Information and Software Technology 54 (2012) 951–967

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Composing web services enacted by autonomous agents through agent-centric contract net protocol

Jonathan Lee, Shin-Jie Lee *, Hsi-Min Chen, Chia-Ling Wu

Department of Computer Science and Information Engineering, National Central University, Jhongli 320, Taiwan

ARTICLE INFO

Article history:
Received 1 June 2010
Received in revised form 21 January 2012
Accepted 1 March 2012
Available online 9 March 2012

Keywords:
Autonomous agents
Web service composition
Agent organization
Dynamic role binding
Reputation model

ABSTRACT

Context: Agents are considered as one of the fundamental technologies underlying open and dynamic systems that are largely enabled by the semantic web and web services. Recently, there is a trend to introduce the notion of autonomy empowered by agents into web services. However, it has been argued that the characteristics of autonomy will make agents become available intermittently and behave variedly over time, which therefore increase the complexity on devising mechanisms for composing services enacted by autonomous agents.

Objective: In this work, we propose an extension to Contract Net protocol, called Agent-centric Contract Net Protocol (ACNP), as a negotiation mechanism with three key features for composing web services enacted by autonomous agents.

Method: (1) A matchmaking mechanism embedded in a middle agent (as a service matchmaker) for discovering web services that are available intermittently is presented based on the concept of agent roles; (2) A selection algorithm based on risk-enabled reputation model (REAL) embedded in a manager agent (as a service composer) is introduced to serve a basis for selecting web services with variant performance; and (3) A negotiation mechanism between a manager agent and contractor agents (as atomic services) is devised and enables both a service composer and the atomic services to request, refuse or agree on adapting changes of services.

Results: The problem of assembling a computer is discussed in this paper.

Conclusion: It is increasingly recognised that web services would become more autonomous by introducing diverse agent technologies to better constitute more complex systems in open and dynamic environments. As web service technologies are best exploited by composite services, it is imperative to devise mechanisms for composing services of autonomy.

1. Introduction

Although efforts have been made on developing declarative descriptions for web service compositions [1–5], and more recently, in dealing with the problem of adaption to service changes in executions of service compositions [6–11], little emphasis has been put on addressing compositions of services enacted by autonomous agents [12].

Recently, agents are considered as one of the fundamental technologies underlying open and dynamic systems that are largely enabled by the Semantic web and web services [13,14]. In contrast with web services, agents are autonomous, namely, having control both over their internal states [15–17] and over their own behaviors [18–21], so as to dynamically respond and adapt to a changing environment [22], which then has been empirically perceived as a main benefit by industry sector after adoptions of agents [23]. There is a trend to introduce the notion of autonomy empowered by agents into web services [22,24,25]. In [22], it has been argued that the code in a web service container can employ the notion of agency to solve simple tasks or provide component functionality to support complex e-business machinery. In [25], the author argues that Web services currently involve a single client accessing a single server, but soon will demand federated servers with multiple clients sharing results. Cooperative peer-to-peer solutions will have to be managed, an area in which agents have excelled.

As autonomous agents can join and leave systems at any given time [26], and requests to agents may be immediately honored, refused, or modified [27], the problem of composing web services enacted by autonomous agents can be best explained as follows:

- *Discovering services*: How to discover web services that are available intermittently for a service composition in an open environment?

* Corresponding author.
E-mail addresses: yjlee@selab.csie.ncu.edu.tw (J. Lee), jielee@selab.csie.ncu.edu.tw (S.-J. Lee), seeme@selab.csie.ncu.edu.tw (H.-M. Chen), wucl@selab.csie.ncu.edu.tw (C.-L. Wu).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.03.001

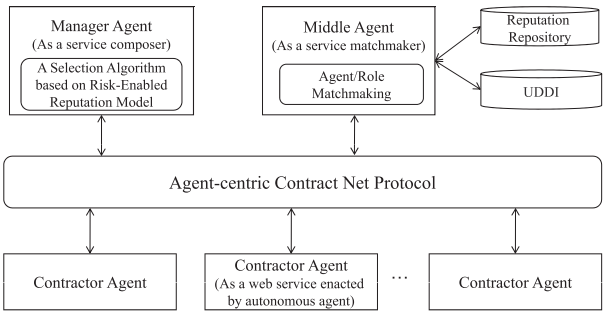

Fig. 1. A high-level system architecture of ACNP.

- *Selecting services*: How to select web services with variant performance for a service composition?
- *Adapting services*: How to adapt to the changes of web services during a service composition?

In this work, we propose an extension to Contract Net protocol [28] based on our previous work in [29], called Agent-centric Contract Net Protocol (ACNP), as a negotiation mechanism with three key features for composing web services enacted by autonomous agents (see Fig. 1. A High-Level System Architecture of ACNP):

- A matchmaking mechanism embedded in a middle agent (as a service matchmaker) for discovering web services that are available intermittently is presented based on the concept of agent roles that are widely adopted to characterize service capabilities in agent research area. Web services can register or unregister their agent specifications to a middle agent and will be matched dynamically by the middle agent with role specifications for a service composition.
- A selection algorithm based on risk-enabled reputation model (REAL) embedded in a manager agent (as a service composer) is introduced to serve a basis for selecting web services with variant performance. Ratings of web services are reported by a manager agent and kept by a middle agent. A measurable comparison shows that our approach can better reflect the recent performance of a web service than the other approaches in the literature [30,31].
- A negotiation mechanism between a manager agent and contractor agents (as atomic services) is devised and enables both a service composer and the atomic services to request, refuse or agree on adapting changes of services. A manager agent is entitled to rebind a new contractor agent dynamically once a contractor agent is not available, and the contractor agents are eligible to refuse or agree on unbinding from a service composition.

Fig. 1 shows a high-level system architecture of ACNP, which involves a manager agent, a middle agent, and multiple contractor agents. A manager agent serves as a service composer assembling web services enacted by autonomous agents (denoted as contractor agents). A selection algorithm based on risk-enabled reputation model is embedded in the manager agent for selecting contractor agents based on their reputations. A middle agent is associated with a reputation repository for keeping ratings of contractor agents, and a UDDI for registration of contractor agents. A matchmaking mechanism is embedded in the middle agent for discovering contractor agents based on agent/role specifications. Agents negotiate through Agent-centric Contract Net Protocol. Both the manager agent and the contractor agents are entitled to the following activities: request, refuse or agree upon adapting services based on the contracts consigned by all the relevant stakeholders.

This paper is organized as follows. We describe the background work in Section 2. Conceptual model of ACNP is elaborated in Section 3. Section 4 fully describes ACNP from a dynamic perspective. An experimental evaluation is discussed in Section 5. In Section 6, we describe the related work. Finally, we summarize the contributions of ACNP in Section 7.

2. Background work

The background work contributing to the development of ACNP includes service compositions, Contract Net protocol and a risk-enabled reputation model.

2.1. Service composition

It is widely accepted [32–34] that combining multiple web services into a composite service is more beneficial to users than finding a complex and preparatory atomic service that satisfy a special request. The resulting composite services can be used as atomic services themselves in other service compositions to satisfy clients' requests. Service composition plays a significant role in the Service-Oriented Computing domain to improve reusability, lower duplication, expedite time to market, and improve security and fault tolerance. Using composition mechanism enables developers to create dynamic business processes and agile applications in a more efficient way.

In ACNP, ontology is developed to better describe the concepts of roles and agent organizations. The key requirement of a potential target language for describing the ontology is that the language should be able to describe service composition processes based on IPO (Input-Process-Output)-style web services to serve as a basis for defining organization, agent and role specifications, and the role matchmaking mechanism. Recently, several languages for describing web services have emerged, for example, BPEL4WS [1], BPEL4SWS [5], OWL-S [2], WSML [35], and WSMO [3].

BPEL4WS [1] provides a mixture of block-structured and graph-structured process models, and variables associated with message types can be specified as input or output variables to invoke, receive, and reply web services. BPEL4SWS [5] uses BPEL as its basis and attaches SWS (semantics web services) descriptions to BPEL such that SWS frameworks like OWL-S and WSMO and their corresponding implementations can be used to discover and select SWS implementing the functionality required for an activity.

OWL-S [2] provides a language for describing service compositions and data flow interactions through modeling services as processes. The functional description of the service is expressed in terms of the transformation produced by the service, and more specifically, it specifies the inputs required by the service and the outputs generated.

In WSMO [3], a twofold view of the operational competence of a web service describes how the functionality of the web service can be achieved: choreography decomposes a capability in terms of interaction (including input and output information) with the web service, and orchestration decomposes a capability in terms of functionality required from other web services. WSML [35] is a family of formal description languages, which can be used as the specifications of the single elements in the WSMO framework. However, as noted in [35], no orchestration syntax has so far been specified by the WSML group, and it is currently not possible to embed an orchestration within a WSML description.

Generally speaking, most of the above-mentioned languages fulfill the requirement of describing the ontology in ACNP, and are eligible for the modeling of ACNP. In our work, we select OWL-S as the formalization mechanism because it serves as a semantics-enabled formalism for modeling web service compositions, and

for establishing a framework within which software agents technology can be deployed, which meets our need to devise an approach to composing web services enacted by autonomous agents.

OWL-S is an ontology consisting of a set of classes and properties for declaring and describing services based on OWL [36]. It is expected to enable automatic web service discovery, invocation, composition and interoperation. In OWL-S, three essential types of knowledge about a service are provided: (1) ServiceProfile, providing the information needed for a software agent to discover the service, such as service name and contacts; (2) ServiceModel, describing how the service is used, such as how to compose service descriptions from multiple services to perform a specific task; and (3) ServiceGrounding, describing how to interact with the service, such as transport protocols.

In OWL-S, a service is modeled as a process which is a specification of the ways a client may interact with the service. An atomic process is a description of a service that expects one message and returns one message in response. A composite process is decomposable into non-composite or composite processes by using control constructs such as Sequence, Split and If-Then-Else. Table 1 shows an example of a composite service description in OWL-S. It defines a composite process Purchase_Hardware which is composed of two sub-processes sell_CPU and sell_HardDisk. The two sub-processes are to be executed concurrently.

In this paper, we introduce the notion of agent organization to better deal with the problem of composing web services enacted by autonomous agents. In ACNP, agent roles define abstract interfaces of web services and can be used to constitute composite service specifications denoted as organizational objectives. A role has responsibilities and an agent (a web service enacted by an autonomous agent) has capabilities. An organizational objective is modeled as a kind of composite process that composes a set of roles' responsibilities or sub-organizational objectives. Each responsibility or capability is a kind of atomic process. A contract contains the information that indicates the relations between responsibilities of a role and capabilities of an agent.

Table 1
An example of a composite service description in OWL-S.

```
⟨process:CompositeProcess rdf:ID=''Purchase_Hardware"⟩
  ⟨owls-process:composedOf> <owls-process:Split⟩
    ⟨owls-process:Perform rdf:ID=''Perform1"⟩
      ⟨owls-process:process rdf:resource=''sell_CPU"/⟩
    ⟨owls-process:Perform rdf:ID=''Perform2''⟩
      ⟨owls-process:process rdf:resource=sell_HardDisk"/⟩
    ⟨/owls-process:Split⟩⟨/owls-process:composedOf⟩
⟨/process:CompositeProcess⟩
```

2.2. The contract net protocol

Contract Net Protocol (CNP) was introduced by Smith [28] to facilitate efficient cooperation in distributed systems. A minor modification of the Contract Net protocol, called FIPA Contract Net Interaction Protocol (IP) [37], was developed as one of the standards supporting inter-operability among agents and agent-based applications [38].

Messages in the protocol, such as *cfp* and *inform-done*, are FIPA communicative acts (CAs) that can differentiate utterances for agent communication language (ACL) messages [39]. For example, *inform* indicates that the sending agent holds that some proposition is true, intends that the receiving agent also comes to believe that the proposition is true, and, does not already believe that the receiver has any knowledge of the truth of the proposition. The formal model of the CAs can be found in [39].

Fig. 2 shows the FIPA Contract Net IP, which is represented based on extensions to UML1.x [40]. In the protocol, one agent (the Initiator) wants to have some task performed by one or more other agents (the Participants) and starts to send a *cfp* message to the participants. *cfp* is a CA in order to call for proposals to perform a given action. The Participants may respond with a proposal for a given task and the rest must refuse. Once the deadline passes, the Initiator evaluates the received j proposals and selects agents to perform the task. After that, the l agents of the selected proposal(s) will be sent an *accept-proposal* CA and the remaining k agents will receive a *reject-proposal* CA. Once the Participant has completed the task, it sends an *inform-done* CA or an *inform-result* CA to the Initiator. However, a *failure* CA will be sent if the Participant fails to complete the task.

Although CNP provides a means to task allocations in multi-agent systems, there are several issues on applying CNP to the service composition problem of assembling a computer:

- *How to discover participants before calling for proposals?* In CNP, it is assumed that an initiator knows the participants in advance and then can send *cfp* messages through broadcasting or point-to-point. However, how to discover the participants has not been addressed. In ACNP, the notion of middle agent and the related interaction protocols are introduced for registrations and discovering of the participants (refer to Section 4.1).
- *How to select participants to award tasks?* In CNP, an initiator maintains a rank-ordered list of proposals that have been received for the task, however, how the initiator ranks the proposals has not been addressed. In ACNP, a risk-enabled reputation model is introduced and extended for ranking

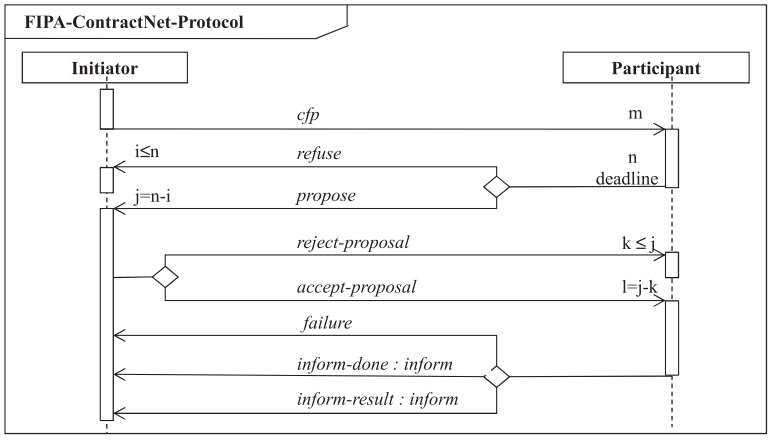

Fig. 2. FIPA Contract net interaction protocol [37].

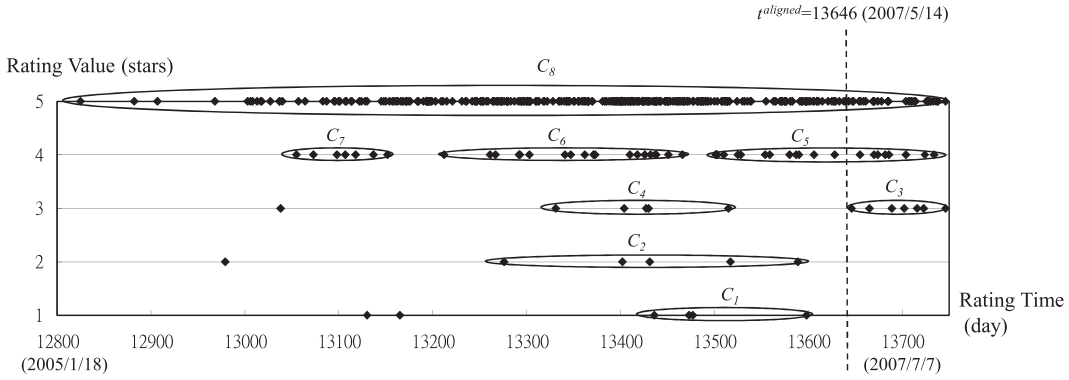

Fig. 3. Analyzing the reputation of an Amazon Auction's seller through risk-enabled reputation model.

and selecting the proposals (refer to Sections 2.3 and 4.2.2) to align with the trend of using reputation models for selecting service providers in open systems.

- *How to adapt to changes of participants?* In CNP, an initiator can cancel a contract after awarding tasks; however, how a participant can actively cancel a contract has not been addressed. In ACNP, a participant is able to inform the initiator of canceling a contract, and then the initiator can try to make contract with another one (refer to Section 4.2.3).

2.3. A risk-enabled reputation model

In our previous work, a risk-enabled reputation model (REAL) is proposed to evaluate a service provider's reputation together with a risk value [41], which has been validated as a better model to reflect a service provider's recent performance than given by the extant on-line auctions sites or the models using the technique of mean values. Moreover, the risk value can be utilized to further identify how risky it is to transact with the service provider. In this work, REAL is adopted to evaluate reputations of web services enacted by autonomous agents. We give the main steps in REAL as follows, and the formal algorithms can be found in [41].

Step 1. *Record ratings*. This step is to record ratings of a service. Rating, as the basic ingredient of the reputation model, records a service consumer's impression of a service on a particular subject at a point of time. A rating is defined as a tuple $r = \langle a, b, s, t, v\rangle$, where a is a service consumer; b is a service judged by a with respect to subject s; t is the time when the judgment takes place; and v is an integer value representing the result of the judgment.
Fig. 3 shows how the reputation of an Amazon Auction is analyzed by REAL. Ratings about one seller (service provider) in Amazon Auctions are projected to a two-dimensional plot with time as the X-axis and value as the Y-axis. Time of each rating is represented as days after January 1, 1970. The value of each rating is an integer from 1 to 5. In the example, there are 423 ratings given by buyers (service requesters) about the seller during the period from May 13, 2005 to October 31, 2007.

Step 2. *Discover active ratings*. This step is to discover the ratings (called active ratings) that can be used to calculate the reputation and risk of the service by the next step, which can be further broken down into the following activities: Firstly, ratings of the same rating value (e.g., 1–5 stars in the example) are clustered into a number of clusters (e.g., $C_1, \ldots, C_8$) based on a density-based clustering algorithm proposed in [42]. Secondly, to identify the clusters recent enough, called active ratings clusters C^{active} (e.g., C_3, C_5, C_8), from the clusters. Thirdly, a time point $t^{aligned}$ (e.g., 13,646) is identified to align C^{active}, which implies ratings with time earlier than $t^{aligned}$ will be removed from C^{active}. The remaining ratings in the active ratings clusters are called active ratings R^{active}. Finally, to identify the ratings, called active outliers $Outl^{active}$, that are not active ratings but their rating time are later than $t^{aligned}$.

Step 3. *Calculate reputation and risk*. This step is to calculate the reputation and risk of the service. The reputation is calculated by Eq. (1):

$$rep = \begin{cases} v^{max}_{R^{active}}, & \text{if } |R^{active}|_{\#} \neq 0; \\ v^{max}_{Outl^{active}}, & \text{else.} \end{cases} \tag{1}$$

If the number of the active ratings clusters is not zero, the reputation is set to be the highest rating value of the active ratings $\left(v^{max}_{R^{active}}\right)$, else the reputation is set to be the highest rating value of the ratings in the outliers $\left(v^{max}_{Outl^{active}}\right)$. Hence, the possible values of the reputation are $l_1, \ldots, l_m$.

A risk value associated with the reputation is calculated (by Eq. (2)) as the sum of the products of the following two values for each rating value smaller than the calculated reputation: (1) the probability of the rating value $\left(\frac{|R^{active}_i \cup Outl^{active}_i|_{\#}}{|R^{active} \cup Outl^{active}|_{\#}}\right)$, where R^{active}_i and $Outl^{active}_i$ are ratings with value i in R^{active} and $Outl^{active}$, respectively; and (2) the difference between the calculated reputation and the rating value $(rep - i)$. A risk value, for example $risk_5$, can represent the potential disappointment of a buyer that he expects a 5-stars performance of the seller but is served a performance below 5 from the seller.

$$risk_{rep} = \sum_{l_1 \leqslant i < rep (i \in L)} (rep - i) \times \frac{|R^{active}_i \cup Outl^{active}_i|_{\#}}{|R^{active} \cup Outl^{active}|_{\#}} \tag{2}$$

If there is only one rating in $\left|R^{active}_{rep} \cup Outl^{active}_{rep}\right|_{\#}$ with value l_m and the rest of the ratings are with value l_1, the maximum risk value will approximate to $l_m - l_1$. If there is no active ratings with value below rep, the minimum risk value is 0. Hence, the possible values of the risk values are real numbers in $[0, l_m - l_1)$.

In the auction seller example (see Fig. 3), the reputation of the seller is calculated as 5 (the highest rating value of the ratings in the active ratings) although he had active ratings with values 3 or 4. Besides, the numbers of the ratings in R^{active}_3, R^{active}_4 and R^{active}_5 are 8, 10 and 37, respectively. The risk associated with reputation 5 can be calculated as follows:

$$risk_5 = (5-3) \times \frac{8}{37+10+8} + (5-4) \times \frac{10}{37+10+8} = 0.47$$

We would say that the highest reputation of the seller in recent times is 5 stars but a buyer would have some degree of risk, i.e., 0.47, to transact with the seller. Section 4.2.2 will discuss how the value of the risk can be further used to select service providers.

In the followings, we present a comparison of REAL, Amazon Auctions [31] and REGRET [30]. The requirement of the comparison is that given a set of service providers' ratings with time and values, calculate the reputations of the service providers.

Fig. 4 shows the plots about selected three sellers' (service providers') ratings that are posted from Amazon Auctions before a certain selected date (2007/8/31). The three sellers denoted as A, B and C are with 213, 423 and 252 ratings, respectively. It can be observed that ratings with bad performance (1 and 2 stars) of seller A in the latest days are getting increased, most latest ratings of seller B are with 3, 4 and 5 stars, and the latest performance of seller C is with ratings of 5 stars. By applying REAL, the align time for sellers A, B and C are 13,713, 13,646 and 13,707, respectively. Only ratings after the align time are selected for calculating reputations and the associated risks.

Table 2 shows the comparison of REAL, Amazon Auctions [31] and REGRET [30] on the analysis of the three sellers' reputations. In REAL, all these three seller's reputations are calculated as 5, which means the highest reputations of the sellers in the latest days are all 5 stars. The risks associated with the reputations of seller A, B and C are 0.85, 0.47 and 0, respectively.

By using the selection algorithm depicted in Section 4.2.2, a buyer can determine the values of minimum expected reputation (MER) and maximum acceptable risk (MAR), e.g., setting MER to 1 and MAR to 4 means to consider all the sellers no matter they are with very low reputations or high risks. Thereafter, the sellers are prioritized by their reputations and risks. Because the sellers are with the same reputations (5 stars), they are prioritized as C, B and A from high to low by their risk values. Finally, seller C can be selected by the buyer for transactions.

In Amazon Auctions, a seller's reputation is calculated as the average of his/her past rating values. The reputations of the three sellers calculated by Amazon Auctions are exactly the same as 4.7, which cannot help us distinguish among the three sellers.

In REGRET, Sabater and Sierra proposed a reputation model to evaluate agents through three dimensions: individual, social and ontological. The individual dimension models the direct reputations that can be calculated from the impressions of the past direct interactions with the target agent. The social dimension models the social reputations that are obtained from the opinions of the society. The ontological dimension models more complex reputations on different aspects, e.g., a reputation on a good seller is based on the aspects of delivery date, product size and product quality. In the experiment, the individual dimension of REGRET is applied because it best fits the requirement of the comparison. The key feature of the individual dimension is to give weights to ratings according to their time. Although more recent ratings will be given more weights on calculating reputations, the result shows that A's reputation is calculated as 4.6, and B and C's reputation are equal (4.7), which cannot be used to distinguish between B and C.

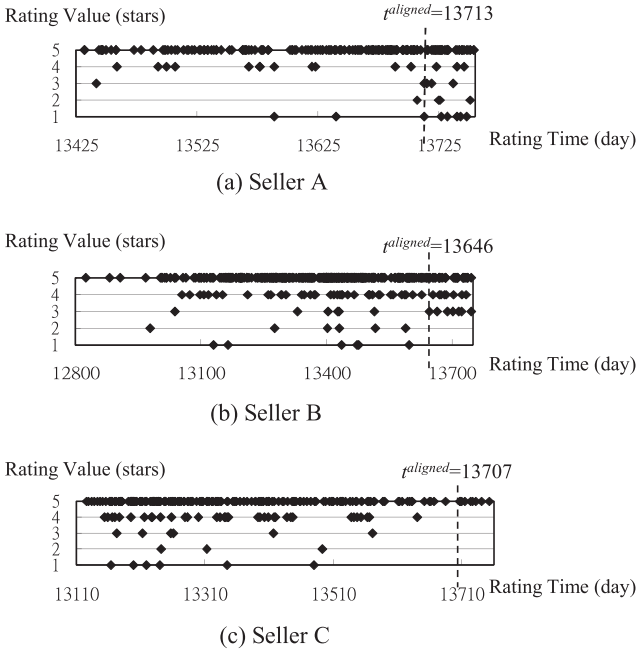

Fig. 4. Ratings of selected three sellers in Amazon Auctions.

Table 2
A comparison of service selection algorithms.

	REAL	Amazon Auctions	REGRET
Seller A	Rep. = 5, $Risk_5$ = 0.85	Rep. = 4.7	Rep. = 4.6
Seller B	Rep. = 5, $Risk_5$ = 0.47	Rep. = 4.7	Rep. = 4.7
Seller C	Rep. = 5, $Risk_5$ = 0	Rep. = 4.7	Rep. = 4.7

3. Conceptual model of ACNP

In this section, we elaborate on the conceptual model of ACNP and its associated ontology. The conceptual model describes the components of ACNP and their relationships. The components include agent organization specifications, organizational objective specifications, role specifications, agent specifications and contract specifications, as well as three types of agents that are manager, middle and contractor agents. The ontology serves as a basis for documenting the specifications of ACNP.

3.1. Conceptual model

Fig. 5 shows the conceptual model of ACNP. An agent organization provides a context for composing the services. An agent organization is described by an agent organizational specification that consists of one or more role specifications and organizational objective specifications. A composite service specification is denoted as an organizational objective specification. An organizational objective specification, which is associated with zero or more role specifications and organizational objective specifications, documents an organizational objective (composite service) that will be achieved by a manager agent. A manager agent acts as a service composer that composes web services enacted by antonymous agents which are denoted as contractor agents.

An agent organization consists of multiple manager agents for achieving organizational objectives. A middle agent and contractor agents are outside the agent organization. A contractor agent is associated with an agent specification registered to the middle agent. A middle agent consists of a registry of storing agent specifications and a reputation repository of recording ratings about contractor agents. A manager agent can discover contractor agents to bind roles by negotiating with the middle agent. Furthermore, a manager agent can make a contract with a contractor agent for binding a role with a contract specification that refers to information of a role specification and an agent specification.

3.2. Ontology

Fig. 6 shows the ontology that can be used to instantiate the specifications of ACNP. The ontology is formulated by OWL-S. Each ellipse represents an OWL class. The dashed and solid lines indicate the OWL subclass and object property relations, respectively.

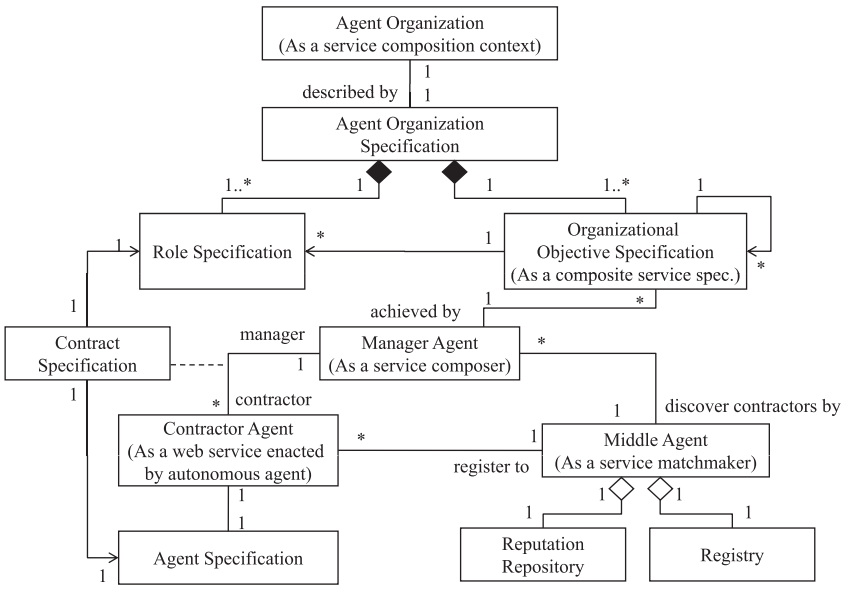

Fig. 5. Conceptual model of ACNP.

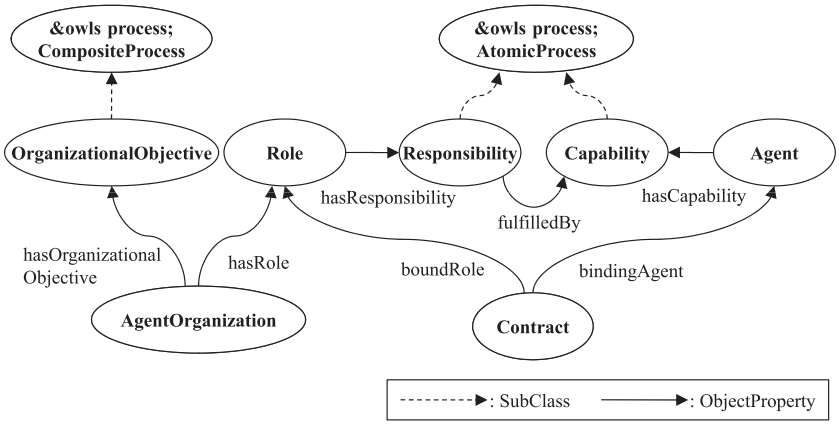

Fig. 6. Ontology for creating the specifications in the conceptual model.

An agent organization can have organizational objectives and roles. A role can have responsibilities. An agent can have capabilities. Each responsibility or capability is a kind of atomic process. An organizational objective is a kind of OWL-S composite process that composes a set of roles' responsibilities or other organizational objectives. As an agent binds a role, a contract will be generated and be associated with the role and the agent, and each responsibility of the role will be associated with a capability of the agent to indicate that the responsibility will be fulfilled by the capability.

3.3. A running example: assembling a computer

A running example is as follows: A person is going to assemble a computer by himself instead of buying a computer pre-assembled by a particular merchant. He starts with getting on a web auction site to buy numerous hardware and software components. The hardware components include a CPU, a mother board, PC memory, a hard drive, a graphic card, a sound card, a DVD ROM, an LCD monitor, a power supply, a mouse, a keyboard and a case. The software components include an OS and an office suite. However, he gets a tedious work to complete for the purchasing in a traditional way: First, he has to input the names or keywords of the components on the web site to search for the sellers (service providers); Second, he has to browse a number of web pages to get an insight of the recent performance (reputation) of each seller, which serves him a basis for selecting a seller to transact for a component. Third, he has to re-prioritize and re-select sellers if he wants to change the selected sellers. After he obtains all the components from the sellers, he can assemble these components to be a computer. In order to assist the person to complete the tedious work, a software system is strongly needed. In the software system, each service provider can be modeled as a web service enacted by an autonomous agent, and the system provides composite services by discovering, selecting and adapting these web services.

In order to assist the person to complete the tedious work, a software system is strongly needed. The requirements of the software system are as follows:

- The system should assist users in discovering service providers (sellers) that may appear and disappear in the open environment.
- The system should assist users in selecting service providers that may act autonomously, such as giving unstable delivery quality over time.

Table 3
Specifications for Agent Organization of Assembling a Computer.

Agent Organization Specification: Computer_Assembling_Organization

```
⟨acnp:AgentOrganization rdf:about=''#Computer_Assembling_Organization"⟩
  ⟨acnp:hasOrganizationalObjective rdf:resource=''& ass;Assemble_a_Computer''/
  ⟩
  ⟨acnp:hasOrganizationalObjective rdf:resource=''& ass;Purchase_Hardware''/⟩
  ⟨acnp:hasOrganizationalObjective rdf:resource=''& ass;Purchase_Software''/⟩
  ⟨acnp:hasRole rdf:resource=''& cpu_seller_role_spec;CPU_Seller''/⟩...
⟨/acnp:AgentOrganization⟩
```

Organizational Objective Specification: Purchase_Hardware

```
⟨acnp:OrganizationalObjective rdf:ID=''Purchase_Hardware''⟩
  ⟨owls-process:composedOf⟩⟨owls-process:Split⟩
    ⟨owls-process:Perform rdf:ID=''Perform_sell_IntelCore2DuoE8500''⟩
    ⟨owls-process:process rdf:resource=''&
    ass;cpu_seller_role_spec.owl#sell_IntelCore2DuoE8500''/⟩
    ⟨/owls-process:Perform⟩...
  ⟨/owls-process:Split⟩⟨/owls-process:composedOf⟩
⟨/acnp:OrganizationalObjective⟩
```

Role Specification: CPU_Seller_Role

```
⟨acnp:Role rdf:about=''#CPU_Seller"⟩
  ⟨acnp:hasResponsibilityrdf:resource=''#sell_IntelCore2DuoE8500''/⟩
⟨/acnp:Role⟩
⟨acnp:Responsibility rdf:about=''#sell_IntelCore2DuoE8500"⟩
  ⟨Process:hasInput rdf:resource=''& basicinfo;BuyerInfoInput"⟩
    ⟨Process:parameterType rdf:resource=''& basicinfo;BuyerInfo"/⟩
  ⟨/Process:hasInput⟩
  ⟨Process:hasOutput rdf:resource=''& basicinfo;CPU_InfoOutput"/⟩
    ⟨Process:parameterType rdf:resource=''& basicinfo;CPU_Info''/⟩
  ⟨/Process:hasOutput⟩
⟨/acnp:Responsibility ⟩
```

Agent Specification: CPU_Seller_Agent

```
⟨acnp:Agent rdf:about=''#CPU_Seller_Agent"⟩
  ⟨acnp:hasCapabilityrdf:resource=''#sell_IntelCore2DuoE8500_Cap"/⟩
⟨/acnp:Agent ⟩
⟨acnp:Capabilityrdf:about=''#sell_IntelCore2DuoE8500_Cap"⟩
  ⟨Process:hasInput rdf:resource=''& basicinfo;BuyerInfoInput"⟩
    ⟨Process:parameterType rdf:resource=''& basicinfo;BuyerInfo"/⟩
  ⟨/Process:hasInput⟩
  ⟨Process:hasOutput rdf:resource=''& basicinfo;CPU_InfoOutput"/⟩
    ⟨Process:parameterType rdf:resource=''& basicinfo;CPU_Info"/⟩
  ⟨/Process:hasOutput⟩
⟨/acnp:Capability⟩
```

Contract Specification: CPU_Seller_Contract

```
⟨acnp:Contract rdf:about=''#CPU_Seller_Contract"⟩
  ⟨acnp:bindingAgentrdf:resource=''& cpu_seller_agent_spec;CPU_Seller_Agent"/⟩
  ⟨acnp:boundRolerdf:resource=''& cpu_seller_role_spec;CPU_Seller"/⟩
⟨/acnp:Contract⟩
⟨owl:Thing rdf:about=''& cpu_seller_role_spec;sell_IntelCore2DuoE8500"⟩
  ⟨acnp:fulfilledByrdf:resource=''& cpu_seller_agent_spec;
   sell_IntelCore2DuoE8500_Cap"/⟩
⟨/owl:Thing⟩
```

- The system should adapt to the changes of service providers, such as finding another one while a service provider unexpectedly stops servicing.

We will use the example throughout this paper to better illustrate our approach, and an experimental evaluation will be discussed in Section 5.

An agent organization is developed for the problem of assembling a computer. Table 3 shows parts of the specifications of the agent organization, in which name space declarations and some auxiliary classes are ignored. The full specification of the agent organization in our implementation consists of three organizational objectives, 11 roles for purchasing hardware components and two roles for purchasing software components.

Organizational objective Assembling_a_Computer will be achieved if its two sub-organizational objectives, Purchase_Hardware and Purchase_Software, are achieved. Purchase_Hardware and Purchase_Software are composed of responsibilities of the roles for hardware and software components through Split control construct, respectively. Each role specification contains at least a responsibility of selling an item, such as CPU_Seller_Role contains a responsibility of selling Intel Core2 Duo E8500 CPUs.

In the agent organization, a seller exhibiting on Amazon Auctions is represented as a contractor agent. For example, a seller selling Intel CPU will be represented as a contractor agent associated with an agent specification CPU_Seller_Agent containing capability sell_IntelCore2DuoE8500_Cap, where performing the capability will request the seller to ship an item to a buyer. Additionally, the contractor agent will periodically generate an agent specification containing up-to-date capabilities through monitoring the status of the items on/off sell and the stock status listed by the seller on the web site, and then update the agent specification to the middle agent of the agent organization. Each contractor agent becomes autonomous because.

- it may appear and disappear in the agent organization as motivated by the update of items on/off sell or the stock status; and
- it may have varied performance in selling items as effected by the seller that is an autonomous entity in the real world.

If a contractor agent is selected to bind a role, a contract specification will be generated and the relations between the role and the agent will be specified through object properties bindingAgent, boundRole and fulfilledBy.

We implement the agent organization in Java. Manager, middle and contractor agents are implemented as OSGi bundles running on Apache Felix OSGi platform (http://felix.apache.org). Communications among the agents are implemented based on OpenJMS (http://openjms.sourceforge.net). A UDDI repository is set up as the registry for publishing agent specifications of contractor agents through UDDI4J (http://uddi4j.sourceforge.net). The reputation repository is realized by a relational database of MySQL (http://www.mysql.com). Manager agent is implemented based on the Protege-OWL API (http://protege.stanford.edu) to be able to instantiate executions of the composite processes defined in organizational objective specifications. The algorithm of selecting a contractor agent to bind a role is implemented based on our previous work on REAL in [41].

4. ACNP: from a dynamic perspective

From a dynamic perspective, ACNP are with a set of sequence diagrams. The sequence diagrams are formulated in UML 2.0 notations [43], and the constructs of the messages are based on FIPA Communicative Acts [39].

Fig. 7 shows the top level sequence diagram of ACNP. Three types of agents are involved, including manager agent, middle agent and contractor agent. The interactions among the agents can in parallel be registering or unregistering a contractor agent or achieving an organizational objective. Through the interactions of registering or unregistering contractor agents, a middle agent will possess a set of up-to-date agent specifications for discovering contractor agents. During the interactions of achieving an organizational objective, a manager agent will select contractor agents to bind roles and compose the results of their fulfillment to achieve an organizational objective.

4.1. Service discovery through middle agent

In ACNP, the activity of service discovery is to find out web services enacted by autonomous agents that are denoted as contractor agents. Fig. 8 shows the interactions for registering or unregistering

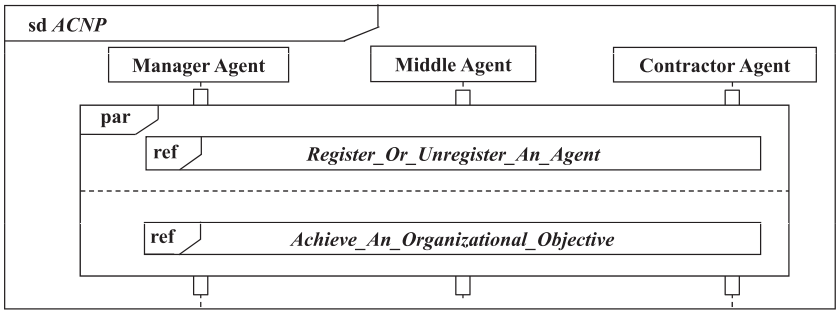

Fig. 7. Sequence diagram of ACNP at the top level.

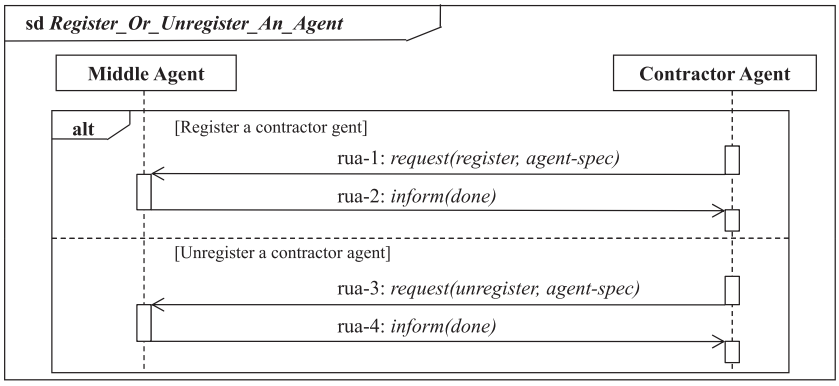

Fig. 8. Interactions for registering or unregistering a contractor agent.

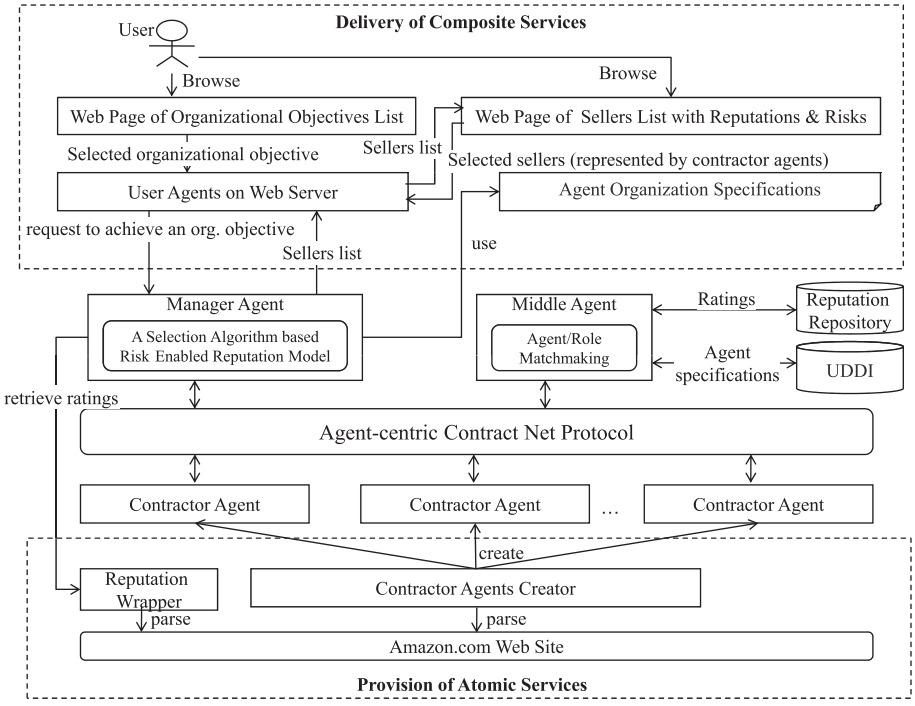

Fig. 9. A detail system architecture of the agent organization of assembling a computer.

a contractor agent. A contractor agent can send a message (rua-1) with its agent specification to make a request to a middle agent for registering. While receiving the message, the middle agent will add the agent specification to its registry and then send back a message (rua-2) to inform that the registering has been done.

If a contractor agent is going to unregister itself from the middle agent, it can send a message (rua-3) to the middle agent with its agent specification for unregistering. While receiving the message, the middle agent will remove the agent specification from its registry and then send back a message (rua-4) to inform the unregistering has been done.

Fig. 9 shows a detail system architecture of the agent organization in the running example. For the provision of atomic services, the contractor agent creator component will crawl Amazon.com web site and parse the web pages to find out sellers. Once a seller is found, a contractor agent together with an agent specification will be automatically created to represent the seller. The contractor agent will publish the agent specification to the UDDI through negotiating with the middle agent based on Register_Or_Unregister_An_Agent interactions. Besides, the manager agent will extract sellers' ratings from Amazon.com through a reputation wrapper and send the ratings to the middle agent.

For the delivery of composite services, a user can search for the organizational objectives through keyword matching to the comment parts (see Fig. 10 for the screen snapshot). All of the organizational objective specifications are hosted on a web server, and the comments parts in the specifications are published to a UDDI registry. The rule of the syntactical matching is represented as a regular expression of the form.*keyword.*, which means that several organization objectives will be matched if their comment parts' sentences contain the keyword. For example, if the user types the keywords "Assemble" and "Computer", a regular expression.*Assemble.*Computer.* will be generated and an organizational objective will be matched if its comment part sentence contains "Assemble" and "Computer". Hence, the user can select an organizational objective, such as Assemble_a_Computer, to be achieved through the web page of organizational objectives list. Once the web server receives the request, it will instantiate a user agent on a web server to request a manager agent for achieving the selected organizational objective. As the manager agent receives the request, agents negotiate through Agent-centric Contract Net Protocol. The user can browse the candidate sellers list and select sellers for the service composition through a web page.

4.2. Service composition through achieving an organizational objective

In ACNP, the activity of service composition is to compose services provided by contractor agents, and more specifically is to select contractor agents, request them to perform their capabilities, and compose and evaluate their results. A composite service is denoted as an organizational objective and is formulated as a kind of OWL-S composite process and documented in an organizational objective specification. An instance of an organizational objective is said to be achieved if a manager agent instantiates an execution of the composite process defined in the organizational objective specification and completes the execution without any failures.

Fig. 11 shows the interactions of achieving an organizational objective. The interactions start with sending a request message (aoo-1) to the manager agent for achieving an organizational objective specification from a user agent that acts as a user of the agent organization. As a manager agent receives the message, it will instantiate an execution of a composite process by interpreting the specification. During the execution, sub-interactions for the following three cases will be conducted repeatedly: (1) if a role's responsibility (an atomic process) needs to be executed, interactions for invoking an agent service will be triggered (see Section 4.2.1); (2) if a sub-organizational objective (a sub-composite process) needs to be executed, interactions for achieving an organizational objective will be iteratively triggered, that is, the manager agent will send a message (aoo-1) to itself or another manager agent that is associated with the sub-organizational objective specification; and (3) if a contractor agent requests the manager agent or is requested by the manager agent to unbind a role, interactions for changing an agent will be triggered (see Section 4.2.3).

Until no responsibilities or sub-organizational objectives need to be executed and no failures occur, the manager agent will send a message (aoo-2) with the output of the composite process execution to the user agent. If any failure occurs during the execution, the manager agent will terminate the execution and inform (aoo-3) the user agent of the failure.

In ACNP, a manager agent acts as a service composer. Fig. 12 shows the state transition diagram of a manager agent. While a manager agent receives a request of achieving an organization objective, it will be in ComposingService state. The key action in this state is to instantiate an execution of a composite process by interpreting the organizational objective specification. During the interpretation, if a responsibility (atomic process) needs to be fulfilled, it will start to find contractor agents that match the role containing the responsibility and be in MachingService state; if a sub-organizational objective (sub-composite process) needs to be

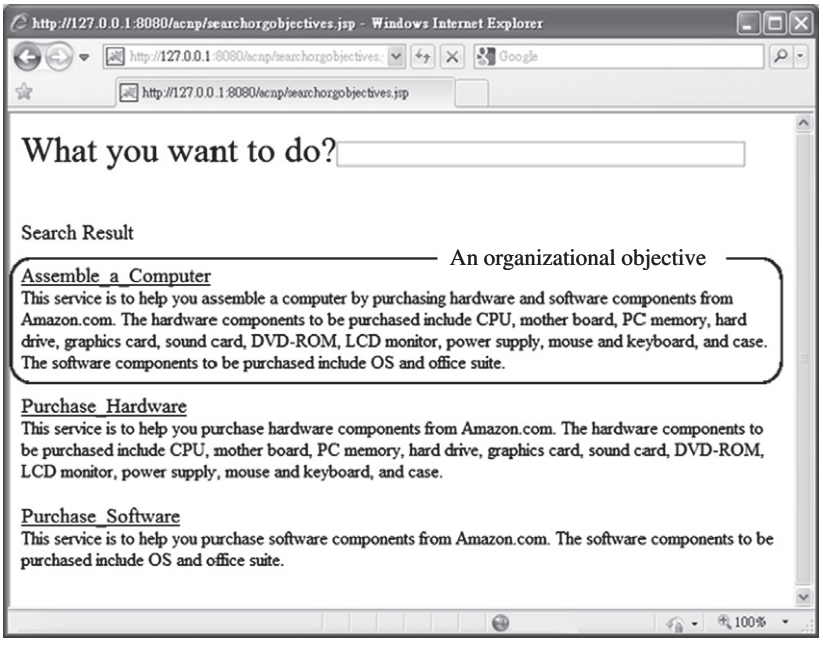

Fig. 10. Organizational objectives matched by keywords through UDDI.

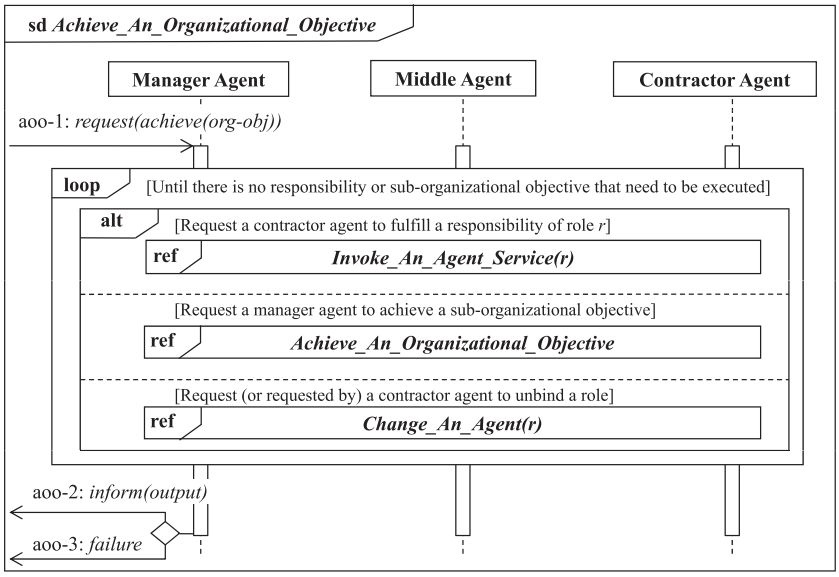

Fig. 11. Interactions for achieving an organizational objective.

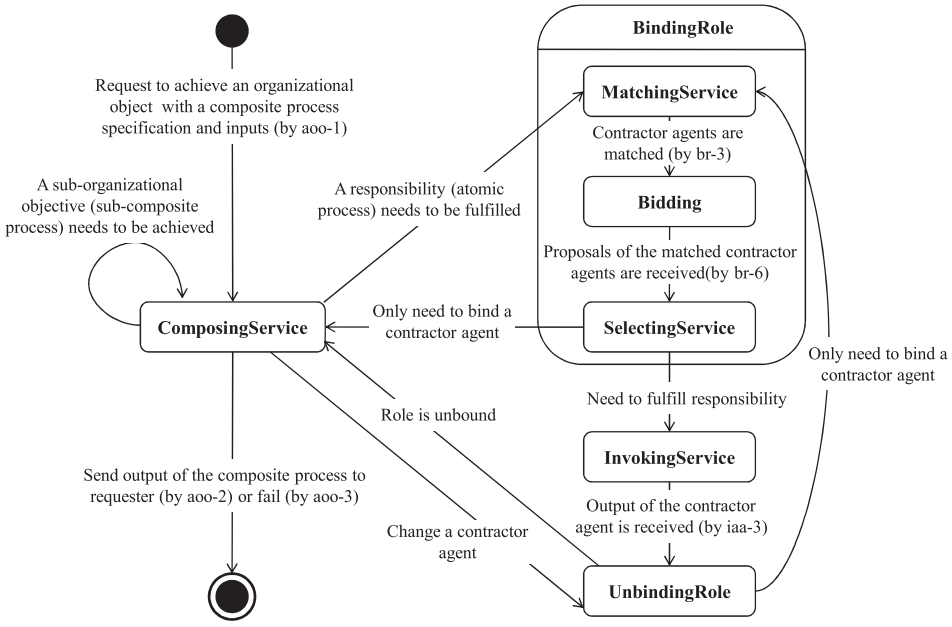

Fig. 12. State transition diagram of a manager agent.

achieved, it will start to recursively send a request of achieving the sub-organizational objective to itself or the other manager agent associated with the objective, and then be in ComposingService state.

As contractor agents are matched, it will then be in Bidding state in which it calls for proposals from the matched contractor agents. Once it received the proposals, it will then be in SelectingService state in which it selects a contractor agent to bind the role through making a contract with the agent. Thereafter, if the responsibility needs to be fulfilled, it will request the contractor agent to execute its capability that implement the responsibility, and be in InvokingService state; if it only requires a contractor agent to bind a role, it will then be in ComposingService state. Once it received the output of the contractor agent in InvokingService state, its state will shift to be UnbindingService state in which it will request the contractor agent to unbind the role through canceling the contract with the agent. After the contractor agent unbinds the role, it will be in ComposingService state. While the manager agent needs to change a contractor agent, it will request the agent to unbind the role and then start to rebind a new contractor agent. States of MatchingSerivce, Bidding and SelectingService are grouped and contained in a state called BindingRole.

4.2.1. Service invocation through fulfilling role's responsibility

In ACNP, the activity of service invocation is to request contractor agents to perform their capabilities to fulfill roles' responsibilities. If a role's responsibility should be executed during the execution of a composite process of an organizational objective specification, the interactions for invoking an agent service will be conducted (see Fig. 13).

Firstly, interactions for binding a role (see Section 4.2.2) will be conducted to discover and select a contractor agent to bind the role, meanwhile, a contract specification will be generated and held both by the manager agent and the contractor agent. Thereafter, the manager agent will request the contractor agent to fulfill the responsibility (iaa-1) with input. As the request is received, the contractor agent can perform the capability specified in the contract specification to fulfill the responsibility. The output of performing the capability will be sent back to the manager agent through a message (iaa-3). If the contractor agent fails to perform the capability (iaa-2), the manager agent will conduct the interactions for changing an agent to find another agent.

In the running example, the manager agent can request a contractor agent to fulfill a responsibility, such as sell_IntelCore2DuoE8500, and then, the contractor agent will execute the capability sell_IntelCore2DuoE8500_Cap, by which the buyer will be provided with an URL to transact with the seller.

As the manager agent receives the reply, it will perform an evaluation of the output (iaa-4). The output of the evaluation is a rating:

$$\langle ma, ca, r, t, v \rangle,$$

where ma is the manager agent; ca is the contractor agent; r is the role bound by the contractor agent; t is the evaluation time; and v is the rating value given by the manager agent. As a rating is given, the manager agent will send a message (iaa-5) with the rating to the middle agent. The middle agent will add the rating into its reputation repository.

In the running example, the manager agent will extract the seller's rating from the web site through a reputation wrapper and send the rating to the middle agent.

After that, the contractor agent will unbind the role through the interactions depicted section in Fig. 14. The interactions for unbinding a role start with a message that requests a contractor agent to cancel a contract (ur-1). The contractor agent will send a message back to the manager agent to inform it has canceled the contract (ur-2).

4.2.2. Service selection through binding role

In ACNP, the activity of service selection is to select contract agents to bind roles in order to have contracts with the agents for achieving organizational objectives. Fig. 15 shows the interactions for binding a role. At first, the manager agent queries (br-1) the middle agent for agent specifications that match a role specification. The middle agent will then perform a matchmaking action (br-2) to discover the matched agent specifications from its registry. The matchmaking action is formulated as two rules in Table 4 which ensures that the input in iaa-1 message for fulfilling a responsibility of a role can be processed by a contractor agent through sub-typing once the contractor agent is selected to bind the role, and the output produced by the contractor agent can be processed by the manager agent as well. For example, the OWL type (i.e., the parameterType property), BuyerInfo, of the input of responsibility sell_IntelCore2DuoE8500 is as the same type of the input of the capability sell_IntelCore2DuoE8500 Cap, and the OWL types of their outputs are as the same as CPU_Info, which means the responsibility and the capability are type compatible, and then the agent specification CPU_Seller_Agent is said to match the role specification CPU_Seller_Role.

While a set of matched agent specifications are discovered, the middle agent will calculate the reputations of the contractor agents associated with the agent specifications based on their ratings stored in the reputation repository. In ACNP, we adopt a risk-enabled reputation model (REAL) to serve a basis for the calculation of the reputations. Afterward, the middle agent will inform (br-3) the manager agent of the matched agent specifications and the associated contractor agents' reputations.

After the manager agent obtains the matched agent specifications, it will send call-for-proposals messages (br-4) to the contractor agents for binding the role. A contractor agent can refuse the message (br-5) or propose a proposal (br-6) to the manager agent. Note that, the content of a proposal may contain domain-specific information, such as an item's price.

In the running example, a contractor agent will propose a proposal to the manager agent if the seller it represents has items in stock. The proposal contains the information from Amazon Auctions including the item's price, the item's condition (new/used) and the seller's information, such as ratings and shipping.

If the manager agent does not receive any proposals as the deadline is due, it will send a failure message (br-7) to the user agent; else, it will perform an action (br-8) to select a contractor agent among the proposals to bind the role. The input of the selection is a set of tuples:

$$\{\langle p, rep_p, rsk_p \rangle\},$$

where p is a proposal; rep_p is the reputation value of the contractor agent that proposes p; and rsk_p is the risk value associated with rep_p.

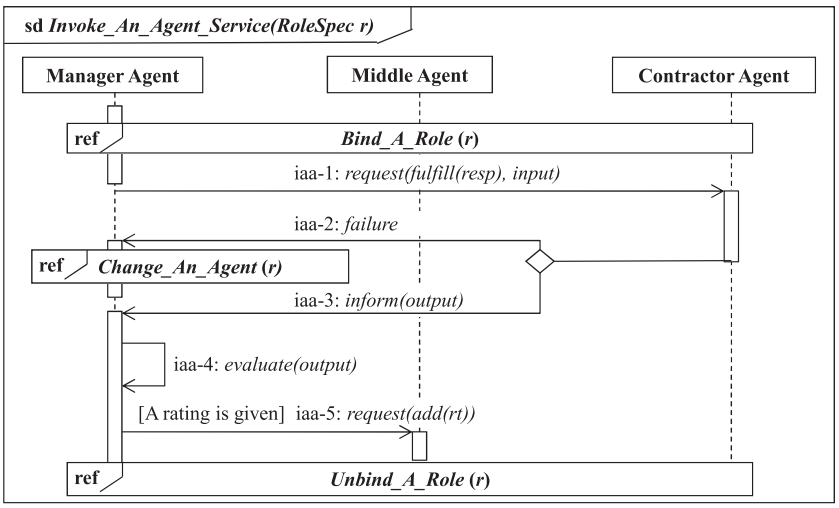

Fig. 13. Interactions for invoking an agent service.

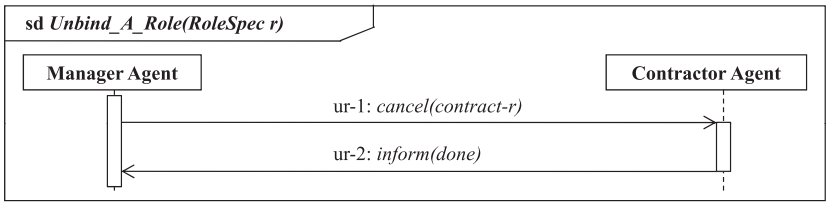

Fig. 14. Interactions for unbinding a role.

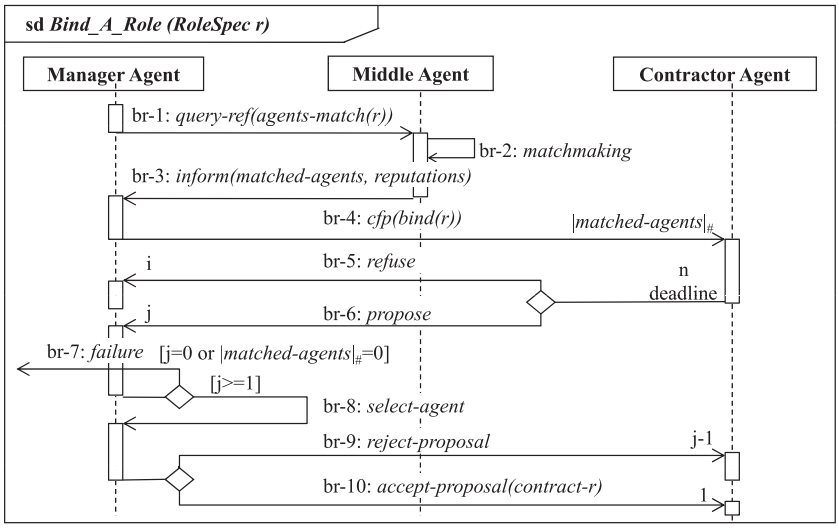

Fig. 15. Interactions for binding a role.

Table 4
Matchmaking rules for roles and contractor agents.

MM-RULE-1	An agent specification is said to match a role specification if and only if for each responsibility in the role specification, there exists at least one capability such that the responsibility and the capability are type compatible
MM-RULE-2	A responsibility and a capability are said to be type compatible if and only if (1) for each input of the responsibility *ri*, there exists at least one input of the capability *ci* such that the parameterType of *ri* is a subtype or the same type of that of *ci*, and (2) for each output of the capability *co*, there exists at least one output of the responsibility *ro* such that the parameterType of *co* is a subtype or the same type of that of *ro*

Table 5
Algorithm of selecting a contractor agent to bind a role.

INPUT	$\{\langle p, rep_p, rsk_p\rangle\}$
OUTPUT	A selected proposal
1	Determine the values of two thresholds: (1) Minimum Expected Reputation (MER), which specifies the minimum reputation value that the manager agent or the user agent expects, and (2) Maximum Acceptable Risk value (MAR), which specifies the maximum risk value that the manager agent or the user agent can accept
2	Remove proposal $p1$ if rep_{p1} is smaller than the determined MER value or rsk_{p1} is greater than the determined MAR value
3	Prioritize the remaining proposals by the following rules: (1) if $rep_{p1} < rep_{p2}$, the priority of $p1$ is set to be lower than the one of $p2$ (2) if $rep_{p1} = rep_{p2}$ and $rsk_{p1} < rsk_{p2}$, the priority of $p1$ is set to be higher than the one of $p2$ (3) if $rep_{p1} = rep_{p2}$ and $rsk_{p1} = rsk_{p2}$, the priority of $p1$ is set to be equal to the one of$p2$
4	Select the proposal with the highest priority automatically by the manager agent or to select proposal manually by system users

The algorithm of the selection action is depicted in Table 5. The first two steps are to filter out the proposals of the contractor agents that are with lower reputations or with higher risk values based on two thresholds determined by the manager agent or the user agent. The ranges of MER and MAR are the possible values of the reputations and the risks that have been discussed in Section 2.3.

The third step is to prioritize the remaining proposals according to reputation values and risk values. A proposal associated with a higher reputation value will gain higher priority. Two proposals with equal reputation values will be prioritized by the associated risk values, namely the proposal with a lower risk value will gain higher priority. The last step is to select the proposal with the highest priority automatically by the manager agent or to select proposal manually by system users.

Once a proposal is selected, the manager agent will generate and keep a contract specification specifying that the role is bound by the selected contractor agent and the relations between the responsibilities of the roles and the capabilities of the contractor agent according to MM-RULE-2. After that, the selected contractor agent will receive the contract specification (br-10), and the other contractor agents will receive reject-proposal messages (br-9). Note that, a contract specification can contain additional constraints that could be claimed in the proposal, such as expiration date, reward or punishment for violation.

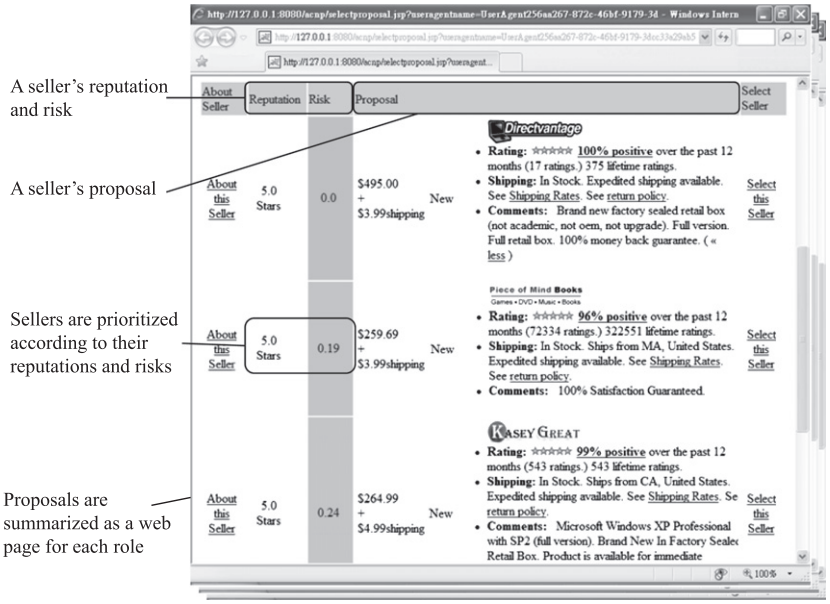

Fig. 16. Prioritized sellers with reputations, risks and proposals.

In the running example, the MER value is set to be 1 and the MAR is set to be 4, which means to consider all the sellers no matter they are with very low reputations or high risks. The manager agent will prioritize the proposals based on the selection algorithm once it receives the proposals. Thereafter, the proposals will be summarized as a web page and sent to the user agent. User agent will return the web page to the user through the web server. Fig. 16 shows a screen snapshot of the web pages that summarize the proposals for the roles in the agent organization. For each role, the user can select a contractor agent to bind the role through clicking "Select this Seller", and then the manager agent will make a contract with the selected contractor agent and request the agent to fulfill its responsibility, such as sell_IntelCore2DuoE8500.

4.2.3. Service adaption through changing agent

In ACNP, the activity of service adaption is to change contractor agents while they wants to or are requested to cancel (decommit) contracts during the execution of a composite process of an organizational objective specification. Fig. 17 shows the interactions for changing an agent. One of the following conditions will trigger a service adaption activity:

- *A contractor agent informs a manager agent that it is to unbind a role.* A contractor agent can request a manager agent that it is to unbind a role (i.e., to cancel a contract) through sending a message (ca-1) with the contract specification associated with the role to the manager agent. The manager agent can refuse the request (ca-2) or agree on canceling the contract (ca-3) according to the contract's constraints, such as expiration date, reward or punishment for violation.
- *A manager agent requests a contractor agent to unbind role.* A manager agent can also request a contractor agent to cancel a contract (ca-4). The contractor agent can refuse the request (ca-5) or agree on canceling the contract (ca-6) according to the contract's constraints.

Once the manager agent and the contractor agent agree on canceling the contact, interactions for unbinding a role will be conducted, and the manager agent will recruit another contractor agent to bind the role through conducting the interactions for binding a role.

In the running example, as the buyer wants to change the selected seller, the manager agent of the buyer will request the selected contractor agent to cancel a contract. If the contractor agent refuses the request, the manger agent will inform the user that the contract cannot be canceled; else, the contract will be canceled and then the manager agent will try to find another contractor agent through showing the user the latest prioritized sellers list excluding the seller.

5. Experimental evaluation

In this section, we present an experimental evaluation result in terms of complexity and scalability.

5.1. Complexity

Fig. 18 shows a comparison between Amazon.com and ACNP on the web pages navigation in the experiment. In Amazon.com, the user starts with inputting the components' keywords through a search page. For each component's keywords, he will get a page containing a list of keywords-matched items list. In the web page, the user can view the items' descriptions and then select an item to buy. After that, a web page containing a list of sellers that claim they can offer the selected item will be presented; in addition, each seller's information, including rating from Amazon.com and shipping, and item's price and condition will also be attached.

Note that the rating of each seller from Amazon.com is the average of the rating values over the past 12 months. In order to better understand the recent performance of each seller, the user has to browse the seller's rating (At-a-Glance) page to view the percentage of positive ratings over the past 30 days or 90 days. If the seller has a large number of ratings, the user has to browse the seller's feedbacks pages and read the feedbacks to figure out the seller's performance in more recent days. After the user makes the comparison among the sellers, he can select a seller and transact with him through a transaction page. If the user wants to change the seller, he has to navigate back to the sellers list page for a new selection.

On the other hand, the user is assisted to browse the web pages in a more efficient way by means of ACNP:

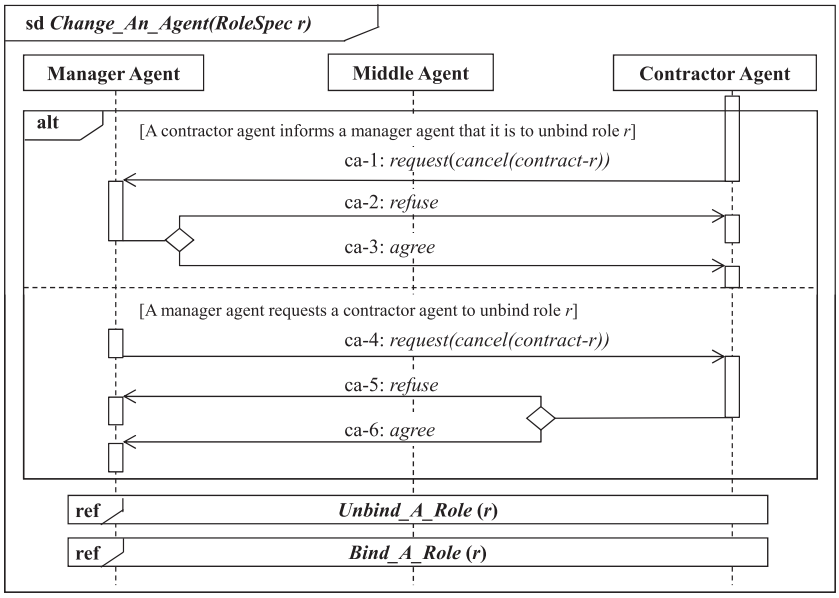

Fig. 17. Interactions for changing an agent.

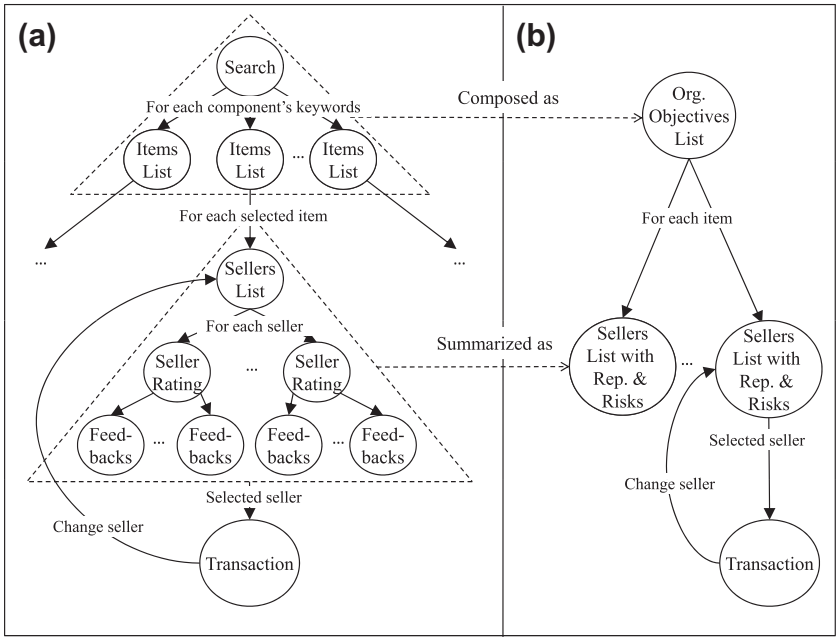

Fig. 18. A comparison of web pages navigation through (a) Amazon.com, and (b) ACNP.

- Web pages of the searched items lists are composed as a page of an organizational objectives list. Instead of having the user read over the items' descriptions and make sure their interfaces' compatibility, the user will be delivered a list of software and hardware suites defined by domain experts and documented as organizational objectives.
- For each item, web pages of the sellers' ratings and feedbacks are summarized as a page of a prioritized sellers list with reputation and risk values. Rather than providing an average rating for each seller over a fixed (i.e., 30, 90 or 365 days) time period, a reputation and a risk value are provided to indicate the recent performance of each seller, which replaces the tedious work in browsing the feedbacks pages.
- Browser will be navigated to the web page of the prioritized sellers list as the user wants to change the selected seller. In contract to having the user recapture the performance of the other sellers by re-browsing the feedbacks pages, the user is provided with an updated list of prioritized sellers.

As a result, if the user is to open n search pages to type keywords for n components, and m feedback pages of o sellers for each component, the number of web pages that need to be browsed in Amazon.com is $m \times n \times o$. In a case where $n = m = o$, the total complexity is $O(n^3)$. Through ACNP, the user only needs to browse one web page for selecting an organizational objective and one page containing a list of sellers with reputations and risks information for each component. Thus, the complexity is $1 + n = O(n)$.

Table 6
A comparison of REAL, Average Method, and REGRET in terms of their informativeness.

	REAL	Average (Adopted by Amazon Auctions)	REGRET
100 Sellers	⟨87, 12⟩	⟨68, 31⟩	⟨68, 31⟩
500 Sellers	⟨435, 64⟩	⟨345, 154⟩	⟨346, 153⟩
1000 Sellers	⟨872, 127⟩	⟨691, 308⟩	⟨697, 302⟩
5000 Sellers	⟨4363, 636⟩	⟨3455, 1544⟩	⟨3446, 1553⟩

5.2. Scalability

Table 6 shows the comparison of REAL, average method, and REGRET in terms of informativeness while scaling up from 100 to 5000 sellers.In the experiment, we randomly collected 100, 500, 1000 and 5000 sellers' ratings from Amazon.com from January 2008 to July 2010. The number of decimal places of the calculated reputation values and risk values is 1, which is also used by Amazon Auctions to present reputation values to their users.

For the case of 100 sellers, the tuple of ⟨87, 12⟩ means that given the risk value of a seller evaluated by REAL, there are 87 sellers with distinct risk values from the seller, and there are 12 sellers with the same risk value as the seller. Meanwhile, the tuple of ⟨68, 31⟩ means that given the reputation values of a seller evaluated by Average Method and REGRET, there are only 68 sellers with distinct reputation values from the seller; in other words, 31 sellers have the same reputation value as the seller being evaluated.

When scaling up to 5000 sellers, given the risk value of a seller evaluated by REAL, there are 4363 sellers with distinct risk values from the seller, and the ratio of the distinct sellers is 0.873 (i.e., 4363/5000) in contrast to the ratio of 0.87 (i.e., 87/100) in the case of 100 sellers. The ratios of the distinct sellers of Average Method and REGRET are close to 0.69 in all the cases. In conclusion, REAL is more informative than the other two approaches to enable the user to make a distinction among a large number of sellers.

6. Related work

Work in a number of fields has made their marks on our ACNP approach, including web services enacted by autonomous agents and web service compositions.

6.1. Web services enacted by autonomous agents

In [22], Payne argues that web services are rarely autonomous unless the notion of autonomy is included in the service design, and researchers are beginning to explore these notions of autonomy and autonomic behavior for web services; moreover, the distinction between agents and web services becomes increasingly blurred through providing the notions of agency, such as persistence, autonomy, and identity, to web services.

In [25], Huhns claims that a web service is not autonomous, and autonomy is a characteristic of agents and many envisioned Internet-based applications; furthermore, agents could be applied in applications that demand federated servers with multiple clients sharing results as they excel web services in managing cooperative peer-to-peer solutions.

In [24], a software agent, autonomously configured software component, is attached to a web service in their proposed Web Service Agent Framework to act as a proxy between the web service and its consumer, exposing the same interface as the service but additionally enabling other functionality, through which the agent can add value to the consumer-service interaction.

In [44,45], the authors propose an automated approach to ensuring robustness of data flow during web service composition. In the approach, the composer and providers of candidate services are represented by autonomous agents that are pre-programmed to reason over the semantic specifications of requirements and candidate services.

In [46], the authors exploit AI planning techniques for automatic service composition by treating service composition as a planning problem. Given a user's objective and a set of Web Services, a planner would find a collection of Web Services requests that achieves the objective.

In [47], the authors advance the idea that BPEL4WS is used as a specification language for expressing the initial social order of the multi-agent system that serves as a basis for workflows. However, the problem on how to select or change service providers is left to be exploited.

In [48], the authors propose a framework, called SEMMAS, for integrating agents and semantics web services. In SEMMAS, the service discovery process is based on the matching between user goal and services' expected output. However, how to compose services is not addressed, and the service selection algorithm is application-dependent.

In [49], agents are coordinated based on message exchanges through environment. In order to control the flow of executions of the agents, the priority levels of the filters for message perceptions and receptions should be carefully designed by designers.

In [50], OWL-S documents are transformed to JAM Plans (a Belief-Desire-Intention architecture), in which the service outputs are mapped to goals and the service inputs are mapped to subgoaling. A user's request is accomplished by elaborating a complex plan from primitive plans considering its goals and the world model.

In [51], OWL-S is extended for addressing the paradox problem of the broker agent, i.e., the provider is not known until the requester reveals its query, and the requester cannot query the broker until a provider is selected. Rather than using a broker approach, in our work, service request/response messages among agents are exchanged directly.

As there is a trend to develop web services enacted by autonomous agents, that is, to introduce the notion of autonomy empowered by agents into web services, this work focuses on proposing a negotiation mechanism for composing web services enacted by autonomous agents with consideration of the case that the web services become available intermittently and behave variedly over time.

6.2. Web service compositions

In [8], the authors propose a QoS-aware middleware platform, called AgFlow, for supporting quality driven compositions of autonomous web services. In their web service quality model, five generic quality criteria have been considered for both types of atomic and composite services: execution price, execution duration, reputation, successful execution rate, and availability. Based on the quality model, two web service selection approaches are described: one is local optimization by which a service is selected individually without considering the other tasks involved in the composite service; and the other is global planning by which the global quality of the composite service will be optimized through integer programming.

In MAIS [6], web service selection problem is formulated as a mixed integer linear programming problem. The goal of the approach is to maximize the aggregated value of multiple quality dimensions through considering all of the possible execution scenarios of a composite service. If a feasible solution for the problem does not exist, negotiation for determining new quality values will be performed.

In [52], a service level agreement negotiation framework is proposed for service composition provision, in which the service con-

Table 7
Comparison of web service composition methods.

	ACNP	AgFlow [8]	MAIS [6]
Service discovery	Service matching based on the notion of agent roles	UDDI	Considering signature of operation and local quality constraints
Service selection (with respect to service reputations)	Risk-enabled reputation model	Average approach	Average approach
Service adaption	Unbinding an atomic service from a composite service should be negotiated based on contract	A composite service execution is re-planned once new candidate services offering better QoS, or existing services raising their QoS	Re-optimization is triggered once a better execution plan is found at runtime

sumer is represented by a set of (negotiator) agents who negotiate quality of service constraints with the service providers for services in the composition; and a decision making model is included to help select service providers based on concession and trade-off strategies. However, how to discover the candidate services and adapt to changing services is less addressed.

We compare our approach with AgFlow and MAIS along with three dimensions (see Table 7): service discovery, service selection and service adaption.

- *Service discovery:* In AgFlow, when an instance of a composite service is initiated, the execution planner contacts the service broker to search for candidate atomic services. The service broker is implemented based on UDDI registry, and WSDL is adopted to specify services. Every Web service is assigned to a tModel, which provides a semantic-based classification of a service's functionality and a canonical description of its interface. In MAIS, candidate services are selected by considering the signature of the operation to be performed and the specified local quality constraints for the task. In ACNP, a matchmaking mechanism for finding candidate services is presented based on the concept of agent roles, formulated as Role Specifications, which are widely adopted to characterize service capabilities in the agent research community. UDDI is used as a repository to manage the role specifications.
- *Service selection:* Both AgFlow and MAIS consider the global optimization problem in selecting services based on multi-dimensional QoS constraints. However, ACNP focuses more on dealing with local optimization problem based on reputation. With respect to applying service reputations on selecting services, average measurement is adopted by AgFlow and MAIS to calculate service reputations. In ACNP, a risk-enabled reputation model (REAL) is introduced to calculate service reputations. Our approach better reflects the recent performance of a service provider than other related approaches in the literature [30,31] (see Section 2.3).
- *Service adaption:* In AgFlow, a composite service execution is re-planned when new candidate services offer better QoS than the existing services, or when existing services raise their advertised QoS. In MAIS, re-optimization is triggered when a better execution plan is found at runtime, and services will be changed for maximizing overall QoS constraints. In ACNP, a composite service can unbind an atomic service and bind another one for playing a role. In addition, a negotiation is required while unbinding an atomic service from a composite service, that is, both the composite and the atomic services are entitled to request, refuse or agree on the unbinding based on the contract. In other words, the atomic service will not be immediately unbound by the composite service without any negotiations, by which atomic web services enacted by autonomous agents can better manage their internal state as they participate in service compositions.

7. Conclusion

It is increasingly recognized that web services would become more autonomous by introducing diverse agent technologies to better constitute more complex systems in open and dynamic environments. As web service technologies are best exploited by composite services, it is imperative to devise mechanisms for composing services of autonomy. This paper proposes an extension to Contract Net protocol, called Agent-centric Contract Net Protocol (ACNP), as a negotiation mechanism for composing web services enacted by autonomous agents through infusing the notion of agent organizations into service compositions. The key contributions of ACNP are threefold:

- Web services can register or unregister to a middle agent and will be matched dynamically by a middle agent with a matchmaking mechanism for a service composition.
- Web services with variant performance can be selected for a service composition through a selection algorithm based on risk-enabled reputation model (REAL) embedded in a manager agent.
- Web services can be adapted through negotiations based on contracts, by which a service composer is entitled to rebind a new atomic service dynamically, and the atomic services are eligible to refuse or agree on unbinding from a service composition.

Additionally, an implementation of ACNP is developed and has been used to build an agent organization for assisting users to complete online purchasing on assembling a computer via a more efficient fashion. Our future plan consists of several tasks: (1) to investigate the possibility of integrating agent organization specifications with the notion of organizational rules based on SWRL (Semantic Web Rule Language), and (2) to enhance role specifications with the feature of resources.

Acknowledgment

This research was sponsored by National Science Council (Taiwan) under the grant NSC 97-2221-E-008-039-MY3 and the grant NSC 100-2631-H-008-006.

References

[1] T. Andrews, F. C, et al., Business Process Execution Language for Web Service (BPEL4WS) 1.1, Technical Report, BEA Systems and International Business Machines Corporation and Microsoft Corporation and SAP AG and Siebel Systems, 2003.
[2] W3C Member Submission, OWL-S: Semantic Markup for Web Services <http://www.w3.org/Submission/OWL-S/>, 2004.
[3] W3C Member Submission, Web Service Modeling Ontology (WSMO) <http://www.w3.org/Submission/WSMO/>, June 2005.
[4] W3C Member Submission, Semantic Web Services Language (SWSL) <http://www.w3.org/Submission/SWSF-SWSL/>, September 2005.

[5] J. Nitzsche, T. van Lessen, D. Karastoyanova, F. Leymann, Bpel for semantic web services (bpel4sws), Lecture Notes in Computer Science 4805 (2007) 179–188.
[6] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Transactions on Software Engineering 33 (6) (2007) 369–384.
[7] S.-Y. Hwang, E.-P. Lim, C.-H. Lee, C.-H. Chen, Dynamic web service selection for reliable web service composition, IEEE Transactions on Services Computing 1 (2) (2008) 104–116.
[8] L. Zeng, B. Benatallah, A.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Qos-aware middleware for web services composition, IEEE Transactions on Software Engineering 30 (5) (2004) 311–327.
[9] J. Harney, P. Doshi, Selective querying for adapting web service compositions using the value of changed information, IEEE Transactions on Services Computing 1 (3) (2008) 169–185.
[10] G. Canfora, M.D. Penta, R. Esposito, M.L. Villani, A framework for qos-aware binding and re-binding of composite web services, Journal of Systems and Software 81 (10) (2008) 1754–1769.
[11] J. Lee, S.-P. Ma, S.-J. Lee, Y.-C. Wang, Y.-Y. Lin, Dynamic service composition: a discovery-based approach, International Journal of Software Engineering and Knowledge Engineering 18 (2) (2008) 199–222.
[12] Y. Charif-Djebbar, N. Sabouret, Dynamic service composition and selection through an agent interaction protocol, in: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2006.
[13] M. Luck, R. Ashri, M. d'Inverno, Agent-Based Software Development, Artech House, 2004.
[14] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American (2001) 29–37.
[15] C.E. Gerede, O.H. Ibarra, B. Ravikumar, J. Su, Online and minimum-cost ad hoc delegation in e-service composition, in: Proceedings of the 2005 IEEE International Conference on Services Computing (SCC 2005), 2005, pp. 103–112.
[16] A. Marconi, M. Pistore, P. Traverso, Implicit vs. explicit data-flow requirements inweb service composition goals, in: Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC 2006), 2006, pp. 459–464.
[17] M. Pistore, A. Marconi, P. Bertoli, P. Traverso, Implicit vs. explicit data-flow requirements inweb service composition goals, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, 2005, pp. 1252–1259.
[18] D. Berardi, D. Calvanese, G.D. Giacomo, M. Lenzerini, M. Mecella, Automatic composition of e-services that export their behavior, in: Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC 2003), LNCS 2910, 2003, pp. 43–58.
[19] X. Fu, T. Bultan, J. Su, Analysis of interacting bpel web services, in: Proceedings of the 13th International World Wide Web Conference (WWW 2004), 2004, pp. 621–630.
[20] J. Lee, F. Liu, Y.-C. Wang, W. Chiang, Possibilistic petri nets as a basis for agent service description language, Fuzzy Sets and Systems 144 (1) (2004) 105–126.
[21] J. Lee, F. Liu, W. Chiang, Modeling uncertainty reasoning with possibilistic petri nets, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 33 (2) (2003) 214–224.
[22] T.R. Payne, Web services from an agent perspective, IEEE Intelligent Systems 23 (2) (2008) 12–14.
[23] S. Munroe, T. Miller, R.A. Belecheanu, M. Pěchouček, P. Mcburney, M. Luck, Crossing the agent technology chasm: lessons, experiences and challenges in commercial applications of agents, The Knowledge Engineering Review 21 (4) (2006) 345–392.
[24] E. Maximilien, M. Singh, Toward autonomic web services trust and selection, in: Proceedings of 2nd Internaltional Conference on Service Oriented Computing (Icsoc 04), 2004, pp. 212–221.
[25] M. Huhns, Agents as web services, IEEE Internet Computing 6 (4) (2002) 93–95.
[26] S.D. Ramchurn, D. Huynh, N.R. Jennings, Trust in multi-agent systems, The Knowledge Engineering Review 19 (1) (2004) 1–25.
[27] N. Jennings, On agent-based software engineering, Artificial Intelligence 117 (2) (2000) 277–296.
[28] R.G. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, IEEE Transactions on Computers 29 (12) (1980) 1104–1113.
[29] J. Lee, S.-J. Lee, H.-M. Chen, Dynamic role binding with agent-centric contract net protocol in agent organizations, in: Proceedings of 2008 IEEE International Conference on Systems, Man and Cybernetics, 2008, pp. 636–643.
[30] J. Sabater, C. Sierra, REGRET: A reputation model for gregarious societies, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2002, pp. 1104–1113.
[31] Amazon Auctions <http://www.amazon.com/>.
[32] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, S. Weerawarana, The next step in web services, Communications of the ACM 46 (10) (2003) 29–34.
[33] M.P. Papazoglou, D. Georgakopoulos, Service-oriented computing: introduction, Communications of the ACM 46 (10) (2003) 24–28.
[34] J. Lee, S.-P. Ma, A. Liu, Service Life Cycle Tools and Technologies: Methods, Trends and Advances, IGI Grobal, 2011.
[35] ESSI WSML Working Group Members, The Web Service Modeling Language WSML <http://www.wsmo.org/TR/d16/d16.1/v1.0/>, August 2008.
[36] W3C Recommendation, OWL Web Ontology Language Overview <http://www.w3.org/TR/owl-features/>, 2004.
[37] Foundation for Intelligent Physical Agents, FIPA Contract Net Interaction Protocol Specification <http://www.fipa.org/specs/fipa00029/SC00029H.html>, 2002.
[38] Foundation for Intelligent Physical Agents, FIPA ACL Specifications <http://www.fipa.org>, 2002.
[39] Foundation for Intelligent Physical Agents, FIPA Communicative Act Library Specification <http://www.fipa.org/specs/fipa00037/SC00037J.html>, 2002.
[40] J. Odell, H.V.D. Parunak, B. Bauer, Representing agent interaction protocols in uml, in: Agent-Oriented Software Engineering, 2001, pp. 121–140.
[41] J. Lee, S.-J. Lee, REAL: A risk-enabled reputation model for electronic commerce, in: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, 2008, pp. 619–624.
[42] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, 1996.
[43] Object Management Group, UML (Unified Modeling Language) <http://www.omg.org/spec/UML/2.2/Superstructure>, 2009.
[44] F. Lecue, U. Wajid, N. Mehandjiev, Negotiating robustness in semantic web service composition, in: Proceedings of Seventh IEEE European Conference on Web Services (ECOWS 2009), 2009.
[45] N. Mehandjiev, F. Lecue, U. Wajid, Provider-composer negotiations for semantic robustness in service compositions, in: Proceedings of the seventh International Joint Conference on Service Oriented Computing (ICSOC 2009), 2009, pp. 205–220.
[46] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, Htn planning for web service composition using shop2, Journal of Web Semantics 1 (4) (2004) 377–396.
[47] P.A. Buhler, J.M. Vidal, Towards adaptive workflow enactment using multiagent systems, Information Technology and Management 6 (1) (2005) 61–87.
[48] R.M.-B.F. García-Sánchez, R. Valencia-García, J.T. Fernández-Breis, An ontology, intelligent agent-based framework for the provision of semantic web services, Expert Systems with Applications 36 (2) (2009) 3167–3187.
[49] F. Balbo, V. Monfort, Improving web services adaptability thanks to a synergy between aspect programming and a multi-agent middleware, in: Web Intelligence, 2009, pp. 422–425.
[50] I.-C. Kim, H. Jin, An agent system for automated web service composition and invocation, Lecture Notes in Computer Science 4277 (2006) 422–425.
[51] M. Paolucci, J. Soudry, N. Srinivasan, K. Sycara, A broker for owl-s web services, in: First International Semantic Web Services Symposium, AAAI Spring Symposium Series, 2004.
[52] J. Yan, R. Kowalczyk, J. Lin, M.B. Chhetri, S.K. Goh, J. Zhang, Autonomous service level agreement negotiation for service composition provision, Future Generation Computer Systems 23 (6) (2007) 748–759.