Case Study on Web Service Composition Based on Multi-Agent System

Shanliang Pan
Department of Computer Science and Technology, Ningbo University, China
PanShanLiang@gmail.com

QinJiao Mao
Department of Computer Science and Technology, Xi'an Jiaotong University, China
maoqinjiao@stu.xjtu.edu.cn

***Abstract*—Rapid development of the Internet and increasing number of available Web services has generated a need for tools and environments facilitating automated composition of atomic Web services into more complex Web processes. However ,reasoning optimization and utilization in such AI related solutions is still an open problem .In this paper, we proposed a novel multi-agent based semantic web service composition model(SWSCPA) which exploits the relationships among different service consumers and providers, together with the corresponding optimization approach to strengthen the effectiveness of Web service composition. We argue that agents and web services are distinct. In our work, agents provide a distinctive additional capability in mediating user goals to determine service invocations. Through the model, an optimization method was proposed based on the substitute relationship and the dependency relationship. The case study and experimental analysis demonstrates the capability of the proposed approach.**

***Index Terms*—Web Service, Service Composition,Agent**

I. INTRODUCTION

As an architectural style for building software applications using service components available in a network, services-oriented architecture(SOA) has made a major impact on distributed computing research. SOA is usually realized through Web services, which is defined as the self-contained, self-describing, modular application that provides business functionality across the Web. Accordingly, the ability to efficiently and effectively integrate an appropriate set of service components to realize a new service that fulfills the users' request is the essential feature of Web services. In the past decade, substantial research effort has been devoted to automated Web services composition systems. Most existing research work falls into the categories of cross-enterprise workflow composition or AI planning.

In this context planning has proved to be one of the most promising techniques for the automated composition of Web services. Several works in planning have addressed different aspects of this problem, see, e.g., [1,2]. In these works, automated composition is described as a planning problem: services that are available and published on the Web, the component services, are used to construct the planning domain, composition requirements can be formalized as planning goals, and planning algorithms can be used to generate composed services. These works share the conception of services as stateless entities, which enact simple query – response protocols.

An even more difficult challenge for planning is the automated composition of Web services at the process level, the composition of component services that consist of stateful business processes, capable to establish complex multi-phase interactions with their partners. In the large majority of real cases, services cannot be considered simply as atomic components, which, given some inputs, return some outputs. On the contrary, in most application domains, they need to be represented as stateful processes that realize interaction protocols which may involve different sequential, conditional, and iterative steps. For instance, we cannot in general interact with a "flight booking" service in an atomic step. The service may require a sequence of different operations including an authentication, a submission of a specific request for a flight, the possibility to submit iteratively different requests, acceptance (or refusal) of the offer, and a payment procedure.

However, although these applications work appropriately, it has been identified that many of them are not capable to jointly face several problems related to Web service composite context, such as: (i) It cannot be expected to have all relevant information on the system local knowledge base; for that reason, the planner, when having incomplete information, will need to collect some information with the purpose of solving composition problem; (ii) Web services possess a few limitations, these limitations include inability to perform effective Transaction Management, automatic Service Composition and lack of Scalability and Robustness. and (iii) The composition service only can find one solution for the goal, but it may be not the best one .

doi:10.4304/jsw.8.4.900-907

This paper proposes a Semantic Web Service Composition Planner Agent (SWSCPA), which fall in the realm of AI planning. In order to overcome the current service composition's shortcomings, SWSCPA implemented in JADE, it looks the process of service composite as a planning problem, and the process model underlying the composite service identifies the functionalities required by the services to be composed and their interactions, component services that are able to provide the required functionalities are then associated to the individual tasks of the composite services. Finally, SWSCPA obtain an optimized plan based on service quality model.

The remainder of this paper is organized as follows. Section 2 illustrates briefly our approach to the integration of Agent and Semantic Web services. Section 3 then discusses our web service quality model for web service. Afterwards, the main idea of web service composition as a planning problem is discussed (Section 4). Section 5 discusses the architecture of SWSCPA, which integrate of semantic web services into agent systems, followed by a concrete example in Section 6.

II. The Mapping Between Agents And Web Services

Agents and web services are compatible. For example, Web service consists of a WSDL file that describes that service, SOAP which specifies the messaging protocol and UDDI enables discovery. Agent has a model of its environment, can communicate with peers and take appropriate actions, can look for other suitable agents. When combing agents and services, BPEL and other WS-Policy specification provides a model, UDDI provides a mechanism for discovery, SOAP offers a standardized mechanism of message exchange. Finally WSDL allows agent to invoke methods of service [4].

As the most important element in agent, the agent action is responsible for changing the agent's state, and can be implemented logically or through procedural code. In general, the agent action can be logically formalized in the form of "action:object". For example, "Start:Reasoner'' means the action of starting the reasoner, and ''Send:Message'' means the action of sending a message. For the specification of action, each agent action must have specification of the preconditions that must be satisfied for the action to be executable, and specification of the effects that will be satisfied after the action is applied on the agent's state.

JADE (Java Agent Development Environment) is a FIPA compliant agent development environment which facilitates the implementation of multi-agent systems. Since Web services middleware has been integrated into JADE, agents implemented in JADE can exploit Web services as computational resources. A Web service can be published as a JADE agent service and an agent service can be symmetrically published as a Web service endpoint. Invoking a Web service is just like invoking a normal agent service. In addition, Web services' clients can also search for and invoke agent services hosted within JADE containers.

So, Our work is focus on the work process of the composite service agent based on JADE. We proposed an service composition agent, which allow an automated Web service composition to construct powerful, robust service network by binding together a number of collaborated agent-based Web services.

In this paper, SWSCPA will generate an optimized service composition plan, which is based on the service quality model. We described an extensible multi-dimensional web service quality model below first.

III. The Model Of Web Service Quality

In a Web environment, multiple web services may provide similar functionalities with different non-functional property values. Such web services will typically be grouped together in a single community. To differentiate the members of a community during service selection, their non-functional properties need to be considered. For this purpose, we adopt a web services quality model based on a set of quality criteria.

An extensible QoS model is used to deal with dynamic QoS values and various kinds of QoS in web service profile. The extensible QoS model, $QoS = \{q_1, q_2, ..., q_n\}$, represents the set of QoS criteria, where q_n presents single QoS information [5]. QoS criteria for different domains may be different. To be more generic and precise, we consider 6 criteria: performance, cost, reliability, availability, reputation and fidelity, it can be represented by using the extensible QoS model as follows: $QoS(S) = \{q_{per}, q_{cost}, q_{rel}, q_{avail}, q_{rep}, q_{fid}\}$:

Performance: The performance q_{per} is the time duration (turn round time) from a request being sent, to when the results are received.

Cost: It refers to the amount of money that the consumer pays for using a service, q_{cost}.

Reliability: The reliability q_{rel} is the probability that the requested service is working without a failure within a specified time frame [6].

Availability: The availability q_{avail} is the quality aspect of whether the service is present or ready for immediate use [6].

Reputation: The reputation q_{rep} is the criterion in measuring total trustworthiness of a service.

Fidelity: The fidelity q_{fid} is the average marks that are given by different consumers to the same QoS criterion.

There are many approaches to collect values of quality metrics, for instance, we can get it directly from the service descriptions, calculation of a quality value based on the defining expression in the service description, or Collection through active monitoring. we get values of these qualities from the OWL-S profile.

Quality of service (QoS) is often used to compute ranking values for comparing Web services having similar functionalities. A Web service ranking method is basically based on two factors: a QoS model and a ranking algorithm, we will discuss this ranking method in detail in 5.4

Next, we will describe the main idea of the paper, which is look web service composition as a planning problem.

IV. Web Service Composition As A Planning Problem

A. Web Service Composition using AI Planning

In general, a planning problem can be described as a five-tuple $< S, S_0, G, A, \tau >$, where S is the set of all possible states of the world, $S_0 \subset S$ denotes the initial state of the world, $G \subset S$ denotes the goal state of the world the planning system attempts to reach, A is the set of actions the planner can perform in attempting to change one state to another state in the world, and the translation relation $\tau \subset S \times A \times S$ defines the precondition and effects for the execution of each action. In order to employ planning, a web service composition problem must be reflected to a planning problem. The desired outcome of the complex service is described as a goal state, while simple web services play the role of planning operators, or actions. The planner then will be responsible for finding an appropriate plan, i.e. an appropriate sequence of simple web service invocations, to achieve the goal state [7]. The produced plan will eventually constitute the description of the complex service. An important benefit of the planning approach in general is the exploitation of knowledge that has been accumulated over years of research on the field of planning. Therefore, well known planning algorithms, techniques and tools can be used to the advantage of efficient and seamless web service composition.

The representation of planning problems has been a concern since 1971 when Fikes and Nilsson developed the STRIPS language. From this time on, other researchers have proposed planning problem representation languages based on STRIPS aiming at developing a more expressive language for real planning problems. In 1998, the Artificial Intelligenge Planning groups made an attempt to standardize a language for real planning problem description proposing PDDL – Planning Domain Description Language. PDDL has been used as the standard language in international planning competitions allowing planning problems to be represented in a comparable notation and planner performance to be evaluated. In its version 2.2, PDDL currently allows planning problem modellers to specify actions with duration and deterministic unconditional exogenous events, which are facts that will become true or false at time points that are known to the planner in advance, independently of the actions that the planner chooses to execute.

A strong interest to Web service composition from AI planning community could be explained roughly by similarity between OWL-S and PDDL representations. Moreover, since OWL-S has been strongly influenced by PDDL language, mapping from one representation to another is straightforward (as long as only declarative information is considered). When planning for service composition is needed, OWL-S descriptions could be translated to PDDL format [8]. Then different planners could be exploited for further service synthesis.

B. Translating OWL ontologies to PDDL

Web service ontologies, initial and goal ontologies about state world are translated to a domain and a problem under Artificial Intelligence Planning approach; this requires transferring specifications of ontologies to PDDL language. The Class mapping (*Class*) and properties (*Property*) included in OWL ontologies of the initial and final state of PDDL types (*Type*) and Predicates (*Predicate*) are necessary. Web services are mapped onto PDDL actions, in which the model of an action represents a Web service. Thus the main relations of conversion among (OWL-S) specifications and their corresponding representation in PDDL are summarized in Fig 1[8].

SWSCPA converts the domain ontology and service descriptions in OWL and OWL-S, respectively, to equivalent PDDL problem and domain descriptions using its OWLS2PDDL converter middleware.

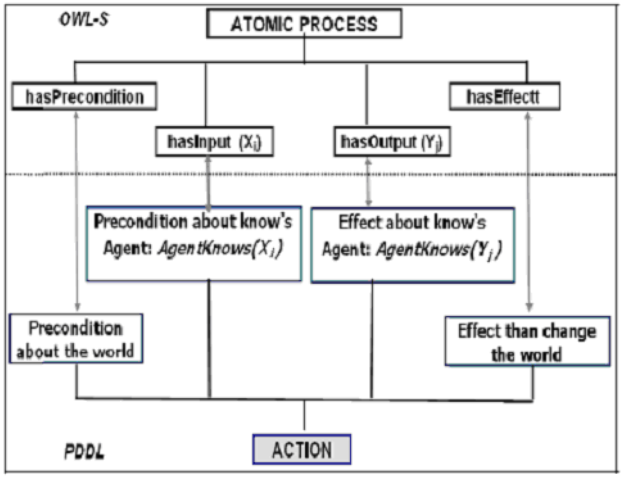

Fig 1: Convertion from OWL-S to PDDL [13]

C. The AI planner Xplan

Xplan is a heuristic hybrid FF planner based on the FF planner developed by Hoffmann and Nebel [9]. It combines guided local search with relaxed graph planning, and a simple form of hierarchical task networks to produce a plan sequence of actions that solves a given problem. If equipped with methods, XPlan uses only those parts of methods for decomposition that are required to reach the goal state with a sequence of composed services.

To use Xplan for semantic Web-Service composition, Xplan is complemented by a conversion tool that

converts OWLS service descriptions to corresponding PDDL descriptions that are used by Xplan as input to plan a service composition that satisfies a given goal [10]. In contrast to HTN planners, Xplan always finds a solution if it exists in the action/state space over the space of possible plans, though the problem is NP-complete. Xplan also includes a re-planning component to flexibly react to changes in the world state during the composition planning process. Together the implementations of Xplan and OWLS2PDDL converter make up our Semantic Web Service Composition Planner Agent(SWSCPA).

V. The Architecture Of The Semantic Web Service Planning Agent

Since we can look service composition problem as a planning problem, Planning problems involve a set of initial states, a set of goals and the corresponding actions that contribute to achieve these goals. A planner agent is an agent responsible for solving planning problems. Proposed planning agent's characteristics are follow the following design principles: (i) *concurrence*: Several processes, such as environment monitoring, execution, and planning are carried out in a concurrent way; (ii) *Reactivity* of the system is favored by an architecture organized by levels, in which highest levels show a more complex behavior and represent information with a higher abstraction level.

The SWSCPA (Planning agent) architecture consists of four main modules (see Figure 2): (i) *A translator of OWL-S [11]specification to PDDL* [12], which translates initial domain and goal state ontologies, together with service descriptions respectively implemented in OWL and OWL-S, in a domain specification and its corresponding planning problem in PDDL; (ii) *an environment model*, which allows planner to have a certain knowledge about external environment. This knowledge is expressed through facts and numerical variables; (iii) Xplan, which tries to find out a plan to reach objectives; (iv) Optimization module, which takes initial plan file as input and produce a optimized plan based on the QoS selection algorithm.

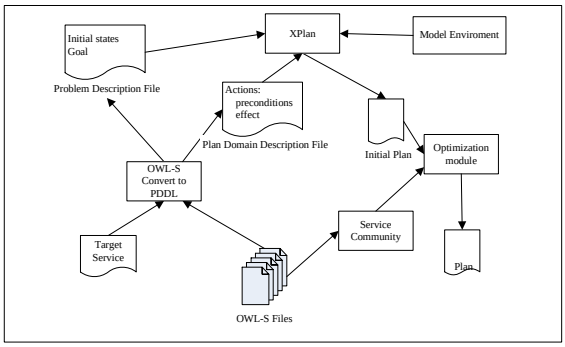

Fig 2: Semantic Web Service Composition Planner Agent(SWSCPA)

A. OWL-S Convert to PDDL

To our knowledge, there are several proposals, e.g. OWL-S [14], WSMO / WSML [15] and WSDL-S [16], to implement semantic Web services. In SWSCPA, OWL-S has been chosen for describing Web services. To describe initial and goal state, the agent use OWL ontologies. As most classical planners, proposed planning agent needs a description of both domain and problem through a modeling language. For that purpose, PDDL language has been chosen as it is currently a planning domain description standard.

SWSCPA takes a set of available OWL-S 1.1 services, related OWL ontologies, and a planning request (goal) as input, and returns a planning sequence of relevant OWL-S services that satisfies the goa, for this purpose, it first converts a given domain ontology and service descriptions in OWL and OWL-S 1.1, respectively, to equivalent PDDL problem and domain descriptions using an integrated OWLS2PDDL converter[17]. The domain description contains the definition of all types, predicates and actions, whereas the problem description includes all objects, the initial state, and the goal state. Both descriptions are then used by the AI planner XPlan to create a plan in PDDL that solves the given problem in the actual domain. An operator of the planning domain corresponds to a service profile in OWL-S, while a method is a special type of operator for fixed complex services that SWSCPA may use during its planning process.

B. Generator An Initial Plan by Xplan

We use Xplan ,as described in [10], to create a plan. Xplan use the Microsoft MSXML Parser for reading PDDXML definitions and generating plans in XML format. The Xplan system consists of one XML parsing module, and following preprocessing modules. First, required data structures for planning are created and filled, followed by the generation of the initial connectivity graph and goal agenda. The core planning modules concern the heuristically relaxed graph-plan generation and enforced hill-climbing search. After the domain and problem definitions have been parsed, Xplan compiles the information into memory efficient data structures. A connectivity graph is then generated and efficiently realized by means of a look up table, which contains information about connections between facts and instantiated operators, as well as information about numerical expressions which can be connected to facts. Xplan uses an enforced hill-climbing search method to prune the search space during planning, and a modified version of relaxed graph-planning that allows to use (decomposition) information from hierarchical task networks during the efficient creation of the relaxed planning graph. Figure 3 shows a fragment of the plan description produced by Xplan, i.e., a sequence of actions, that is the composed sequence of corresponding OWL-S services, that can be executed by the agent.

```
- <step number="6" name="http://127.0.0.1/health-
  scallops/services/FindNearestAirport.owl#FindNearestAirportService" id="88" new="false"
  ps="7">
  <param name="http://127.0.0.1/health-
    scallops/services/FindNearestAirport.owl#Address">http://www.owl-
    ontologies.com/Ontology1.owl#DepartureHospitalAddress</param>
  <param name="http://127.0.0.1/health-
    scallops/services/FindNearestAirport.owl#AirportAddress">http://www.owl-
    ontologies.com/Ontology2.owl#DepartureAirportAddress</param>
  <param name="http://127.0.0.1/health-
    scallops/services/FindNearestAirport.owl#Airport">http://www.owl-
    ontologies.com/Ontology2.owl#DepartureAirport</param>
- <effects>
  - <effect id="88">
    - <adds>
      - <fact name="agentHasKnowledgeAbout" id="7">
          <param>http://www.owl-
            ontologies.com/Ontology2.owl#DepartureAirportAddress</param>
        </fact>
```

Fig 3: a Fragment of the plan description produced by Xplan

After Xplan produce a plan, it will generator a sequence of actions, that is the composed sequence of corresponding OWL-S services, as Fig 4(a). SWSCPA looks the actions as tasks, and every task connect to a corresponding OWL-S service, as Fig 4 (b). We definite a plan as below:

Definition (plan) A set of pairs $p = \{\langle t_1, s_{i1}\rangle, \langle t_2, s_{i2}\rangle, \cdots, \langle t_N, s_{iN}\rangle\}$ is a plan iff:

- $\{t_1, t_2, ..., t_N\}$ is the set of tasks.
- For each 2-tuple $< t_i, s_{ij} >$ in p, service s_{ij} is assigned of task t_i.
- t_i is a direct successor of one of the task in $\{t_1, t_2, ..., t_{i-1}\}$
- t_i is not a direct successor of one of the task in $\{t_{i+1}, t_{i+2}, ..., t_N\}$

Fig 4(b) provides aggregation functions for a composite service using plan $p = \{\langle t_1, s_1\rangle, \langle t_2, s_2\rangle, \cdots, \langle t_N, s_N\rangle\}$

We call the plan, which generated by Xplan, as initial plan. As we know, maybe there is a set of candidate Web services s_{ij} that are available to for each task t_i. So, the initial plan may be not a best plan for the goal. In our Service composition framework, System provide a service quality model(described at 3), and each Web service s_{ij} associated with a quality vector. Based on these service quality vector and service community information, the optimization module of the SWSCPA can adjust initial plan to a optimize plan.

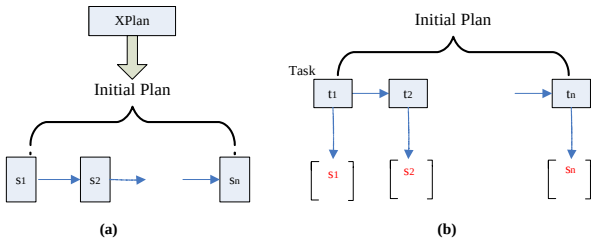

Fig 4 an Initial Plan Generated by SWSCPA

C. Web Service Community

The concept of web service community addresses the issue of composing a large and changing collection of web services, Service communities provide descriptions of a desired functionality without of referring to any actual service.

The set of members of a community can be fixed when the community is created, or it can be determined through a registration mechanism, thereby allowing service providers to join, quit, and reinstate the community at any time. When a community receives a request to execute an operation, this request is delegated to one of its current members. In our Service composition framework, every web service agent can register to any communities, and community also provide a mechanism to search special web services.

D. Optimization of the Planning

Initial plan(described in 5.2) produce a process model of a composite service ,which only identifies the functionalities required by the services to be composed, and component services that are able to provide the required functionalities are then associated to the individual task of the composite service. So, initial plan is a partial solution for the goal, and may not have a complete view of the global solution.

Assume that a initial plan has k tasks $p\{t_1, t_2, ..., t_k\}$, each task can achieved by one web service, there are many web service which have the same function. Thus, the optimize module select a best web service for each task. The optimize module adopts the following approach to obtain an optimal plan.

- Given a task t_i, if only one web service can achieve the t_i, then the optimize module select that web service for the t_i. For example, in Fig 5, the task t_3 only have one web service S_3. In this case, S_3 is used to execute t_3.
- Given a task t_i there are a set of web service that can be used to execute t_i. In this case, the optimize module needs to select one web service from the set.

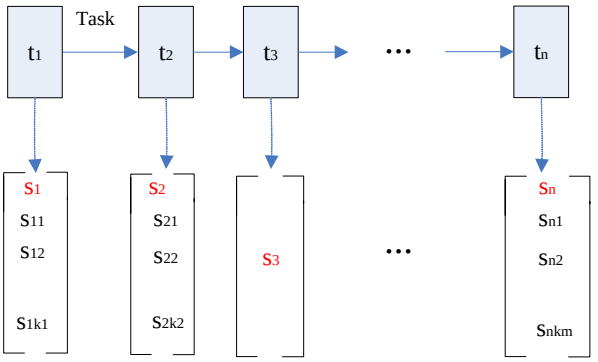

Fig 5: Service selection for the tasks

SWSCPA provide an optimization module to optimize the initial plan, which based on web services QoS selection algorithm.

QoS-based selection of services is very complex, not only due to the diversity of multifarious quality metrics with different value types, value range, and measurements, but also since an effective algorithm, which evaluates all metrics in combination, is missing.

We assume that the quality profile of *m* candidate services in set S for task t_i is denoted as $t_{iS} = \{S_{i1}, S_{i2}, ..., S_{im}\}$, where $S_{ij} = \{q_{ij1}, q_{ij2}, ..., q_{ijk}\}$. It defines that the advertisement of service S_{ij} has *k* quality metrics provided. It is quite obvious that it is rather unlikely that any S_{ij} will have the same number of quality metrics. So, we should have a preprocess to the quality metrics as below:

- To re-arrange the metrics of S_{ij} in the same order.
- If S_{ij} is lacking a quality, then one can add a metric and set its value to 0.

Therefore, the matrix of QoS for for task t_i $M_{ti} = \{S_{i1}, S_{i2}, ..., S_{im}\}$ looks like:

$$M_{ti} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1k} \\ q_{21} & q_{22} & \cdots & q_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ q_{m1} & q_{m2} & \cdots & q_{mk} \end{pmatrix}$$

Here, M_{ti} is a $m \times k$ matrix, with the quality information of candidates services in the each rows. Each column contains values of the same quality property. For uniformity, matrix M_{ti} has to be normalized with the objective to map all real values to a relatively small range, i.e., the elements of the final matrix are real numbers in the closed interval [0*;* 1]. The main idea of the algorithm is to scale the value ranges with the maximum and minimum values of each quality metric for thousands of current candidate services. Accordingly, the maximum and minimum values are mapped to the uniform values 1 and 0.

Some of the criteria used could be negative, i.e., the higher the value is, the lower the quality is. This includes criteria such as execution time and execution price. Other criteria are positive criteria, i.e. the higher the value is, the higher the quality is. and, there are a kind criteria that the user expects the value of this quality to be as close the given value as possible, we call this kind criteria as nearby criteria.

So, for negative criteria, values are scaled according to Equation 1:

$$v_{ij} = \begin{cases} 1 - \dfrac{q_{ij} - q_{\min}}{q_{\max} - q_{\min}} & if\,(q_{\max} \neq q_{\min}) \\ 1 & if\,(q_{\max} = q_{\min}) \end{cases} \quad (1)$$

For positive criteria, values are scaled according to Equation 2:

$$v_{ij} = \begin{cases} 1 - \dfrac{q_{\max} - q_{ij}}{q_{\max} - q_{\min}} & if\,(q_{\max} \neq q_{\min}) \\ 1 & if\,(q_{\max} = q_{\min}) \end{cases} \quad (2)$$

For nearby criteria, values are scaled according to Equation 3:

$$v_{ij} = \begin{cases} 1 - \dfrac{q_{\max} - q_{ij}}{q_{\max} - q_{\min}} & if\,(\theta \geq q_{\max}) \\ 1 - \dfrac{q_{ij} - q_{\min}}{q_{\max} - q_{\min}} & if\,(\theta \leq q_{\min}) \\ 1 - \dfrac{|q_{ij} - \theta|}{q_{\max} - q_{\min}} & if\,(\theta \in (q_{\min}, q_{\max}) \end{cases} \quad (3)$$

where $q_{\max} = \max_{i \in (1,m)} (q_{ij})$, $q_{\min} = \min_{i \in (1,m)} (q_{ij})$, $j \in (1,k)$. θ is a given value. By taking the Formula 3 as an example, it describes the case that the value of a quality as close as possible to θ is good.

After the scaling, we obtain a matrix $V = (v_{ij})$, the weighted value for each quality metric is defined in the web service ontology. The following formula is used to compute the overall quality score for each service:

$$Score\ (S_i) = \sum_{j=1}^{k} (v_{i,j} * w_j)$$

where $w_j \in [0,1]$ and $\sum_{j=1}^{k} w_j = 1$, w_j is the weight of the criteria. The optimize module will choose the web service which has the maximal value of Score(Si) for the task. If there are more than one web service which have the same maximal value of Score(Si), then a web service will be selected from them randomly.

In our service composition agent framework, there are 6 QoS criteria, they are performance, cost, reliability, availability, reputation and fidelity, it can be represented by using the extensible QoS model as follows: $QoS(S) = \{q_{per}, q_{cost}, q_{rel}, q_{avail}, q_{rep}, q_{fid}\}$, so the criteria q_{per}, q_{cost} are scaled according to Equation 1, and the criteria $q_{rel}, q_{avail}, q_{rep}, q_{fid}$ are scaled according to Equation 2.

VI. EXPERIMENT

SWSCPA ,which based on OWLS-Xplan and JADE[19], has been implemented in Java, and provides an integrated graphical user interface. OWLS-Xplan uses the Microsoft MSXML parser for PDDXML definitions and generating plans in XML format. In addition, OWLS-XPlan provides an integrated PDDXML editor that allows the experienced user to edit the goal, and the initial state ontology of given planning problem. JADE, a

Java based agent development environment, can be used to develop the agents and to establish communication between them. JADE also provide the environment for implementing the FIPA Contract Net Protocol [18] for negotiation between the agents involved in the system. Figure 6 shows the layout of SWSCPA in the form of blocks, with each block representing different tasks for various activities such as CreateMedicalTransportAccountService, FindNearestAirportService, BookMedicalFlightService involved in a Medical Planning request.

SWSCPA imports the definitions of simple, atomic web services expressed in OWL-S and translates them to planning operators. Inputs and preconditions of OWL-S web services are treated as relations to be queried in the precondition list, while outputs are treated as atoms to be added through the operator's add list. Effects are also atoms to be either added through the add-list or deleted, through the delete-list. Finally, SWSCPA generate a optimize plan based on services QoS selection method.

In the following case study, SWSCPA takes 30 available OWL-S services which belong to different communities, a domain description consisting of relevant OWL ontologies and a planning query as input, it returns a initial plan sequence of composed services that satisfies the query goal, the task name is the service name, the screen shot like Fig 6(a). There are several available services in each community. For example, Table 1 shows that there are four web services in *Flight Account Community*, this four service have the same IOPE parameters, but their QoS quality criteria values are different. In order to test selection mode, we assign a test values for this QoS quality criteria values as show in table 2.

TABLE 1:
THE SERVICES OF FLIGHTACCOUNT COMMUNITY

Community name	Web service	Service Parameter
FlightAccount	CreateMedicalFlightAccountService CreateMedicalFlightAccount2Service CreateMedicalFlightAccount3Service CreateMedicalFlightAccount4Service	**Input:** CreateMedicalFlightAccount2.owl#CreditcardInformation CreateMedicalFlightAccount2.owl#DesiredAccountData **Output:** NONE **Precond:** NONE **Effects:** MedicalFlightCompany2_Ontology.owl#ValidAccount

TABLE 2:
TEST DATA

Service Name	performance	cost	reliability	availability	reputation
CreateMedicalFlightAccountService	53	100	33	13	12
CreateMedicalFlightAccount2Service	3	30	43	43	12
CreateMedicalFlightAccount3Service	63	20	63	23	22
CreateMedicalFlightAccount4Service	103	120	133	113	2

From the definitions of each quality criterion, we know that *Cost* and *Performance* are expected to be smaller, *Reliability*, *Availability*, and *Reputation* are to be bigger.

The result of normalization carried out by our algorithm for the four candidate services referring to v_{ti} is:

$$V_{ti} = \begin{pmatrix} 0.5 & 0.2 & 0 & 0 & 0.5 \\ 1 & 0.9 & 0.1 & 0.3 & 0.5 \\ 0.4 & 1 & 0.3 & 0.1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad (4)$$

Assuming the weighted value for each quality metric as $w = \{0.2, 0.5, 0.1, 0.1, 0.1\}$, we apply Formula 4. to obtain a qualiity evaluation set, named $Score(t_i) = \{0.25, 0.73, 0.63, 0.6\}$. That is, in case of putting a high weight on price, service *"CreateMedicalFlightAccount2Service"* is the best choice for the task. As show in Fig. 6(b), after optimizing the initial plan, SWSCPA get a optimized plan for the composite service.

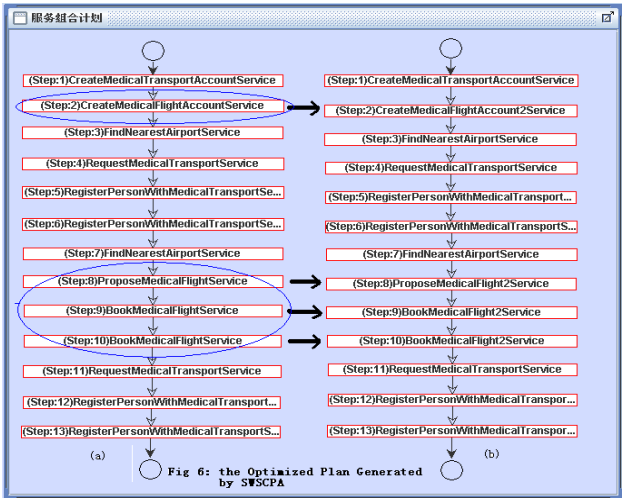

Fig 6: the Optimized Plan Generated by SWSCPA

REFERENCE

[1] D. Wu, B. Parsia, E. Sirin, J. Hendler, D. Nau, Automating DAML-S Web services composition using SHOP2, in: Proc. ISWC'03, 2003

[2] S. McIlraith, R. Fadel, Planning with complex actions, in: Proc. NMR'02, 2002.

[3] Huhns, M.N. et al, "Research Directions for Service-Oriented Multiagent Systems" *IEEE Internet Computing*, IEEE Computer Society 2005

[4] S. McIlraith, S. Son, Adapting Golog for composition of semantic Web services, in: Proc. KR'02, 2002.

[5] Liu, Y., Anne H.H. Ngu and Zeng, L., "QoS Computation and Policing in Dynamic Web Service Selection", *IEEE Computer Scociety*, ACM Press, New York, May 2004.

[6] Kalepu, S., Krishnaswamy, S. and Loke, S. W., "Verity: A QoS Metric for Selecting Web Services and Providers", *WISEW'03*, 2004, pp. 131-139.

[7] D. Mc Dermott, Estimated-regression planning for interactions with Web services, in: Proc. AIPS'02, 2002, pp. 204–211.

[8] Web Services Planning Agent in Dynamic Environments with Incomplete Information and Time Restrictions

[9] Hoffmann,J.The Metric-FF planning system: Translating Ignoring Delete Lists to Numeric State Variables. *Artificial Intelligence Research (JAIR), vol 20*. 2003.
[10] Maamar Z, et al. Toward an agent-based and context-oriented approach for Web services composition. IEEE Transactions on Knowledge and Data Engineering 2005:686–97.
[11] F. Casati, M. Sayal, and M.-C. Shan. Developing e-services for composing eservices. In Proceedings of 13th International Conference on Advanced Information Systems Engineering(CAiSE), Interlaken, Switzerland, June 2001. Springer Verlag.
[12] F. Casati, S. Ilnicki, and L. Jin. Adaptive and dynamic service composition in EFlow. In Proceedings of 12th International Conference on Advanced Information Systems Engineering(CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.
[13] Sycara K, Paolucci M, Soudry J, Srinivasan N. Dynamic discovery and coordination of agent-based semantic web services. IEEE Internet Computing 2004:66–73.
[14] Burstein M, Yaman F, Laddaga R, Bobrow R. POIROT: acquiring workflows by combining models learned from interpreted traces. In: Proceedings of the fifth international conference on knowledge capture. ACM; 2009. p. 129–36.
[15] Calvanese D, De Giacomo G, Lenzerini M, Rosati R. Actions and programs over description logic ontologies. In: Proceedings of the twentieth international workshop on description logics (DL-2007); 2007.
[16] Dikenelli O, Erdur RC, Gumus O. Seagent: a platform for developing semantic web based multi agent systems. In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems. ACM; 2005. p. 1272.
[17] Niu W, Li G, et al. Multi-granularity context model for dynamic Web service composition. Journal of Network Computer Applications 2011;34:312–26.
[18] Burstein M, Yaman F, Laddaga R, Bobrow R. POIROT: acquiring workflows by combining models learned from interpreted traces. In: Proceedings of the fifth international conference on knowledge capture. ACM; 2009. p. 129–36.
[19] Feier C, Roman D, Polleres A, Domingue J, Stollberg M, Fensel D. Towards intelligent web services: the web service modeling ontology (wsmo). In: International conference on intelligent computing (ICIC). 2005.