent in a fraction of times can't hear anything although there is a sound in that cell.

When an agent moves in a cell it tries to sense the environment based on its sensing capabilities. If an agent doesn't have the sensing power for some features of the environment it assumes that it doesn't know about that feature. If it has the sensing power for that feature and can't sense the presence of that feature it assumes that the feature is not present in that cell and finally if it can sense that feature it assumes that it is present. Although with the faulty sensors for agents the last two possibilities are not always trustable and in fact an agent can be wrong about a feature that has the capability of sensing it.

10. SAMPLE EXPERIMENT SCENARIOS

To show how the architecture for sharing mental models can help the agents to make a rather complete and accurate model of the game map we used our architecture in a few scenarios. In Figure 4, we have a wumpus and three agents around it. The neighbor cells are smelly and noisy and the three agents have sensors for breeze and glow detection and smelling and hearing. In this particular snapshot of the world, one agent senses that there is a sound in its cell. The other agent senses that there is a smell in its cell and the last agent senses that there is no breeze in its cell. Without any mental model sharing, the agent who hears a sound can only reason that there is a gold or wumpus in one or more cells of the eight adjacent cells. By sharing mental models between three agents in the Figure 4 each agent knows that the surrounding cells of the wumpus cell have sound and smell. This implies that the cell in the center probably has a wumpus in it according to the neighbor cells and the shared information of the agents. In Figure 5 a situation is shown that agents might reason something incorrect even after the sharing of mental models. The cells around the wumpus are smelly and noisy. The cells around the hole are windy and shiny. And the cells in the middle that are highlighted are smelly, shiny, windy and noisy at the same time. Now imagine that we have an agent with sensor for glow detection and hearing agent in the highlighted area. After sharing, they probably guess that one of the neighbor cells have a gold piece in it. The reason is that there is no information about the presence of smell in these cells because there is no agent in that area that can sense the smell. This example shows that although mental model sharing can help in generating a more accurate map of the world, it is not always enough if there is some information missing.

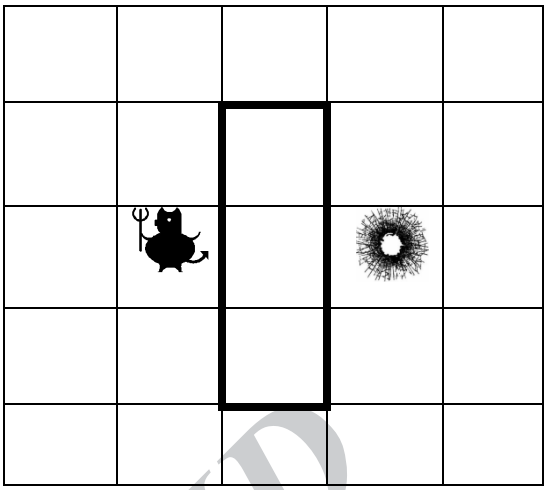

Figure 5. A sample misleading scenario

In Figure 6 a situation is shown that shows the importance of navigation time in the accuracy of the generated map. The more we give time to the agents to move around the map, the more they consider a situation when there are only three agents in the map (the agents are that bolder). They sense glow and sound in their cells and their neighbors are gathered and they can decide about the possibility of the presence of different entities with more confidence. After sharing they probably guess that there is a hole in the high-lighted cell at the center which is wrong. Now reconsider the example with the fourth agent (the brighter agent) contributing to the sharing process. The fourth agent doesn't sense any wind or glow. This, rules out the possibility of having a hole in the highlighted area. It is trivial that by giving time to agents to move around the map, the more knowledge about cells and their neighbors are gathered and they can decide about the possibility of the presence of different entities with more confidence.

11. RESULT

Test environment was introduced in the last section. The agents in this environment operate in two steps. In the first step, they move around in the map and collect sensory data for each visited cell. In the second step, they share their information about the map to resolve the destructive conflicts. In data gathering step, each agent can either silently move (no sharing strategy) and collect sensory data or ask other agents for complementary information (sharing strategy) about suspicious cells. In the sharing step, the agents can behave in 3 different ways. The first option is to use our propounded architecture to share the mental models and

resolve conflicts with only the agents that it has destructive conflicts with, in that context (partial sharing strategy). Another option for agents is to use our architecture and share their mental models with all other agents in the system (complete sharing strategy). The last option is that the agents don't share their information with each other and only a simple voting mechanism is applied for deciding the final state of each cell (no sharing strategy). There are five different scenarios with different strategies for the two steps. These five scenarios are:

1. *Scenario-n-p*: in this scenario our agents collect data without communication in first step (no-sharing strategy) and use our mental model and algorithm in second step (partial sharing strategy). This scenario presents capabilities of our model and sharing algorithm.
2. *Scenario-n-c*: in this scenario our agents don't communicate with each other in first step (no sharing strategy) but in second step they use complete sharing strategy to share their mental models. This scenario shows best performance in this environment since our agents have complete and compatible knowledge.
3. *Scenario-n-n*: in this scenario agents don't have any communication in both steps.
4. *Scenario-s-p*: this scenario use sharing strategy to move around the map in the first step and partial sharing strategy to share their mental models in the second step.
5. *Scenario-s-n*: in this scenario agents use sharing strategy in first step but agents do not share their mental models in the second step.

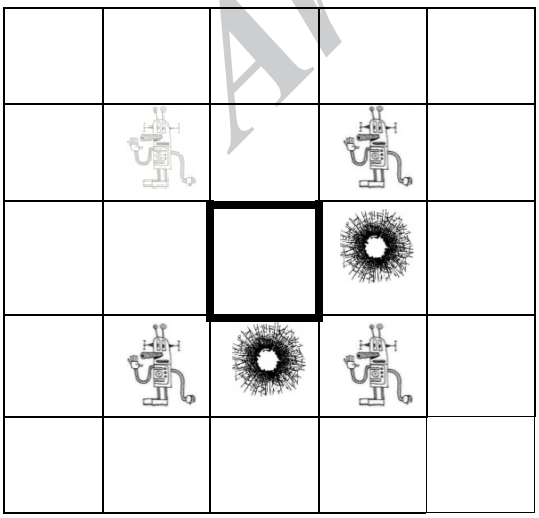

Figure 6. A sample scenario with conflicts

Agents try to participate in guessing the right game map and this map is compared with the world map in the game. In those scenarios that agents share their mental models in the second step, agents use the proposed sharing strategy to reach agreement and select appropriate state for cells. But in other scenarios, a broker agent is used to poll the opinion of agents for each cell's state. An agent has no opinion about a cell's state if and only if the calculated probability of cell's state for each state is below a threshold. The system can't have any opinion about the state of a cell if and only if no agent has any opinion about that cell's state. We use three measurements to compare these scenarios, recall, precision and total message length. Recall is the fraction of cells that are not unknown over the number of cells in the world map. Precision is the fraction of cells that are predicted correctly over of the number of predicted cells. Total message length is the sum of all message lengths that are passed between agents in the game.

Figure 7 presents the precision of five scenarios and their comparison. Vertical axis represents precision and horizontal axis represents the time of movement in environment. As Figure 7 suggests, the precision of our proposed method (scenario-n-p) in the 0.002 second time step is about 89% and the performance of the scenario with the complete information sharing (scenario-n-c) is about 89.3%. All other methods have noticeable lower precision than these two methods. As the time passes to 50 seconds, scenario-n-p and scenario-n-c both have 94% precision and other methods have precisions lower than 90%. Since agents only make a decision about a cell's state if they have a strong confidence in their answers, the precision of all scenarios are acceptable. Since the calculated probability of the cells' state in scenario-n-n for all cells is under the assigned threshold, precision in this scenario is 0. That's because agents don't share any information so they don't have sufficient information for deciding the state of the cells. Scenario-n-c has the highest precision in guessing cells' states because the agents in this scenario have complete and compatible knowledge about cells. The next best method after scenario-n-c is our introduced method (scenario-n-p). In scenario-n-c, all agents share their information about every cell that they don't agree upon its state to resolve any possible conflicts. In our method (scenario n-p) the agents try to share their information about the cells that they don't agree upon their state (destructive conflicts). Also in our method, only the agents that have different opinions about a cell's state are participating in the sharing process but in the best scenario all agents take a part in the sharing process. Between scenario-s-p and scenario-s-n, the higher precision of scenario-s-p is due to the sharing of mental models in the step 2 of the work in scenario-s-p.

In scenario-s-n agents share their information only in data gathering step. The time needed for sharing is taken from the overall time for data gathering step so agents have less time for moving around the game map and in map prediction step, they don't share their mental model. In scenario-s-p agents share their information in both of the steps. Since agents share information in data gathering step, they have less time for moving around the map and gathering sensory data. Due to less gathered data in scenario-s-p in comparison with scenario-n-p, the performance of scenario-n-p is better.

The recall of the answers in each scenario is presented in Figure 8. In this figure, vertical axis represents recall and horizontal axis represents the time of movement in environment (duration of data gathering step). In the first time step the recall of scenario-n-p is about 84% which is 9% lower than the scenario-n-c. The other two scenarios have much less recall in this time step (31% and 0%). In 0.1 second data gathering time the scenario-n-p and scenario-n-c have a recall of nearly 100%, but the scenario-s-p's recall is about 76% and the scenario-s-n has a recall of about 5% and the scenario-n-n has 0% recall. Scenario-s-p reaches to the recall of 100% in the 10 seconds data gathering time and scenario-s-n's recall is 90% in 50 seconds data gathering time. Scenario-n-c has highest recall as well as the highest precision. As mentioned before recall is the number of predicted cells over the number of cells in the map. Therefore, if the number of predicted cells in one scenario gets more, recall is increased. The prediction of a cell's state is based on the information that agents gathered and received during sharing process. In scenario-n-c agents have the most information about the map because of the complete information sharing in map prediction step. That is why this scenario has the best recall among other scenarios. Scenario-s-n is not good at recall because it misses a notable amount of time for gathering data and it also bypasses the mental model sharing process. Our proposed method (scenario-n-p) outperforms scenario-s-p according to recall. Because the sharing of the mental models done in the scenario-s-p takes the time for moving around the map and gathering data.

Figure 9 shows the multiplication of precision and recall for each scenario. This figure shows the percent of cells that was predicted correctly. As expected, scenario n-c has the best result and our method is standing in the second place with a very little difference from the best scenario.

Figure 10 presents the total number of bytes in the messages passed between agents in all scenarios. As it is easily noticeable, the scenario-n-c has the highest number of bytes in its communication messages and the difference between this scenario and others is huge especially in 0.002 second time step. Scenario-n-c transfers a total number of nearly 1,304,000 bytes and the scenario-n-p transfers 176,000 bytes for sharing purposes. This huge difference is the result of sharing all gathered information between all agents in the scenario-n-c and the directed, optimized sharing done in scenarios with partial sharing like our method (scenario n-p). Although we didn't take into account the effect of

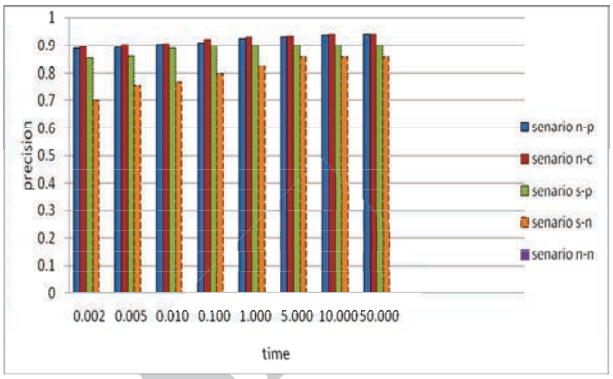

Figure 7. Precision of five scenarios

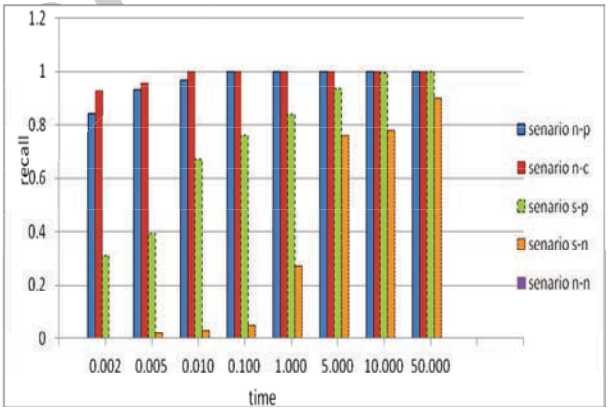

Figure 8. Recall of answer in each scenario

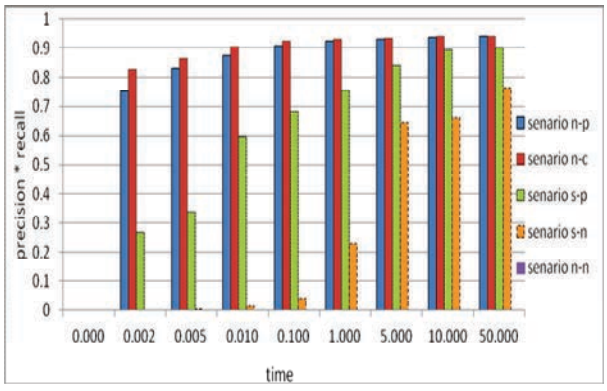

Figure 9. The multiplication of precision and recall for each scenario

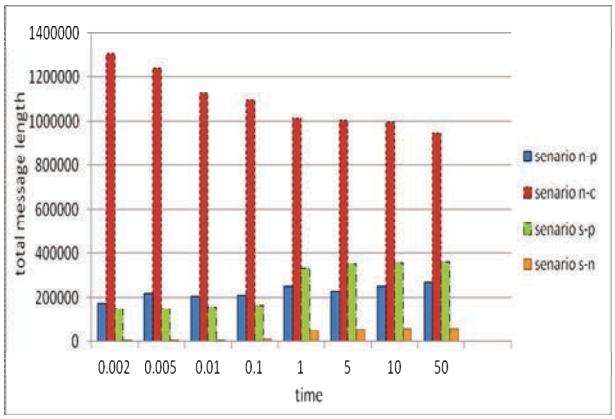

Figure 10. the total number of bytes in the messages passed between agents in all scenarios

communication time and cost on our test environment, but in real applications it has a very important effect on the effectiveness of the system. In lots of domains, communication is a slow, resource consuming task and should be used wisely in order to keep the performance of the system at its peak. Using it unwisely may even result in network congestion. Our method tries to optimize the use of communication facilities by applying the partial sharing strategy. It is nearly as good as the complete sharing strategy according to precision and recall but with very less message lengths.

12. CONCLUSION

In this paper, a new three layered architecture for sharing mental models has been introduced. This architecture is specially designed to help multi-agent system designers to add sharing capabilities between agents in their systems. It also embodies a new sharing strategy that can detect and resolve any harmful conflicts between mental models. This sharing strategy is based on contexts' information and a shared ontology called background.

We have applied our architecture on a complex test environment that no agent can perform well individually and without sharing. We equipped our agents with a context aware architecture and applying sharing strategy empowered with semantic movement. Comparing our proposed scenario with other sharing methods, the results were promising. The result of our method is close to original map while the total messages passed between agents are optimized.

Our sharing method is dependent on information in context layer and background layer. Therefore, any incompetency in these layers affects the sharing accuracy. Moreover, another limitation of our method is the existence of right layer in conflict resolution step. We use a superior agent in this layer to judge between conflicting agents if it is necessary. In ad hoc teams with large number of agents, this superior agent can become a bottleneck. One solution is to cluster agents based on mental or physical neighborhood and use a superior agent for each cluster.

We believe that our architecture can also perform well in application domains that have real-time constraints. That is because our architecture uses projected mental models according to the current context which makes other steps like reasoning, planning and decision making faster than before.

This work can be improved by adding learning mechanisms for automatic context formation or background information therefore a domain expert is not needed anymore. Also, our sharing strategy can be augmented with trust related methods to enable agents to treat agents differently.

13. REFERENCES

1. P. Johnson-Laird, "Mental models: Towards a cognitive science of language, inference and consciousness", Harvard University Press, (1986).
2. M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," ***International Journal of Ad Hoc and Ubiquitous Computing***, Vol. 2, (2007), 263-277.
3. Y. Zhong, "Study on Cognitive Decision Support Based on Learning and Improvement of Mental Models," ***Computing, Communication, Control, and Management, CCCM'08,*** (2008), 490-494.
4. C. Jonker and J. Treur, "A dynamic perspective on an agent's mental states and interaction with its environment**",** ***The first international joint conference on Autonomous agents and multiagent systems: part 2***, (2002), 865-872.
5. Xiaocong Fan and John Yen," Modeling and simulating human teamwork behaviors using intelligent agents," ***Physics of Life Reviews***, Vol.1, (2004), 173-201.
6. Yu Zhang," Role-Based Shared Mental Models," ***Collaborative Technologies and Systems***, (2008), 424-431.
7. John Yen, Xiaocong Fan, Shuang Sun, Timothy Hanratty and John Dumer," Agents with shared mental models for enhancing team decision makings," ***Decision Support Systems***, Vol.41, (2006), 634-653.
8. Xiaocong Fan, Po-Chun Chen and John Yen," Learning HMM-based cognitive load models for supporting human-agent teamwork", ***Cognitive Systems Research***, Vol.11, (2010), 108-119.
9. Xiaocong Fan and John Yen," Realistic Cognitive Load Modeling for Enhancing Shared Mental Models in Human-Agent Collaboration," AAMAS'07 May 14–18, 2007, Honolulu, Hawai'i, USA, (2007), 60.
10. Daniel Fuller, Brian Magerko," Shared Mental Models in Improvisational Performance," Proceedings of the Intelligent Narrative Technologies III Workshop, ACM, (2010), 15.
11. J. Hendler, "Agents and the semantic web," ***Intelligent Systems, IEEE***, Vol. 16, (2001), 30-37.
12. N. Gibbins, S. Harris and N. Shadbolt, "Agent-based semantic

web services," ***Web Semantics: Science, Services and Agents on the World Wide Web,*** Vol. 1, (2004) 141-154.

13. D. S. Coalition, A. Ankolekar, M. Burstein, and J. R. Hobbs, "DAML-S: Web service description for the semantic Web," ***The Semantic Web ISWC*****,** (2002), 348-363.
14. S. McIlraith and T. C. Son, "Adapting golog for composition of semantic web services," ***principles of knowledge representation and reasoning-international conference***, (2002), 482-496.
15. S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," ***11th international conference on World Wide Web***, (2002), 77-88.
16. P. Buhler and J. M. Vidal, "Semantic web services as agent behaviors," ***Agentcities: Challenges in Open Agent Environments,*** Citeseer, (2003), 25–31.
17. D. Greenwood and M. Calisti, "Engineering web service-agent integration," ***Systems, Man and Cybernetics, 2004 IEEE International Conference,*** Vol. 2, (2004), 1918-1925.
18. E. M. Maximilien and M. P. Singh, "Agent-based architecture for autonomic web service selection," Proc. of the 1st International Workshop on Web Services and Agent Based Engineering, Sydney, Australia, (2003).
19. E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web services selection," ***Internet Computing, IEEE,*** Vol. 8, (2004) 84-93.
20. Z. Maamar, S. K. Mostefaoui and H. Yahyaoui, "Toward an agent-based and context-oriented approach for Web services composition," ***IEEE Transactions on Knowledge and Data Engineering,*** (2004), 686-697.
21. C. Jonker, M. van Riemsdijk and B. Vermeulen, "Shared Mental Models," ***Coordination, Organizations, Institutions, and Norms in Agent Systems VI,*** Vol. 1, (2011), 32-151.
22. Kaivan Kamali, Xiaocong Fan and John Yen," Multiparty Proactive Communication: A Perspective for Evolving Shared Mental Models," American Association for Artificial Intelligence, (2006), 685.
23. F. Schmitt, J. Cassens, M. Kindsmller and M. Herczeg, "Mental models of ambient systems: a modular research framework," Modeling and Using Context, (2011), 278-291.
24. E. Phillips, S. Ososky, J. Grove and F. Jentsch, "From Tools to Teammates," ***Proceedings of the Human Factors and Ergonomics Society Annual Meeting,*** (2011), 1491-1495.
25. Brigitte Burgemeestre, Jianwei Liu, Joris Hulstijn and Yao-Hua Tan, "Early requirements engineering for e-customs decision support: Assessing overlap in mental models," ***Proceedings of CAiSE Forum***, (2009), 31-36.
26. Kennedy, W.G and Trafton, J.G, "Using Simulations to Model Shared Mental Models," ***Proceedings of the Eighth International Conference on Cognitive Modeling,*** (2007), 253-254.
27. M. N. Huhns, "Agents as Web services," ***Internet Computing, IEEE,*** Vol. 6, (2002), 93-95.
28. K. Sycara, M. Paolucci, J. Soudry and N. Srinivasan, "Dynamic discovery and coordination of agent-based semantic web services," ***Internet Computing, IEEE,*** Vol. 8, (2004), 66-73.
29. Hong, J., Suh, E. and Kim, S., "Context-Aware Systems: A Literature Review and Classification." ***Expert Systems with Applications***, Vol. 36, (2009), 8509-8522
30. J. Payton, "Simplifying Context-Aware Agent Coordination Using Context-Sensitive Data Structures," ***DTIC Document***, (2004).
31. K. Arabshian and H. Schulzrinne, "Distributed context-aware agent architecture for global service discovery," The Second International Workshop on Semantic Web Technology for Ubiquitous and Mobile Applications (2006).
32. N. M. Sadeh, T. C. Chan, L. Van, O. Kwon, and K. Takizawa, "Creating an open agent environment for context-aware m-commerce," ***Agentcities: Challenges in Open Agent Environments,*** Vol. 70, (2003).
33. M. Hattori, K. Cho, A. Ohsuga, and M. Isshiki, "Context-aware agent platform in ubiquitous environments and its verification tests," ***Systems and Computers in Japan,*** (2003), 547-552.
34. H. J. Lee, J. E. Park, E. J. Ko, and J. W. Lee, "An agent-based context-aware system on handheld computers," ***International Conference on Consumer Electronics, 2006. ICCE'06. 2006 Digest of Technical Papers,*** (2006), 229-230.
35. W. Chun-Dong and W. Xiu-Feng, "Multi-agent Based Architecture of Context-aware Systems," ***International Conference on Multimedia and Ubiquitous Engineering, 2007. MUE'07,*** (2007), 615-619.
36. W. S. Wong, H. Aghvami and S. J. Wolak, "Context-aware personal assistant agent multi-agent system," ***IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, 2008. PIMRC,*** (2008), 1-4.
37. B. Y. Lim, A. K. Dey and D. Avrahami, "Why and why not explanations improve the intelligibility of context-aware intelligent systems," ***27th international conference on Human factors in computing systems,*** (2009), 2119-2128.
38. B. Lim, "Improving Understanding, Trust, and Control with Intelligibility in Context-Aware Applications," Human-Computer Interaction, (2011).
39. O. Kwon, S. Choi and G. Park, "NAMA: a context-aware multi-agent based web service approach to proactive need identification for personalized reminder systems," ***Expert Systems with Applications,*** Vol. 29, (2005), 17-32.
40. H. Harroud and A. Karmouch, "A policy based context-aware agent framework to support users mobility," Telecommunications, 2005. Advanced industrial conference on telecommunications /service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete, (2005), 177-182.
41. Burkle, W. Muller, U. Pfirrmann, M. Schenk, N. Dimakis, J. Soldatos, and L. Polymenakos, "An agent-based architecture for context-aware services supporting human interaction," International Conference on Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM, (2006), 146-152.
42. G. T. Furlong, "The conflict resolution toolbox: models & maps for analyzing, diagnosing and resolving conflict" Wiley, (2005).

A Context-aware Architecture for Mental Model Sharing through Semantic Movement in Intelligent Agents

S. Salehi [a], F. Taghiyareh [a], M. Saffar [a], K. Badie [b]

[a] *Department of ECE, University of Tehran, Postal Code: 14395-515, Tehran, Iran*
[b] *Department of IT, Iran Telecommunication Research Centre, Tehran, Iran*

P A P E R I N F O

Paper history:
Received 12 October 2011
Received in revised form 05 February 2012
Accepted 17 May 2012

Keywords:
Autonomous Agent
Semantic Movement
Shared Mental Model
Mental Model
Context-aware
Architecture
Intelligent Agent

چکیده

مطالعات اخیر در سیستم‌های چند عامله توجه ویژه‌ای را به الگوی طراحی عامل‌های هوشمند با استفاده از مفهوم‌های الهام گرفته شده از انسان‌ها داشته است. یکی از مفهوم‌های شناختی مهمی که در بسیاری از روش‌های اخیر در سیستم‌های چند عامله نقش محوری ایفا می‌کند،مفهوم مدل ذهنی مشترک است. در این مقاله ما یک معماری برای اشتراک مدل ذهنی عامل‌ها بر اساس یک مفهوم جدید با عنوان حرکت معنایی معرفی می‌کنیم. این معماری با الهام از مدل ذهنی انسان‌ها طراحی شده و می‌تواند به صورت هم زمان توسط عامل‌ها در چندین هم‌بافت مورد استفاده قرار گیرد. این معماری از حرکت معنایی به عنوان یک ساز و کار رفع تضاد استفاده می‌کند. حرکت معنایی نوعی حرکت در حالت‌های ذهنی افراد با هدف برطرف کردن تضادهای مضر موجود در مدل‌های ذهنی است. همچنین برای ارزیابی این معماری، ما از مجموعه‌ای از سناریوهای پیچیده استفاده کردیم که در آن‌ها بدون فرآیند اشتراک گذاری نمی‌توان به جواب مناسب رسید. معماری ما نسبت به روش‌های جایگزین اشتراک گذاری عملکرد بهتری را نشان داد. ما اعتقاد داریم که معماری ارائه شده می‌تواند عامل‌ها را برای ایجاد رفتارهایی سازگارتر بین خودشان توانمند کند. همچنین این معماری می‌تواند برای محیط‌های مذاکره بین عامل‌ها مناسب باشد.

doi: *10.5829/idosi.ije.2012.25.03b.10*