

AGORA Multi-agent Architecture for Implementing Virtual Enterprises

Sobah Abbas Petersen¹, Jinghai Rao¹, Mihhail Matskin²

¹*Department of Computer and Information Sciences,
Norwegian University of Science and Technology, 7491 Trondheim, Norway
Email: {sap, Jinghai}@idi.ntnu.no*

²*Department of Microelectronics and Information Technology,
Royal Institute of Technology, SE-164 40 Kista, Sweden
Email: misha@imit.kth.se*

Abstract

The formation of a Virtual Enterprise and the selection of its partners is an important process in the lifecycle of a Virtual Enterprise. In this paper, we present the Virtual Enterprise formation process as an Agent Interaction Protocol and an approach to its implementation. We have focussed on the selection of partners within the formation process in order to understand these interactions and the contents of the messages that are exchanged between the agents. Based on this, we propose the contents of the knowledge base of an agent in the VE. We also describe how the AGORA multi-agent architecture can be used to support the formation of a Virtual Enterprise.

1. Introduction

The formation of a Virtual Enterprise (VE) is an important phase in the lifecycle of a VE. The selection of the partners that will do the work in the VE is central to the formation phase and is one of the success factors for a VE, [5]. There are several definitions of a VE; we consider the project-based view of a VE, [4], where the project team constitutes the VE and the members of the team are the partners of the VE. The partners of a VE, who may be human beings, organisations and/or software agents, collaborate to achieve the goals of the VE.

In this paper, we present a multi-agent architecture, AGORA, to support the formation of VEs. We believe that software agents, (hereafter referred to as agents), are a suitable means of representing the partners of a VE. One important reason is that by delegating the agents to conduct the negotiation on behalf of the partners, the partners could then have the time to do the actual work required in the current VE. The objective of this work is to support the decision making process of the partners and not to replace the human user.

We have attempted to represent the VE formation process as best as we can by studying a number of industrial situations. We believe that the interactions between the agents during the VE formation process can be described using Agent Interaction Protocols (AIP). In our work, we have focussed on the selection of partners within the formation process in order to understand these interactions and the contents of the messages that are exchanged between the agents. Based on this, we propose the contents of the knowledge base of an agent in the VE.

AGORA, based on the idea of a cooperative node, provides the necessary infrastructure to represent a VE scenario. We describe the AGORA multi-agent architecture as well as the architecture for a single agent to represent the partners in a VE. The VE is described using an agent-based model and the VE formation scenario is illustrated using a simple hypothetical example.

The rest of this paper is organised as follows: Section 2 describes an agent-based model of a VE, Section 3 describes the VE formation process, Section 4 presents the model of a single agent, Section 5 describes the AGORA architecture, Section 6 illustrates how AGORA can be used to support VE formation using an example, Section 7 discusses the literature related to this work and Section 8 discusses the conclusions.

2. Model of a Virtual Enterprise

We have developed an agent-based Enterprise Model of a VE by analysing the entities in a VE, their relationships and how they can be used in an agent context. In our model, the VE has a *goal* which is achieved by a set of *activities* which are performed by a set of *roles*. The agents that fill these roles are the members of the VE and the agents are selected on the basis of how well they meet the *requirements* for the roles. For our work, we assume that the goals, the activities, the roles and their requirements are available before the VE is announced and that this information is used in the VE announcement. For a detailed description of this model, see [13].

The agents in a VE can be classified as *VE Initiator* (who may also be the customer), who takes the initiative to form the VE and *VE Partner* (who may also be the VE Initiator), who are the entities that form the VE. A VE Partner evolves from someone that is interested in becoming a part of the VE to someone who is actually a part of the VE. We use the terms *Interested Partner*, one that is interested in becoming a part of the VE and submits a bid for the work, and *Potential Partner*, one that is considered for the VE and a contract is negotiated.

3. Virtual Enterprise Formation

VEs have a limited lifetime; thus they need to be formed very quickly in order to meet the deadlines of the goals and there is a need to form them often. During the formation phase, the Interested Partners compete and negotiate to become partners of the VE. Thus, an important part of the formation is the selection of partners, who are selected based on their ability to fulfil the requirements of the VE.

The formation of a VE consists of announcing the VE, accepting and evaluating the bids to select the partners of the VE and then awarding the contract to the selected partners. The process of selecting the partners and the selection criteria that is used varies from VE to VE. This is dependent on the type of work that the VE needs to do as well as the kinds of negotiations that need to be conducted. We have analysed a number of VE partner selection processes in a series of case studies in order to understand this process and to be able to develop an agent-based support for this process. The details of this analysis are available from [14].

3.1. Agent Interaction Protocol

The AIP corresponding to the formation of a VE is shown in Figure 1. We have used Agent UML, [3], to illustrate the AIP. The VE Initiator announces the VE and awaits bids from the Interested Partners. The VE announcement contains the main goal of the VE and the roles that are required for the VE. Agents that are interested will request for the details of the roles that they wish to bid for and propose a bid. Figure 1 does not show the detailed matching process, rather the main steps in the selection process.

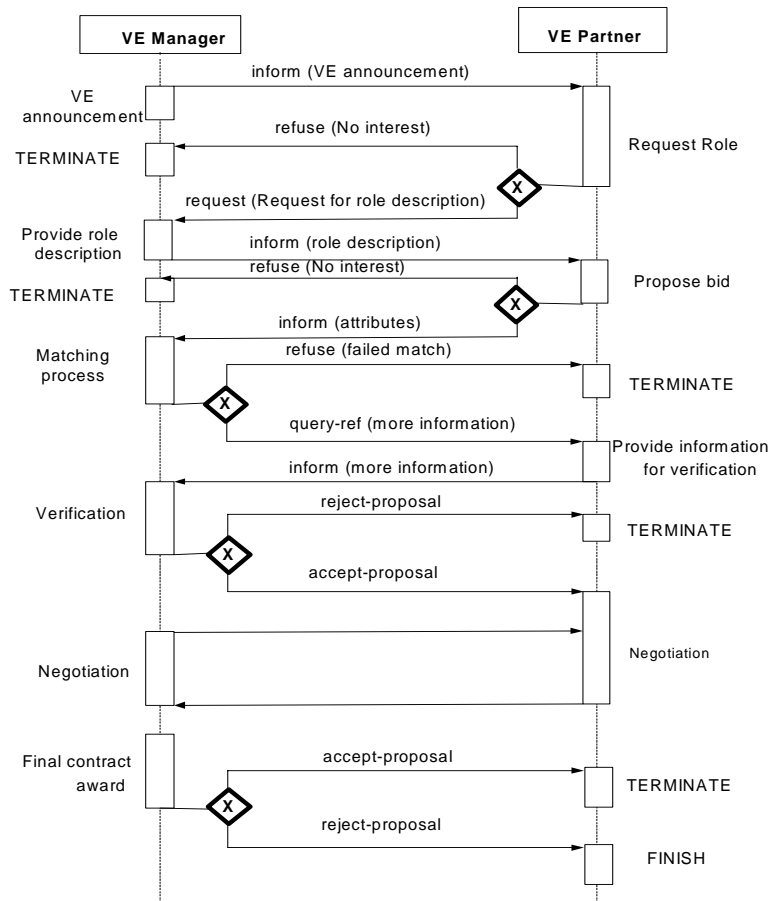


Figure 1: Agent Interaction Protocol for VE Formation

3.2. VE Partner Selection Process

An overview of the process of selecting partners for a VE is shown in Figure 2. The first step is the alignment of the goals of the Interested Partners with the goals of the VE.

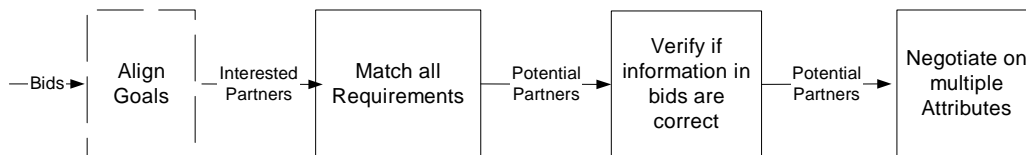


Figure 2: VE Selection Process

The next step is matching the Interested Partners to the requirements of the roles. The requirements are structured into skills and capabilities, availability and cost requirements. Thus, the matching process consists of several steps, where, for example, the availability and the price are matched only if the skills and competencies fulfil the requirements.

The next step is the verification of the information provided in the bids. This is to ensure that the Potential Partner indeed has the experience and the means of delivering to the VE as claimed. In reality, this is often conducted in the form of interviews and workshops. Once the verification is conducted, the VE Initiator and the Potential Partners (or the Potential Partners themselves) negotiate to agree upon the terms of the contract. This is a multi-attribute negotiation and multi-attribute utility theory and optimisation techniques can be used to support this process.

4. Model of an Agent

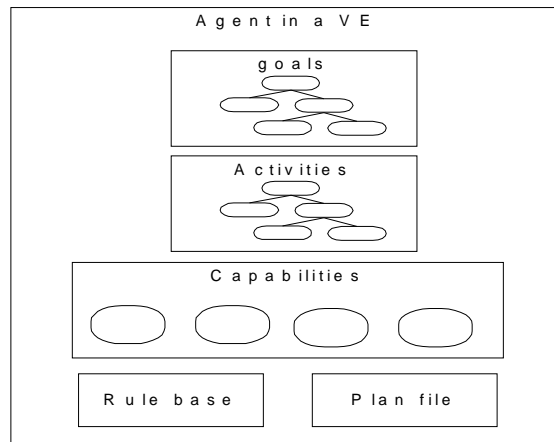


Figure 3: Agent's Knowledge Base

The basic components of an agent's knowledge base required to support the selection process was proposed in [14], see Figure 3. In this paper, we consider the components of the agent's knowledge base with respect to the information that is exchanged between the agents during the formation of the VE. A brief description of the components is given below:

- **Goals:** Goals are described by a set of attributes; the name of the goal, the product area that the VE intends to work on, the delivery date for the product, and the total amount of money that the VE can spend to achieve the goals. The goals have a simple tree structure.
- **Activities:** Activities are performed towards achieving one or more goals of the VE. They are described by a set of attributes; the name of the activity, when the activity is performed, the number of times an activity has been performed and names of agents with whom the activity was performed in the past. Activities also have a simple tree structure.
- **Capabilities:** Capabilities of an agent are a list of attributes, some of which may also be a set of attributes. They describe the requirements for a role and the attributes of an agent that are matched against the requirements for a role.

- **Rule Base:** The rule base of an agent contains the rules for making decisions such as the matching rules.
- **Plan:** An agent’s plan tells an agent what it should do at any point in time. It is based on the AIP and assists an agent in deciding what action to take next, based on the agent’s knowledge and the messages received by the agent.

In our approach, we consider the same agent architecture for both the agents representing the VE Initiator and the partners. This is because the VE Initiator does not always play the role of a “broker” only, but can also be a partner in the VE. During the formation process, the VE Initiator represents the VE and thus the information represented by the VE Initiator reflects the VE whereas the information represented by the VE Partners reflects that particular partner. Hence, the information that is represented by the goals, the activities and the capabilities of the VE Initiator and the VE Partners are slightly different and this is summarised in Table 1.

Table 1: Information represented by the Agent Model

Entity	VE Initiator	VE Partner
Goals	Goals of the VE	Goals of the partner
Activities	Activities that need to be performed to achieve the goals of the VE.	Set of experience of the partner.
Capabilities	Requirements (skills, time, costs, etc.) for the roles of the VE.	Work that the partner is capable of doing.

5. Forming Virtual Enterprises Using AGORA

In this section, we describe the AGORA multi-agent architecture and how it can be used to support VEs.

5.1. AGORA Multi-agent Architecture

AGORA is a multi-agent infrastructure which provides support for implementation of software agents and agent-based marketplaces, [9]. The central concept is that of an *Agora node* which is a cooperative node facilitating communication, coordination and negotiation among the agents. When an *Agora node* is created, 3 default agents are created and connected to the agora node automatically. They are the *Agora Manager* (for performing general management functions), *Coordinator* (for supporting a coherent behaviour between agents in the node) and *Negotiator* (for dealing with conflict resolution via negotiation). A standard *Agora Manager* implements general management functions such as the registration of agents and matchmaking. The coordinator and negotiator are agents that manage corresponding protocols. Some default functionalities are implemented for the Manager, Coordinator and Negotiator. However, these functionalities can be overwritten by the user if needed. In addition to *Agora nodes* and default agents, the system also has *registered agents*. In a general marketplace scenario, the registered agents can act as either buyers or sellers. They can communicate with each other directly or via the *Agora manager*.

The structure of a single agent, either a default agent or a registered agent is illustrated in Figure 4. An agent uses the *Message Proxy* and the *Log System* to interact with the outside world. It communicates with other agents using FIPA ACL [7] and the FIPA messages are sent and received through the *Message Proxy*. The *Log System* is the interaction channel between the agent and its owner, normally a human user. First, the user

is able to monitor the executing status of an agent through the Log System. This information can be presented either through a GUI log window or stored as a log file. The second function of the Log System is that the agent should be able to get instructions from the user at runtime. This is important for e-market applications because, in many cases, the final decision is made by the user and not by the software agent.

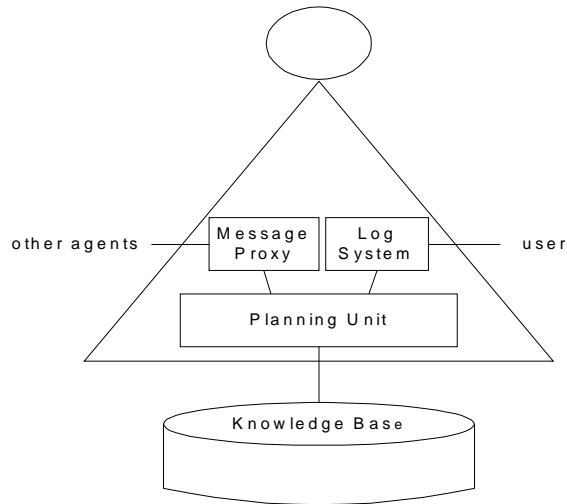


Figure 4: Structure of an Agent

We use a Prolog-based presentation for messages, facts and rules in the *Knowledge Base*, implemented using the XProlog system, [1]. In order to integrate the FIPA messages with the Knowledge Base, a *Compiler* between FIPA messages and Prolog clauses is implemented. Currently, the compiler just translates 7 FIPA message elements that are important for the decision making process: performative, ontology, sender, receiver, content, reply-with and in-reply-to. The last two are used to uniquely identify a message. This set of elements can be extended as required by the specific application.

The *Planning Unit* decides the agent's next action by a set of explicitly defined rules. In Agora, the plan is specified in a XML-based scripting language. Each step in the plan has an action to be performed and post-conditions. The action refers to an outgoing FIPA message or a method (function) written in Java or Prolog. Post-conditions are described as a reaction of the agent to a communicative act received from another agent.

The AGORA provides a default implementation for all its components, which include agent communication, planning and basic matching capabilities. Using the default implementation, the application developer just needs to specify the plan and insert rules into the Knowledge Base. The developer can overwrite the default implementation or add new components into the system due to AGORA's open architecture.

5.2. Virtual Enterprises in AGORA

The idea of using AGORA for supporting VEs was introduced in [12], where the VE Initiator and the partners of a VE could meet at an Agora node. A structure containing several Agora nodes can be created for the different activities throughout the lifecycle of a VE. The Agora nodes for VE formation are presented in Figure 5. It contains a general Agora node (VE Formation) for the complete process and five Agora nodes (VE

Announcement, Goal Alignment, Matching Requirements Verification and Contract Award) for each step in the VE formation process, (see Figure 2).

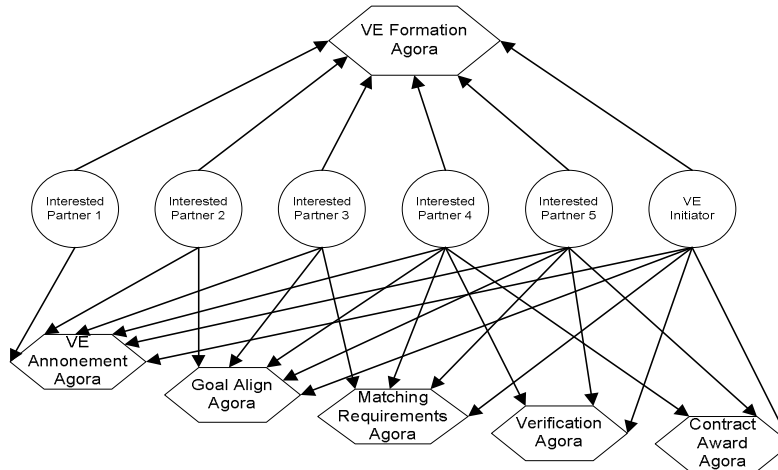


Figure 5: The AGORA Multi-agent Architecture

Each Agora node has a Manager, a Negotiator and a Coordinator. Registered Agents (Interested Partners) can be registered in more than one Agora node. It is also important to note that the number of Interested Partners can be equal or less at each subsequent node as the formation of the VE progresses. Each Agora node provides the right context for the specific step in the formation process, i.e. the right support and a meeting place for all the participants. Having a separate Agora node for each step ensures information security and privacy, e.g., an Interested Partner whose bid has been refused cannot register at the next Agora node. This ensures that unauthorised agents cannot access information passed between the VE Initiator and the Interested and Potential Partners.

6. Example

We will use a simple example to illustrate the VE formation process and the selection of VE partners by matching agents to roles in a VE. Consider a VE formed to design and create an Intranet for a company. The main goal of the VE, Create an Intranet, can be decomposed into two subgoals, Design Intranet and Create Intranet. The two subgoals are achieved by performing the activities, Design Intranet and Create Intranet. The two roles that are required for this VE are an Intranet Designer and a Webpage Developer. The VE Initiator is looking for two partners that meet the requirements for these roles. The requirements can be represented as a set of constraints. In the rest of this section, we will describe how the formation of this VE, as described by the AIP in Figure 1, is implemented in AGORA.

6.1. VE Announcement Agora node

The VE announcement consists of the main goal of the VE, (described by the attributes name, deadline and maximum cost) and the set of roles that need to be filled by the partners.

```
ve_announcement(
    goal(create_intranet, 280203,40000),
    roles([intranet_designer,web_program_developer])).
```

If an agent is interested in performing any of the roles, then it requests for more information on that specific role. The VE Initiator will then respond with the requirements for the requested roles(s). Table 2 shows the set of requirements and the matching conditions for the role Webpage Developer.

Table 2: VE Requirements and Matching Conditions

Requirements	Range & Matching Conditions
Skills	HTML, JAVA, XML
Min. no. of skills required	>=2
Experience	>=2 years
Availability	Start_date<010103, end_date=<280203, 80% of the time, Matching condition: computed no. of hours =<300
Cost per hour	<60
Performance rating	Range: 1..10, >=6
Commitment	Range: 1..10, >7

The Interested Partners return bids after they receive the requirements for the roles. In the bids, the partners fill their real values for each attribute in the requirement. Below is the bid containing the skills, submitted by agent “Programmer1” for the role of Webpage Developer:

```
bid_skill(programmer1,
          role(Webpage_developer),
          attributes([java,xml,html]),experience_by_year(3),
          performance_rating(7),commitment(8)).
```

6.2. Goal Alignment Agora node

A goal can be defined by a set of attributes. The partner’s goals are aligned with that of the VE if there is a goal in the VE’s goal structure that matches that of the partner’s. The matching is based on the attributes of the goal, where each attribute is matched using a set of matching conditions, similar to those defined for roles.

6.3. Matching Requirements Agora node

The requirements for the roles are structured into skills, availability and cost requirements. The matching process consists of matching first the skills, then the availability and finally the costs. The Interested Partners are requested to submit the part of the bid containing availability information only if the skills meet the requirements. And similarly, the cost information is required only if the Interested Partner meets the availability requirements. For example, the rule to match the skills is shown below:

```
match_skill(outgoing_msg(Receiver, InReplyTo,
                        Performative, ve, Content)):-
    incoming_msg(Receiver, InReplyTo, inform,
                ve, bid_skill(_, role(Role),
                              attributes(skills(Skills),
                              experience_by_year(Experience),
                              performance_rating(Performance),
                              commitment(Commitment))),
                role_descr(Role,_,requirements(
                              skills(RequiredSkills),
```

```

experience(RequiredExp),
performance_rating(RequiredPerf),
commitment(RequiredComm,_,_,_)),
member(RequiredSkills, Skills),
Experience >= RequiredExp,
Performance >= RequiredPerf,
!,
Performative = query-ref,
Content = availability.

match_skill(outgoing_msg(_,_,Performative,ve, Content)) :-
    Performative = refuse,
    Content = fail_to_match_skills.

```

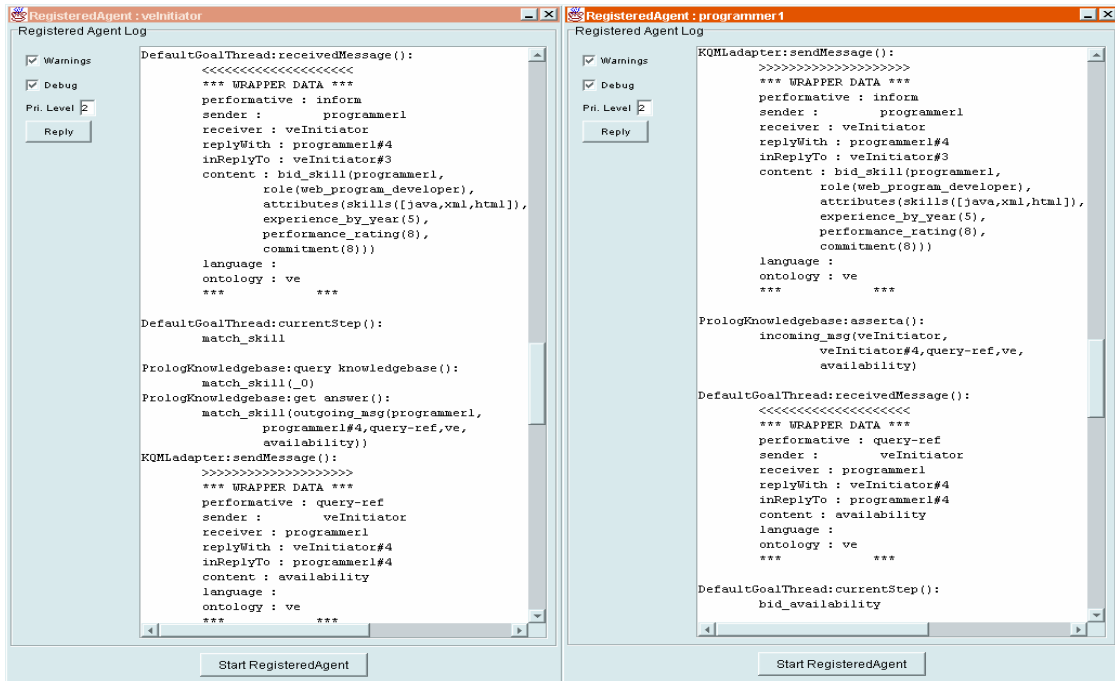


Figure 6: AGORA: Interaction between the VE Initiator and an Interested Partner

The output of the rule is a Prolog term that specifies the outgoing message. If the interested partner’s skills are matched, the VE Initiator requests for the availability information, otherwise, it rejects the interested partner.

The log windows of the VE Initiator agent (called `veInitiator`) and a VE partner agent (called `programmer1`) are shown in Figure 6. We can see the interaction between the two agents from these log windows. In addition, it shows the output (a prolog term) of an agent’s current action and the translation between the Prolog term and the message wrapper.

6.4. Verification *Agora* node

If the Interested Partner meets all the requirements for the role, s/he becomes a Potential Partner and the VE Initiator would now like to verify if s/he actually does have the experience claimed in the bid. Thus, the VE Initiator requests for the Potential Partner’s activities or experience structure. The Potential Partner can choose to submit the complete structure or the node(s) that are relevant to the desired activity. For example, if the Potential Partner bids for the role of Webpage Developer to perform the activity “Create

Intranet”, then the VE Initiator looks for a node in the Potential Partner’s experience structure that matches this activity.

6.5. Contract Award Agora node

The VE Initiator informs the Potential Partners whose bids are rejected and a contract is signed with the Potential Partners whose bids are accepted. Since AGORA provides a means for decision support, it is the human user (partner) that will make the final decision. Thus, the user will instruct the agent through a dialogue window (Log System).

6.6. Agent Plans

An agent’s plan consisting of all the communication exchange and protocols (in particular, the AIP (e.g. Figure 1)) are presented in the form of plan-files of corresponding Agora Managers, Negotiators or Coordinators. The part of the VE Initiator’s plan file that corresponds to the process of matching an agent’s skills to the requirements is shown below:

```
<step>
  <id>match_skill</id>
  <action>match_skill</action>
  <case>
    <postcondition>
      <performative>inform</performative>
      <ontology>VE</ontology>
    </postcondition>
    <nextstep>match_availability</nextstep>
  </case>
  <case>
    <postcondition>
      <performative>refuse</performative>
      <ontology>VE</ontology>
    </postcondition>
    nextstep>TERMINATE</nextstep>
  </case>
</step>
```

Some steps in the plan file refer to agent actions which are presented separately in action files. There are two kinds of bodies in an action:

- (i) *wrapper* for a message that can be predefined, for example, *send_announcement*, where the outgoing message is always the same. In such cases, the message is included in the action.
- (ii) *implementedBy* is used when the message cannot be predefined, e.g., the output of the action “align_goals” could be either refuse or query-ref, depending on the VE Partner's goals. This kind of action refers to a method which is written either by Java or by Prolog. The input of the method is the knowledge base of the agent and the output is the outgoing message.

The part of the action file for the VE Initiator for matching the skills is shown below:

```
<action>
  <id>match_skill</id>
  <implementedBy>
    <code>
      <codeLanguage>xprolog</codeLanguage>
      <codeMethod>match_skill</codeMethod>
    </code>
  </implementedBy>
</action>
```

7. Related Work

Most of the recent agent models employ a BDI-based model, [15]. (The BDI model of an agent takes into account the mentalistic notions of an agent such as its beliefs, desires and intentions.) We have not underlined the BDI aspect of our model in this paper. However, the VE agents' goals, activities/experiences and roles/capabilities in our model can be expressed in terms of beliefs, desires and intentions as well. Our agent architecture can be related functionally to those implemented, for example, in PRS, [10]. The parts of the knowledge base presented in these architectures are mostly focused on the agent's own self rather than the knowledge about the other agents, the acquaintance or the cooperation model.

A language called LARKS, for agent advertisements and requests, was defined in [16]. They also present a flexible and efficient matchmaking process, where both syntactic and semantic matching can be conducted. An institutionalised electronic organisation (e-institutions) to effectively design and construct agent societies was proposed in [6]. The e-institutions are specified by a formal approach and the interactions between agents can be defined by role/role relationship. A scene is specified by a graph where the nodes of the graph represent the different states of the agent society and the arcs connecting the nodes represent the agent's actions that changes the state of a scene. The idea of changing states by means of agent interactions is quite similar to what we use in AGORA.

The idea of using agents to represent the partners in a VE is not new, e.g. [2] and [11]. In [11], the VE Initiator, called the market agent, plays the role of a broker. Thus, they distinguish between the architecture of the VE Initiator and that of a VE Partner. With respect to our application domain where the agents represent human beings or organisations, there is a need to address the notion of trust and commitment. In this paper, we have not considered these aspects. The notion of trust is addressed in [8].

8. Conclusions

This paper describes how the AGORA multi-agent architecture can be used to support the formation of VEs. The partners in a VE are represented by agents. The main aim of this work is to support the human user in his/her decision making process.

We have described the VE formation process using AIPs and presented a prototype implementation using the AGORA architecture. The main contribution of this work is the agent-based approach itself, which uses an agent-based model to describe the VE and the description of the VE formation process as interactions among the agents, (i.e. as an AIP). We have not addressed the negotiation process(es) during the VE formation. However, we believe that this is an important part of the support for VE formation and thus we plan to focus more on this aspect in the future. We also plan to extend our work to include a more detailed description of the VE requirements and to support the automatic translation of AIPs to agent's plans.

9. Acknowledgements

This work is partially supported by NFR in the framework of the Information and Communication Technology (IKT-2010) program and the ADIS project.

10. References

- [1]. <http://www.iro.umontreal.ca/~vaucher/xprolog/>
- [2]. Barbuceanu, M., Fox, M.S., The Information Agent: An infrastructure agent supporting collaborative enterprise architectures, In 3rd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1994.
- [3]. Bauer, B., Muller, J. P. and Odell, J., Agent UML: A Formalism for Specifying Multi-agent Interaction, in Proc. of Agent-oriented Software Engineering, P. Ciancarni and M. Wooldridge (eds.), Springer Verlag, Berlin, 2001, p. 91-103.
- [4]. Bernus, P., Bertok, P., Nemes, L., Modelling Dynamic Management Features of Virtual Enterprises, in *Computer Applications in Production Engineering*, Plonka and G. Olling (Eds.), Chapman and Hall, 1997, p. 643-655.
- [5]. Camarinha-Matos, L. M., Afsarmanesh, H., Virtual Enterprise Modelling and Support Infrastructures: Applying Multi-agent System Approaches, in *Multi-agent Systems and Applications*, M. Luck, V. Marik, O. Stpankova, R. Trappl (eds.), in LNAI 2086, Springer, July 2001.
- [6]. Esteva, E., Rodriguez-Aguilar, J., Sierra, C., Garcia, P., Arcos, J. L., On the Formal Specification of Electronic Institutions, in Agent Mediated Electronic Commerce, No. 1991 in LNAI, Springer Verlag, 2001, p. 126-147.
- [7]. FIPA Agent Communication Language. <http://www.fipa.org/>
- [8]. Gans, G., Jarke, M., Kethers, S., Lakemeyer, G., Ellrich, L., Funken, C., Meister, M., Towards (Dis)Trust-Based Simulations of Agent Networks. in: Proc. of the 4th Workshop on Deception, Fraud, and Trust in Agent Societies, Montreal, May 2001, p. 49-60.
- [9]. Matskin, M., O. J. Kirkeluten, S. B. Krossnes and Ø. Sæle. Agora: An Infrastructure for Cooperative Work Support in Multi-Agent Systems. T. Wagner, O. Rana (eds.) Infrastructure for Agents, Multi-Agents, and Scalable Multi-Agent Systems. Springer Verlag, LNCS, Vol. 1887, 2001.
- [10]. Myers, K. L., User Guide for the Procedural Reasoning System, Technical Report, Artificial Intelligence Centre, SRI Int., Menlo Park, CA., 1997.
- [11]. Oliveira, E. and Rocha, A. P., Agents' Advanced Features for Negotiation in Electronic Commerce and Virtual Organisation Formation Process, European Perspectives on Agent Mediated Electronic Commerce, Springer Verlag, June 2000.
- [12]. Petersen, S. A., Divitini, M., Matskin, M., An Agent-based Approach to Modelling Virtual Enterprises, *Int. Journal of Production Planning & Control*, Vol. 12, No. 3, Apr. 2001, p. 224-233.
- [13]. Petersen, S. A., Gruninger, M., An Agent-based Model to Support the Formation of Virtual Enterprises, Int. ICSC Symposium on Mobile Agents and Multi-agents in Virtual Organisations and E-Commerce (MAMA'2000), in Woolongong, Australia, 11-13 Dec. 2000.
- [14]. Petersen, S. A., Matskin, M., Agent Interaction Protocols for the Selection of Partners for Virtual Enterprises, Multi-agent Systems and Applications III, 3rd Int. Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic, June 16-18, 2003. V. Maik, J. Müller, M. Pchouek (Eds.), LNAI 2691, Springer-Verlag, 2003.
- [15]. Rao, A. S., Georgeff, M. P., Modelling Rational Agents within a BDI Architecture, In R. Fikes and E. Sandewall (Eds.) Proc. of Knowledge Representation and Reasoning (KR&R91), Morgan Kaufmann Publishers: San Mateo, CA., 1991, p. 473-484.
- [16]. Sycara, K., Widoff, M., Klusch, M., Lu, J., Larks: Dynamic Matchmaking among Heterogeneous Software Agents in Cyberspace, *Autonomous Agents and Multi-agent Systems*, 5(2), 2002, p. 173-203.