Mediator Based Open Multi-agent Architecture for Web Based Learning

Hongen Lu

School of Information Technology
Deakin University
221 Burwood Highway, Burwood
VIC 3125, AUSTRALIA
helu@deakin.edu.au

Abstract. Web based learning plays an important role in modern teaching environment. Many Web based tools are becoming available on this huge marketplace. Agent technology contributes substantially to this achievement. One of the fundamental problems facing both students and education services providers is how to locate and integrate these valuable services in such a dynamic environment. In this paper, I present a mediator based architecture to build open multi-agent applications for eLearning. An agent services description language is presented to enable services advertising and collaboration. The language exploits ontology of service domain, and provides the flexibility for developers to plug in any suitable constraint languages. Multiple matchmaking strategies based on agent service ontology are given to help agents finding appropriate service providers. The series of strategies consider various features of service providers, the nature of requirements, and more importantly the relationships among services.

1 Introduction

The World Wide Web has the largest collection of knowledge ever in man kind history. It is one of the most important resources in modern education. With the success of search engines, such as Google, and the vast acceptance of online learning systems, such as WebCT, students and teachers can search text and images efficiently. These tools are changing our learning process in schools and universities all over the world everyday. However, the Web has not reached its full potential. At its early stage, the Web is solely a huge collection of digital information. Nowadays, it is evolving into a huge growing marketplace for information providers and consumers. Agent technology makes a substantial contribution to this achievement.

However, how to find information providers and how to integrate information agents in such an open environment are new challenges. Information agents, such as Ahoy [6], ShopBot [3], and SportsFinder [5], are programs that assist people to find specific information from the Web. They are information service providers, which have the capabilities to find information for users, for example locating

W. Zhou et al. (Eds.): ICWL 2003, LNCS 2783, pp. 339–350, 2003.

a person's homepage, finding the cheapest available prices for music CDs, or finding sports results of a team or a player. For a novice user, a challenge is how to find these services; for an information agent, the challenges are how to locate the service providers, and how to communicate with them to solve its tasks cooperatively. This is one of the basic problems facing designers of open, multi-agent systems for the Internet is the connection problem — finding the other agents who might have the information or other capabilities that you need [2].

In [4], two basic approaches to this connection problem are distinguished: *direct communication*, in which agents handle their own coordination and *assisted coordination*, in which agents rely on special system programs to achieve coordination. However in the Web application domain, where new agents might come into existence or existing agents might disappear at any time, only the latter approach promises the adaptability required to cope with the dynamic changes in the environment.

2 Related Works

2.1 Ontology

Ontologies are *content* theories about objects, their properties, and relationships among them that are possible in a specific domain of knowledge [1]. In a given domain, its ontology clarifies the structure of knowledge in the domain. It forms the heart of any system of knowledge representation for that domain. Without an ontology, or the formal conceptualisations, there can not be any vocabulary for representing knowledge, let alone automatic knowledge reasoning and inference. An ontology gives the terms used in a certain domain, as well as their relationships. So that we can use these terms provided to assert specific propositions about a situation. For example, in computer science education domain, we can represent a fact about a specific unit: unit SCC303, Software Engineering, is a third year undergraduate unit, where SCC303 is an instance of the concept *unit*. Once we have the basis for representing propositions, we can also represent more advanced knowledge, such as hypothesise, believe, expect, ect. Thus, we can construct a domain ontology step by step to describe the world.

2.2 Web Service Description Languages

Web services are Web accessible programs and devices that not only provide information to a user, but to enable a user to effect change in the world. Web services are among the most important resources on the Web, and they are garnering a great deal of interest from industry. Many emerging standards are being developed for low-level descriptions of Web services.

- **WSDL.** Web Service Description Language provides a communication level description of the messages and protocols used by a Web service. WSDL is

an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described in abstract, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.

– **Semantic Web.** The huge collection of information on the Web is fare beyond a person's ability to search and index. So machine-understandable data is a high priority to automatic processing online information. Semantic web is a step to define and link data on the Web in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications.

2.3 WebCT

WebCT is one of the leading on-line education tools. It provides teachers a powerful and convenient way to build up websites dedicated to publishing teaching materials for their subjects; meanwhile it is also a place for students to feedback their progress. No wonder WebCT is widely accepted in various levels of education institutes, especially for long distance learning. However, WebCT is a closed system. It can only let the teachers and students in the same university or in the same class to communicate each other. In this point of view, WebCT has not taken the full advantage of the World Wide Web, which now is a fast growing collection of services. WebCT is still based on the conventional client-server architecture. While the Web offers more flexible options, for example everyone on the Web could be an information provider and consumer at the same time. Peer to peer communication is becoming the mainstream of on-line publishing and marketing. I believe this is the future trend for on-line education, because in such architecture teachers and students can easily swap their roles and learn from each other. In addition, this architecture is open for everyone to join in.

3 Mediator Based Architecture

A mediator is a special kind of information agent acting as middle man to take as input, a request to find an agent that provides a service, and returns as output, a list of such agents and their cooperation relationships. A mediator also stores the services offered by different agents in the existing environment, and when a new agent is introduced into the environment it can register its capability to the mediator, using an agent service description language, if this agent wants its service to be used by others. Information agents also can unregister their services to the mediator when they want to quit the cooperation or exit. Also when an information agent receives a query or a subtask within a query that can not be solved by itself, it can request the mediator to find out other agents that have

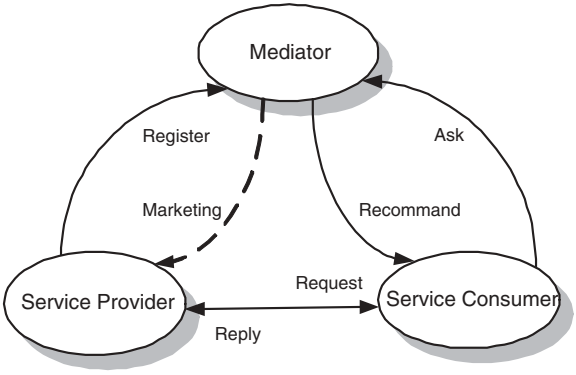

Fig. 1. Mediator Based Architecture

the capability or a set of agents who can work cooperatively to provide that service.

4 Agent Services Ontology

Since information agents are developed geographically dispersed over the Web, their capabilities are different from each other. SportsFinder [5] can find the sports results of golf, cycling, football and basketball etc. for users; while Ahoy [6] is good at locating people's homepages. In an application domain, such as Computer Science subjects, there exists a hierarchy relationship among these information agents. For example, information agent $\mathcal{A}$ can answer students' query about Software Engineering, while agent $\mathcal{B}$ is only capable of consulting on Risk Analysis, which is a part of the subject Software Engineering; in this case the service agent $\mathcal{B}$ can provide is a subset of agent $\mathcal{A}$, i.e. Service($\mathcal{B}$) $\subset$ Service($\mathcal{A}$).

To construct agent services ontology , it is necessary to identify their relations. Let S_i denotes the service of information agent $\mathcal{IA}_i$, and a service identifier to express in short what kind of service the agent can provide. For the above example, we have Service($\mathcal{B}$)={Software Engineering}, while Service($\mathcal{A}$)={Risk Analysis}.

- **Identical Service:** $S_1 = S_2$. This means the two services can provide the same function in spite of the fact that they may have different service names. As we know, information agents are being built over the Web using different programming languages and architecture. It is no surprised to have two agents running on different hosts that can offer the same service. Obviously, two identical services can substitute each other.
- **Subservice:** $S_1 \subset S$. This relationship characterises two services offered by agents, in which one service's function is only a part of another. For instance, an expert on C/C++ programming is good at tutoring lab project on Object

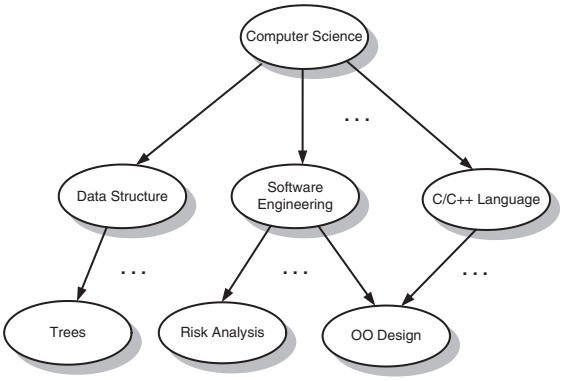

Fig. 2. Fragment of Computer Science Subjects Ontology

Oriented Design in Software Engineering unit; but he/she may not capable at formal methods in the same unit. In this point of view, the service offered by a tutor on C/C++, is only a part of a lecturer on the whole subject.

- **Substitute Service:** a service S_1 can be substituted by service S_2, $S_1 \leftrightarrow S_2$. From the above description, we know that identical service and subservice are two special cases of substitute service relationship. But the difference is that identical services can substitute each other, while the subservice can only be alternated by its "*parent*" service, not vice versa.
- **Partial Substitute Service:** $S_1 \cap S_2 \neq \phi$. This relationship describes two services that have some common subservices. In some circumstances, partial substitute services can be alternated with each other, such as where the service agent is offering, just by chance, the common subservice with its partial substitute service, that is, the agent is not offering its full service to others at the moment.
- **Reciprocal Service:** $\exists S = (S_1 \cup S_2)\ AND\ (S_1 \cap S_2) = \phi$, then S_1 and S_2 are reciprocal with S. If two services are reciprocal, that means they have no subservices in common, but they can work together to offer a "*bigger*" service. From this definition we know that in case there is no current agent available to provide the "*bigger*" service, these two reciprocal services can cooperate as a single agent for this task. This gives us a message that by combining the current agents in a different manner, we can tailor the system to meet new requirements.

Agent service ontology gives a formal method to describe the relationships among agent services. An agent service ontology contains all the services of information agents as well as their relationships. Basically, a directed cyclic graph (DCG) is able to present the relations between agent services. The nodes in the graph present the services, and the edges are labeled with the service relation-

ships. In Figure 2, a fragment of the ontology on computer science subjects is given, in sense of the content of the topic and their relationships.

```
<asdl> ::= ( service
               :service-id <name>
               :constraint-language <name>
               :input ( <param-spec>+ )
               :output ( <param-spec>+ )
               :input-constraints ( <constraint>+ )
               :output-constraints ( <constraint>+ )
               :io-constraints ( <constraint>+ )
               :service-ontology <name>
               |:<relation> <name>
               |:privacy <name>
               |:quality <name> )

<param-spec> ::= ( <name> <term> )
<relation>   ::= identical | subservice | substitute
                  | reciprocal | part-sub
<term>       ::= <constant> | <variable> |
                  ( <constant> <term>+ )
<constant>   ::= <name>
<variable>   ::= ?<name>
<name>       ::= <Identifier>

<constraint> ::= << expression in constraint-language >>
```

Fig. 3. Syntax for Agent Service Description Language in BNF

5 Ontology Based Agent Service Description

The proposed language in Figure 3 allows plugging in of an independent *constraint language*, that is the syntax of our ASDL is open at this point. This is described in the **constraint-language** field, which tells what language is used to present the constraints that should be hold on input, output and input-output. Also the **cap-id** field allows the specification of a name for this capability. The name for the capability is used to enable the middle agent to build a service ontology, and allows the **isa** field to naming a capability from which this capability will inherit the description. These two fields make it easier and simple to write a service description based on the already existed service ontologies, which is given as the value of **cap-ontology** field. The **privacy** and **quality** fields describe to what degree can other agents access this service and what the quality of this service is respectively. Depends on different domains, privacy and quality could be described in terms or functions.

6 Mediating Agent Services on the Web

Mediating is a process that utilise the knowledge on service domain to introduce service providers and consumers. Mediating is a high-level services matching and brokerage, in terms of level of knowledge applied, and directions of information flow. First of all, why do we need to mediate agent services on the Web? Let us look at the vast diversity of services that can be provided by agents all over the Web. Services are different in many aspects, I just name a few in the following:

- **Function.** It is obvious to note that different services have different functions. A sports agent has a totally different function to a shopping agent;
- **Constraints.** Even agents with the same function may impose different constraints on their input, output and input-output. For example, two lecturers both can be tutors on the subject, Data Structure and Algorithms, but one can only answer C questions, while the other is good at Java. Despite that they are able to consult on the same assignment question, but they require it in their capable language.
- **Quality and Privacy.** Quality and privacy are also varied from agent to agent, since they are run on different machines. Even when agents have the same function, due to the different implementations of the function, the qualities of their services may vary;
- **Names.** Agents may have different names despite the fact that they can provide the same service and have the same constraints and quality and privacy values.

The reasons that cause so many differences among agent services are mainly because of the open feature of the environment. Agents are developed over the Internet with heterogeneous architecture, and their functions vary from one to another. Due to diversity of agents, the requests of services are also various. In most cases, we can not expect that for a service request there is at least one agent to exactly provide that service, even through we suppose the service advertisement and request can fully express what the services are. In fact, a single agent can not have a global view of the whole system, it is not practical to do that, its request of service is also limited by the agent's "*partial*" knowledge of the environment.

7 Multiple Strategies for Services Matching

7.1 Type Matching

In the following definition, if type t_1 is a subtype of type t_2, it is denoted as $t_1 \preceq_{st} t_2$.

Definition 1. Type Match *Let $\mathcal{C}$ be a service description in our ASDL containing: an input specification $I^{\mathcal{C}}$ containing the variables $v_1, \ldots, v_n$, and output specification $I^{\mathcal{O}}$. Let $\mathcal{T}$ be a service request in ASDL with input specification $I^{\mathcal{T}}$ containing variables $u_1, \ldots, u_m$, and output specification $O^{\mathcal{T}}$. $\mathcal{C}$ is type matched with $\mathcal{T}$, if*

$$I^{\mathcal{T}} \preceq_{st} I^{\mathcal{C}} \text{ and } O^{\mathcal{C}} \preceq_{st} O^{\mathcal{T}}$$

where $I^{\mathcal{T}} \preceq_{st} I^{\mathcal{C}}$ *means* $\forall v_i \in I^{\mathcal{C}}\ \exists u_j \in I^{\mathcal{T}}$ *that* $u_j \preceq_{st} v_i$ *and for* $i \neq k$, $u_j \preceq_{st} v_i$, *and* $u_l \preceq_{st} v_k$, *we have* $j \neq l$.

This is the simplest strategy that only matches the types in the input and output fields of service advertisements against the correspondent field in requirements. It makes sure that a provider can take the inputs of requester, and its outputs are compatible with the requester's.

7.2 Constraint Matching

Definition 2. Constraint Match *Let* $\mathcal{C}$ *be a capability description in ASDL with input constraints* $C_I^{\mathcal{C}} = \{ C_{I_1}^{\mathcal{C}}, \ldots, C_{I_{k_{\mathcal{C}}}}^{\mathcal{C}} \}$ *and output constraints* $C_O^{\mathcal{C}} = \{ C_{O_1}^{\mathcal{C}}, \ldots, C_{O_{l_{\mathcal{C}}}}^{\mathcal{C}} \}$. *Let* $C_I^{\mathcal{T}} = \{ C_{I_1}^{\mathcal{T}}, \ldots, C_{I_{k_{\mathcal{T}}}}^{\mathcal{T}} \}$ *and* $C_O^{\mathcal{T}} = \{ C_{O_1}^{\mathcal{T}}, \ldots, C_{O_{k_{\mathcal{T}}}}^{\mathcal{T}} \}$ *be the input and output constraints respectively of service* $\mathcal{T}$. $\mathcal{T}$ *is constraint matched with* $\mathcal{C}$ *if*

$$C_I^{\mathcal{T}} \preceq_\theta C_I^{\mathcal{C}} \text{ and } C_O^{\mathcal{C}} \preceq_\theta C_O^{\mathcal{T}}$$

where $\preceq_\theta$ *denotes the* θ*-subsumption relation between constraints. For* $C_I^{\mathcal{T}} \preceq_\theta C_I^{\mathcal{C}}$ *means* $\forall C_{I_i}^{\mathcal{T}} \in C_I^{\mathcal{T}}\ \exists C_{I_j}^{\mathcal{C}} \in C_I^{\mathcal{C}}$ *that* $C_{I_i}^{\mathcal{T}} \preceq_\theta C_{I_j}^{\mathcal{C}}$ *and for* $i \neq k$, $C_{I_i}^{\mathcal{T}} \preceq_\theta C_{I_j}^{\mathcal{C}}$, *and* $C_{I_k}^{\mathcal{T}} \preceq_\theta C_{I_l}^{\mathcal{C}}$, *we have* $j \neq l$.

Since all the constraints are given in `constraint-language`, the details of θ-subsumption depends on the constraint-language. In first order predicate logic (FOPL), which is the constraint-language used in examples, constraints are a set of clauses. θ-subsumption in FOPL means there exists a substitution between two clauses.

7.3 Exact Matching

Exact match is most strict matching. It requires both the types and constraint fields are well matched. This strategy deals with the services that have the same functions but with different variable and type names. Considering the huge amount of Web-based applications which implemented over times and locations, there are many cases that developers may select different naming space.

7.4 Partial Matching

Definition 3. Partial Match *Let* $\mathcal{C}$ *be a service description in our ASDL containing: an input specification* $I^{\mathcal{C}}$ *containing variables* $V_{I_1}^{\mathcal{C}}, \ldots, V_{I_{n_{\mathcal{C}}}}^{\mathcal{C}}$, *and output specification* $O^{\mathcal{C}}$ *with variables* $V_{O_1}^{\mathcal{C}}, \ldots, V_{O_{m_{\mathcal{C}}}}^{\mathcal{C}}$, *and* $\mathcal{C}$*'s input constraints* $C_I^{\mathcal{C}} = \{ C_{I_1}^{\mathcal{C}}, \ldots, C_{I_{k_{\mathcal{C}}}}^{\mathcal{C}} \}$ *and output constraints* $C_O^{\mathcal{C}} = \{ C_{O_1}^{\mathcal{C}}, \ldots, C_{O_{l_{\mathcal{C}}}}^{\mathcal{C}} \}$. *Let* $\mathcal{T}$ *be another agent service with the correspondent description parts as: input* $I^{\mathcal{T}}$ *containing variables* $V_{I_1}^{\mathcal{T}}, \ldots, V_{I_{n_{\mathcal{T}}}}^{\mathcal{C}}$, *and output specification* $O^{\mathcal{T}}$ *with variables* $V_{O_1}^{\mathcal{T}}, \ldots, V_{O_{m_{\mathcal{T}}}}^{\mathcal{T}}$, *and* $\mathcal{T}$*'s input constraints* $C_I^{\mathcal{T}} = \{ C_{I_1}^{\mathcal{T}}, \ldots, C_{I_{k_{\mathcal{T}}}}^{\mathcal{T}} \}$ *and output constraints* $C_O^{\mathcal{T}} = \{ C_{O_1}^{\mathcal{T}}, \ldots, C_{O_{l_{\mathcal{T}}}}^{\mathcal{T}} \}$. *We define* $\mathcal{T}$ *is partial matched with* $\mathcal{C}$ *if*

$$\exists V_{I_i}^{\mathcal{T}} \in I^{\mathcal{T}}, \exists V_{I_j}^{\mathcal{C}} \in I^{\mathcal{C}} \ that \ V_{I_i}^{\mathcal{T}} \preceq_{st} V_{I_j}^{\mathcal{C}}$$
$$\exists V_{O_j}^{\mathcal{C}} \in O^{\mathcal{C}}, \exists V_{O_i}^{\mathcal{T}} \in O^{\mathcal{T}} \ that \ V_{O_j}^{\mathcal{C}} \preceq_{st} V_{O_i}^{\mathcal{T}}$$
$$\exists C_{I_i}^{\mathcal{T}} \in C_I^{\mathcal{T}}, \exists C_{I_j}^{\mathcal{C}} \in C_I^{\mathcal{C}} \ that \ C_{I_i}^{\mathcal{T}} \preceq_{\theta} C_{I_j}^{\mathcal{C}}$$
$$\exists C_{O_j}^{\mathcal{C}} \in C_O^{\mathcal{C}}, \exists C_{O_i}^{\mathcal{T}} \in C_O^{\mathcal{T}}, \ that \ C_{O_j}^{\mathcal{C}} \preceq_{\theta} C_{O_i}^{\mathcal{T}}$$

The above definition means for two capability descriptions, if some of their input, output variables have subtype relations, and there are constraint clauses in their input and output constraint specifications that are θ- subsumption, these two services are partial matched. Semantically, in some circumstances, i.e. the unmatched variables and constraints are irrelevant; the partial matched service is applicable.

7.5 Privacy Matching

Due to a service provider agent's privacy restriction, the matching result actually is sent to the service provider instead to the service requester. In other words, the provider agent wants to control the communication with consumers, it does not want to expose itself before knowing who are requesting its service. For instance, when recruiting for qualified software developers, some companies may not like their names known by their competitors, so they ask the agencies (middle agents) to keep their privacy. After the agency provides them with the resumes of potential experienced programmers, they can decide whom they would like to interview. Compared with other "*conventional*" matching strategies, privacy matching actually matches a service advertisement against service requests each time, while all the other strategies are vice versa; the information flow is different; the result of matching is transferred in a different direction. From the mediator's perspective, privacy matching is a service for capability providers. It supplies service request information to providers to help them marketing their services.

7.6 Cooperative Matching

Matching is a process based on a cooperative partnership between information providers and consumers. In cooperative matching process, the mediator first tries to find out from the current available information agents who have the capability that the query agent (information consumer) is asking for. In case no available agent can fulfill the queried service singly, the mediator will infer the relationships among available services, according to the domain ontology, to find a set of available information agents that can cooperate in some way to provide the requested service. This strategy requires an arbitrary amount of deduction and knowledge to match any given service and request. It exploits service ontology, knowledge on the application domain, to discover the hidden relationships among currently available services. It returns the agents contact information and their relationships.

8 TutorFinder: An Open Online Learning Tool

One great advantage of Web based learning is its openness. Everyone on the Internet can participate the learning and education process at any time they like. Traditional computer aided instruction (CAI) systems based on client-server architecture can not cope with this requirement. In order to take the full advantages offered by the Web, a new trend of online learning is open systems architecture, which introduces middleware to solve the connection problem.

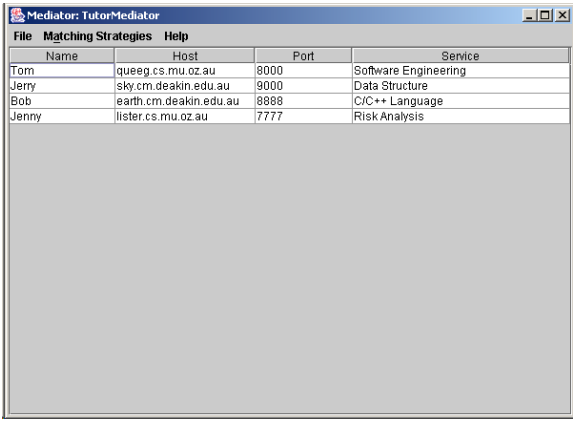

Fig. 4. Tutor Mediator

Based on the above mediator architecture and strategies, TutorFinder, an online tool for students and lecturers to locate suitable tutors, is presented in this section. TutorFinder is a mediator based open system. Any new available agent, who is able to offer services related to a specific eLearning subject, can register or advertise its ability to the TutorMediator, shown in Figure 4, who acts a middle man to mediate services requests and advertisements. This paradigm is open to any educators who wish to make their tools public over the Internet; in addition it is also open to any learners who are seeking some kind of helps. Service requests and advertisements are written in the proposed agent services description language, which can be easily plugged into any agent communication language. TutorMediator applies the multiple matching strategies to find out a or a team of service providers to inform to a consumer. The matching process can be reversed as a marketing campaign, in case the service provider would like to remain unknown until it knows who are seeking its services, and then the provider will target its marketing to the potential consumers. This procedure is depicted in Figure 1 as the dash line labeled with “Marketing”.

8.1 Services Description and Matching in TutorFinder

The presented agent services description language based on ontology provides a meaningful tool for service providers to express their capabilities. This is critical in a Web based learning environment, considering the open nature of eLearning. Using this language, online learning service providers can prescribe what kind of services they can offer to the community. For example, a Web service dedicated to answer students' queries on subject SCC303 Software Engineering can register its service to the above TutorMediator in the following format:

```
( service
    :service-id SCC303Tutor
    :constraint-language fopl
    :input ( (SCC303Question ?question) )
    :output ( (Answer ?answer) )
    :input-constraints (
       (elt ?question Question)
       (SubjectIn ?question SoftwareEngineering) )
    :io-constraints (
       (Correct ?question ?answer) )
    :service-ontology ComputerScience )
```

In this description, we know that the service SCC303Tutor takes questions in subject SCC303, Software Engineering, as input, and gives the correspondent answers. It requires an input to be a valid question defined in Computer Science Subjects ontology, and the question should be in the topics of Software Engineering; on these conditions, SCC303Tutor is able to give a correct answer. Please note that the constraints in this example are written in First Order Predicate Logic (FOPL), which is specified in `constraint-language` field. Actually, developers can choose any formal languages independent from ASDL to write constraints, and simply specify it in this field.

The ontology of Computer Science subjects is not only exploited in service advertising, in which it defines all the terms and their relationships used in the description, but also in service matching. Here I present a scenario in Figure 4.

In this scenario, there are four information agents available, and they can provide tutoring services on subjects of Software Engineering, Data Structure, C/C++ Language, and Risk Analysis to students. These four agents can be located at different universities and institutes. When a student or an agent requests services on Computer Science, TutorMediator can recommend a provider, or a list of service providers working as a team in case that the requested service can not be accomplished by any single agents. Considering a student who is doing a programming project on Object Oriented Design and Analysis, at the current situation, there is no single agents has the capability on OO Design and Analysis programming; but this requested service can be achieved by two agents Tom and Bob, who have the expertise on Software Engineering and C/C++ Language respectively, working cooperatively as a team. So by exploiting service ontology and cooperative matching strategy, TutorMediator can reply the student's query

with Tom and Bob's contact information, as well as their relationship in forming the team. Without ontology and various matching strategies, this can not be achieved. Powered with knowledge on the domain and a series of matching strategies, TutorMediator in our architecture is not a conventional middle agent, but an intelligent mediator who can reason and refer service providers' relationships, and guide them into cooperation.

9 Conclusion

The proposed agent service description language gives a flexible method for developers to plug in a suitable independent constraint language; it is more expressive for service quality and the privacy of service providers. The mediator, TutorMediator, in the presented open multi-agent architecture serves as middle agent that not only solves the connection problem, but also infers the cooperation relationships among information agents, this will direct service providers to forge a cooperation to answer a user's query. In such a way, tutoring agents can improve their capabilities, and online learning system becomes open and more scalable. This architecture with the service description language and matching strategies provides a solution to build open online learning system step by step. It also enables developers to integrate new tutoring services with legacy eLearning systems, since the architecture and language are open. This is critical for the success of online education, because both the educator and learner can take the full advantage of the World Wide Web, which gives people the freedom to pursue education from anywhere at anytime.

References

1. B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, January/February 1999.
2. Keith Decker, Katia Sycara, and Mike Williamson. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, December 1996.
3. Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the World Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.
4. Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, July 1994.
5. Hongen Lu, Leon Sterling, and Alex Wyatt. Knowledge discovery in SportsFinder: An agent to extract sports results from the Web. In *Methodologies for Knowledge Discovery and Data Mining, Third Pacific-Asia Conference (PAKDD-99) Proceedings*, pages 469–473. Springer, 1999.
6. Jonathan Shakes, Marc Langheinrich, and Oren Etzioni. Dynamic reference sifting: A case study in the homepage domain. In *Proceedings of the Sixth International World Wide Web Conference*, pages 189–200, 1997.