Requirements Driven Agent Collaboration

Liwei Zheng
Academy of Mathematics and System Science, CAS
Graduate University of Chinese Academy of Sciences
Beijing, China
wizlw@amss.ac.cn

Zhi Jin
Academy of Mathematics and System Science, CAS
Institute of Computing Technology, CAS
Beijing,China
zhijin@amss.ac.cn

ABSTRACT

This paper proposes the requirements driven agent collaboration. This proposal assumes that there are plenty different service agents distributed in Internet. When a request for accomplishing a particular task occurs, these autonomous agents can recognize the newly emergent requirements and dynamically aggregate together to compete with others for fulfilling the requirements. This paper presents a preliminary framework for the requirement driven agent collaboration based on the automated mechanism design.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems,Coherence and coordination*

General Terms

Theory

Keywords

Multi-Agent, Collaboration, Mechanism design

1. INTRODUCTION

A multi-agent system (MAS) is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [3]. Currently, agent-based systems technology is particular attractive for creating software that operates in environments that are distributed and open, such as the internet. At present, there are already many MASs, e.g. those registered in the agentcities [1]. However, there are mainly two kinds of computing mechanisms for these systems. The first one is centered by a system manager or a coordinator, e.g. the *description database* with the *kernels* in MACE [6]. In which, the manager allocates tasks and chooses strategies and the other agents behave according to the commands issued by the manager. FIPA [2] gives a standard for this kind of MASs. The second includes those multi-agent systems based on game theory. In such systems, agents form coalitions, make solutions by negotiation, and work under some manually given rules which are designed for particular purposes [5] [7].

Currently, service-oriented architecture, connecting Web services, has been paid lots of attentions for distributed computing and communication on loose coupling and heterogeneous platforms [9]. In this architecture, Service Agency is responsible for managing and discovering services; Service Providers publish their services by registering them in Service Agency and controls the visiting to their services; Service Requestors search suitable services in the Agency and establishes bindings with the Provider to use services. When considering the Web services as the Service Agents, this mechanism of service-oriented computing may appeal a new computing mechanism for MASs. This computing mechanism can be demonstrated by the following scenario. A requestor publishes a task request onto the Web. The available Service Agents on the Web detect the newly emergent requirements, recognize those (sub-)tasks which they can accomplish, and compete for being a member of the candidate agents for the request. After all the candidate Service Agents gather together, the technique of mechanism design is used to generate the protocols to make these service agents collaborating to fulfill the task request. We call this kind of computing mechanism the *requirement driven agent collaboration*.

Compared with the service-oriented architecture, in the requirement driven agent collaboration, we have also the Agent Providers for providing service agents and the Requestors for submitting the task requirements. But Web services have been replaced by autonomous intensive Service Agents. And we do not need the Service Agency. Instead, these published Service Agents can perceive the requests, recognize the sub-tasks that they can accomplish and decide whether they are willing to make contribution to the task accomplishment.

Figure 1 demonstrates the scenario of the requirement driven agent collaboration in distributed and open Web environment. In this figure, the larger ellipses are the requests submitted by requestors which are usually represented by some kind of task specification. The smaller circles are available service agents. For an ellipse, there are some circles around it. That means those service agents have recognized that they are able to make contribution to the request sat-

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

isfaction and are willing to compete with others for some of the sub-requests. In this sense, the requirement driven agent collaboration means the process of agent congregation for completing an emergent requirements. In this process, each request seems like a magnet which can make the service agents moving towards it. That is the reason we use the term of 'magnet effect' for capturing the relationship between the requirements and the service agents.

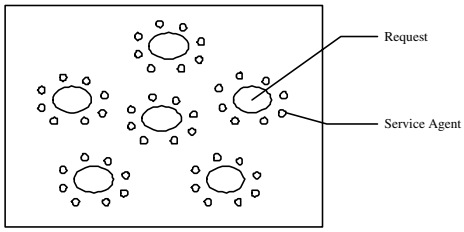

Figure 1: Magnet effect between requests and agents

This paper proposes a preliminary framework for the requirement driven agent collaboration. This research aims to develop another computing mechanism for multi-agent systems which we believe will help to realize the vision for the next Web revolution.

This framework contains two parts. The first is the facility built for supporting the service agents to understand the specification of the request. A *Functional Ontology(FO)* [10] is constructed for this purpose. This ontology provides the terminology for describing both the requested tasks and the service agents' capability. So, with this ontology, service agents can understand the requirements and decide if they need to be engaged for fulfilling the requirements.

Another part is the mechanism design for making the service agents automatically collaborating. A two-step model of agents collaboration has been built for this purpose. The first step is generating the protocols, so that the desirable outcomes will follow if the service agents use these protocols according to some stability solution concept, e.g. dominant strategies, Nash equilibrium or its refinements, etc. Our mechanism design extends the Automated Mechanism Design (AMD) [8] by introducing the task requirements and the process of generating the outcomes from the task requirements. Figure 2 depicts the extended AMD. The second step is the decision making for choosing a final solution for a certain outcome. The requests can be satisfied if all the outcomes have solutions.

2. REQUIREMENT DRIVEN AGENT COLLABORATION

For enabling the requirements driven agent collaboration, agent have to understand the specification of requirement. A *Functional Ontology* is constructed for this purpose. This ontology gives the the same terminology for describing the capability of agents and requests.

The Functional Ontology is represented in a hierarchy of controllable *resources* together with the effects on them[4]. In the Functional Ontology, each individual *resource* has *attributes*. These *attributes* are dived into two groups: the *static attributes*, i.e. their values are fixed once the *attributes* are instantiated, and the *dynamic attributes*, i.e. their values are changeable along different situations. We call the *static attributes* with their values the "*information*" of the *resource*

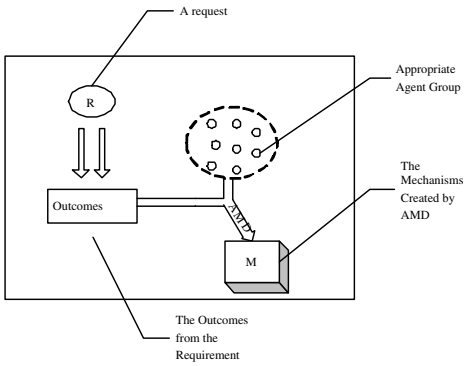

Figure 2: Extended AMD

and call the *dynamic attributes* with their values the "*states*" of the *resource*. Any set of state changes of a resource is called an effect on this resource.

It is worth to notice that "effect" can bridging the requests and the agents' capability if we model both of them via the resources that the agents operate on or that the requests want to operate on and the effects the agents can impose or the requests desire to impose. More formally, a request is represented by a three corresponding vectors, i.e. a resource vector, a starting state vector, and an ending state vector. In which, the ending state of each resource is called an outcome of the request. All of outcomes will be needed by the process of mechanism design.

With the same terminology, we represent an agent with a four-part-structure, i.e. a set of behaviors, a set of resources which it operates on, a set of state transitions that it enables, and a function from the behavior set to the transition set.

Then the process for an agent to recognize a given request includes the following steps:

1) Find its operatable resources in the request's resource vector. If there is no matched resource, the request is not suitable for it, else record all the matched resources.

2) Get the beginning state and the ending state of each matched resource from the requirement.

3) Computing state transition pathes from the starting state to its corresponding end state for each matched resource according to FO.

4) Check whether it has a behavior which can enable one of the effective state transitions.

5) If it has no matched behaviors, the request is not suitable for it, else it becomes one of the candidates of the request.

Any appropriate agent group has a time limitation for waiting its candidate agents. This time limit is different for each request which is decided by the request provider base on his/her experiences. After the waiting time, a local environment composed of a request and its candidate agents will be constructed. This environment is called a *requirement domain*. The candidate agent set in a requirement domain is also called an appropriate agent group of the requirement. These agents in the appropriate agent group will collaborate to complete the request.

The agent collaboration in a requirement domain can be finished in two steps. The first is the process of automated

mechanism design. That would give optimal outcomes for different agent groups. The second step is the agent negotiation process for given outcomes. That would produce effective agent behavior sequence for a given outcome.

The model of automated mechanism design in the first step includes the preconditions for AMD, the mechanism definitions, and the constraints definitions.

The preconditions of AMD include the finite set of outcomes of the request: O, and the finite Agent set composed of all the agents in the requirement domain. For each agent i, it has a finite set of types Θ_i and a utility function $u_i : \Theta_i \times O \to \mathbb{R}$. Here the utility function is common knowledge given by the request.

In our extended AMD, mechanisms are deterministic. A deterministic mechanism without payments consists of an outcome selection function $o : \Theta_1 \times ... \times \Theta_N \to O$. A deterministic mechanism with payments consists of an outcome selection function $o : \Theta_1 \times ... \times \Theta_N \to O$ and for each agent i, a payment selection function $\pi_i : \Theta_1 \times ... \times \Theta_N \to \mathbb{R}$. The payment is for the request provider.

Two types of constraints are used in our AMD process, IR(individual rationality constraints) and IC(incentive compatibility constraints). IR constraints ensure every agent participator would gain its lower limit of payment at least. IC constraints is to ensure the agents will never misreport their type. For example the definition of an ex interim IR constraint is: for any agent i, and any type $\theta_i \in \Theta_i$, we have

$$E_{(\theta_1,...,\theta_{i-1},\theta_{i+1},...,\theta_N)|\theta_i}[u_i(\theta_i, o(\theta_1, ..., \theta_N)) - \pi_i(\theta_1, ..., \theta_N) - \delta] \geq 0$$

in which $\delta \in \mathbb{R}$ is the lower limit of agent i's payment, $\pi_i : \Theta_1 \times ... \times \Theta_N \to \mathbb{R}$ is a payment selection function for the requirement provider.

AMD finds whether there exists a mechanism which can satisfy all the constraints. And then with a mechanism we can get a group of appropriate agents for a certain outcome.

The negotiation process gives an optimal solution for a given outcome. For any outcome $o_i \in O$, we assume its corresponding agent group has n members. Each member gives a solution for o_i. Each solution corresponds to a payment vector $P = (x_1, x_2, ..., x_n)$, x_i is the payment for a_i. Assume for any agent i, the desired payment vector is $P_i = (a_{i1}, a_{i2}, ..., a_{in}), i = 1, ..., n$, and the payment vector of the final solution is $P = (x_1, x_2, ..., x_n)$. Then the loss of every agent in the final solution can be described as a distance function:

$$d_i(P) = \sqrt{(x_1 - a_{i1})^2 + (x_2 - a_{i2})^2 + ... + (x_n - a_{in})^2}$$

The loss of the agent group also can be described as a holistic loss function:

$$f(p) = d_1^2(P) + d_2^2(P) + ... + d_n^2(P)$$

This function suggests the group's dissatisfaction degree when using solution P. Assume P can minimize f, then we have:

$$P = (x_1, x_2, ..., x_n) = \frac{1}{n}\sum_{i=1}^{n}(a_{i1}, a_{i2}, ..., a_{in}) = \frac{1}{n}\sum_{i=1}^{n} P_i$$

But P is only an ideal payment vector, and sometimes there is no solution corresponding to it. So we select the closest P_i to P as the final payment vector. That means the solution which has a smallest $d_i(P)$ is the final one. The solution corresponding to $d_i(P)$ can be described as: $Solution = ((a_i, b_k^{a_i}), (a_j, b_m^{a_j}), ...)$ $i, j, k, m \in \mathbb{N}$,a_i denotes an Agent i, $b_k^{a_i}$ is a_i's k_{th} behavior, the behavior sequence of the solution corresponds to an unique transition sequence, and the ending state of the transition sequence is o_i. For each $o_i \in O$, a solution can be obtained through this negotiation process. And the requirement would be completed by the execution of the solutions.

3. CONCLUSIONS

This paper proposes a preliminary framework for the requirement driven agent collaboration. The main contributions of our work include that: (1) We propose a new computing mechanism for multi-agent systems on the Web that has been termed the requirements driven agent collaboration; (2) We propose a preliminary framework for the requirements driven agent collaboration. Our future work mainly includes the Automated Mechanism Design for the non-deterministic outcomes in requirement domain and the approach for the optimal payment vector in the group decision-making model.

4. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Fund for Distinguished Young Scholars of China under Grant No.60625204, the Key Project of National Natural Science Foundation of China under Grant No.60496324, the National 973 Fundamental Research and Development Program of China under Grant No.2002CB312004, the National 863 High-tech Project of China under Grant No.2006AA01Z155, the Knowledge Innovation Program of the Chinese Academy of Sciences and MADIS.

5. REFERENCES

[1] http://www.agentcities.org/.

[2] http://www.fipa.org/.

[3] L. V. Durfee, E.H. and D. Corkill. Trends in cooperative distributed problem solving. In *IEEE Transactions on Knowledge and Data Engineering*, volume 1, pages 63–83, 1989.

[4] L. Hou and Z. Jin. Fect: A modelling framwork for atuomatically composing web services. In *WAIM 2005 Conference Proceedings,LNCS 3793*, pages 320–332, 2005.

[5] M. Klusch and O. Shehory. Coalition formation among rational information agents. In *LNAI No. 1038, Agents Breaking Away. Springer-Verlag*, pages 204–217, 1996.

[6] C. B. L. Gasser and N. Hermann. Mace: A flexible testbed for distributed AI research. *Distributed Artificial Intelligence*, pages 119–152, 1987.

[7] S. K. Onn Shehory. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[8] T. Sandholm. Automated mechanism design: A new application area for search algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming(CP)*, 2003.

[9] A. T. An overview of standards and related technology in web services. *Distribute and Parallel Databases*, 12:135–162, 2002.

[10] L. Zheng and Z. Jin. Requirement driven agent collaboration based on functional ontology and *AMD*. In *11th IEEE International Workshop on Future Trends of Distributed Computing Systems*, March 2007.