J. Serv. Sci. & Management, **2009, 2: 1-9**
Published Online March 2009 in SciRes (www.SciRP.org/journal/jssm)

A Virtualization-Based Service System Development Method

Tong Mo[1]*, Zhongjie Wang[1], Xiaofei Xu[1], Xianzhi Wang[1]

[1]School of Computer Science and Technology, Harbin Institute of Technology, 150001, China.
Email: motong_hit@126.com*; rainy@hit.edu.cn; xiaofei@hit.edu.cn; wangxianzhi@hit.edu.cn

Received December 30th, 2008; revised January 24th, 2009; accepted February 18th, 2009.

ABSTRACT

Amazon and Taobao publish product introduction online for customers' personal choice and become the successful examples of modern service based on information and communication technologies. However, if they want to achieve a complex service through composing some simple services, the research of service composition platform is still rudimental. In order to achieve this goal, a virtualization-based service system development method is proposed. The service elements are described standardized as service component at first. Then the service component is virtualized to be a web service and is deployed on a service platform, which is similar to Amazon. Customers put forward their demands on this platform and the platform will auto-match and dispatch task to the component to build a real-world service system. Finally, an application in ocean logistics service is briefly introduced.

Keywords: *service description, service system development, service composition, service component, online publish-choice, ocean logistics*

1. Introduction

The services industry has increasingly grown over the last 50 years to dominate economic activity in most advanced industrial economies. According to relating statistics, in some developed countries above 90%'s labor force are working for service industry and it occupies over 70% of the Gross Domestic Product (GDP) [1]. The era of service economy has arrived.

Assisted by development of information and communication technologies, service pattern has been transformed from provider-customer traditional mode into a complex social technical ecosystem [2]. More and more service links and participators increase the cost of service interoperation significantly. In order to reduce the cost, a popular way of these new service systems is using a virtualization-based service for composition. Providers display what they can provide through virtualized information on the web, such as pictures and some textual descriptions, to replace the physical display. And customers can select and get what they want (or inversed) [3]. The virtualization-based service can ignore the space-time distance and make information transformation, service composition and data collection more conveniently. On one hand, it can reduce service cost such as time, money, energy, etc; on the other hand, the original data that is saved automatically provide strong support for analysis and management. Some successful examples are Amazon [4], Taobao [5], etc.

Though there are such a lot of effective applications, most of them are "composition for physical services". What are published by provider or customer are physical things, such as books, commodities, etc. The services, for example, transportation, healthcare, consultation, etc., are also remained in state of single publish-choice. But with increasingly fining of social division, service has become more and more complex and cross-domain. If customers want to get a complex service that is composed by a lot of service elements that are provided by multi-providers, this kind of composition is always done by manpower. But on the internet, the number of providers is much larger than the number in local range. So we need a computer-assistant method that can develop a complex service system by service composition.

The complex service system in this paper is not only an information system, but also includes non-software elements such as people and hardware. But the main idea of the method is similar with that we use software services building business information system based on the idea of Software as a service (SaaS) in software engineering [6]. Travel service gives us a good reference in this domain. A travel program can be seen as a composition of the single service elements such as showplace, catering, accommodation, transport, etc, and they are still often composed by the travel agents or tourists themselves.

• The benefits of these new service system development methods are as follows:
• Fast service query on-line and selection.
• Computer-assistant composition based on customers'

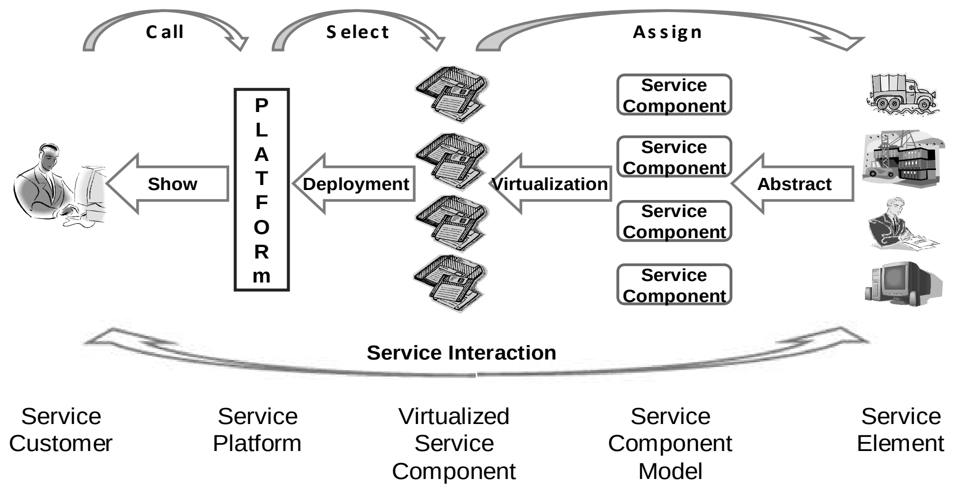


Figure 1. Big picture of building complex service based on virtualized service elements

voice and individual adaptation.

• Assign the service element in real world with the invoking and controlling service on web.

The rest of this paper is organized as follows. The big picture of the method is shown in Section 2. In Section 3, how to describe service elements is discussed and the virtualization method is proposed in Section 4. In Section 5, the arithmetic of service matching is briefly introduced and an application prospect is discussed based on a case of ocean logistic in Section 6. Finally the paper comes to the conclusions.

2. Big Picture: Virtualization-Based Service System Development

The big picture of building a complex service based on virtualized service elements is shown in Figure 1.

The process of this method includes three main phases and can be divided into eight steps.

Deployment phase:

Step1. The key elements of service such as behavior, actor, and resources are packaged as service components.

Step2. The service components are virtualized as software.

Step3. The virtualized service components are deployed onto the service platform.

Matching phase:

Step4. Customers log in the platform and input their individual demands through the interfaces that fits the habits of people. These demands are transformed into a standards XML format file.

Step5. According to the customer's demand on specific steps of service, platform will filter the relevant components, and get a smaller set of candidates. If some of the demands can find eligible components, the components closest to the demand will be reserved as a substitute.

Step6. According to the partial and overall demand of service, platform will match the candidate's components and feedback all the results in line with the demand. The result will be ranked with customer-defined policies to facilitate customer's choice.

Step7. Customer can self-edit the service based on the results, and submit a satisfactory outcome of the choice.

Scheduling phase:

Step8. Platform will assign the work to the software component that is selected by customer. And the *tasklist* of the component will add a record. Then the actor will execute service interaction following the *tasklist* with customer.

The key techniques and challenges in three phases are as follows:

• Standardized expression of the elements

In order to find what the customer want easily, the published things should comply with certain standards. The providers and customers often have different understanding and description with the same service element based on their own backgrounds and angles of view. This gap can be merged easily by similarity judgment and further communication likes a phone call or face to face talk in single-service-choice. But in complex service composition, cumulative effect of these gaps may make the result meaningless. Both of the above two successful composition platforms have a standard mode for each owner to show their goods and payment. But in this situation, a standardized expression for service is more complex than for physical goods. It not only contains the functional contents (input, output, precondition, quality index, etc), but also some other related elements. For example, the actor may be a very important factor that affects the customer satisfaction, and what they use is also need to be shown.

• Invoking service on web and assign the service element in real world

The service that published on web is not just a literal description of a variety of formats. It also can be called just as we operate software. A similar scene is that if we use telephone to appoint a service, there must be a call center (front) as a kind of interface. So for each service behavior, it also needs a software interface which is deployed on web for us to call. Though we publish and choose our service on web; it finally should be mapped and implemented in the real world. In Amazon, the book that a customer selects and chooses is just a virtualized image of real book on-line, but what the customer finally gets is that real book.

- Matching service based on customers' voice

Based on customers' voice, the most suited groups of services should be selected automatically. These results are collated by the integral sufficiency and key Quality of Service (QoS) parameters.

These three techniques and challenges will be discussed in the following sections.

3. Service Component: A Uniform Description of Service Elements

3.1 Information that Needs to be Described

Because there is still lacking a recognized definition for behavior that applies to all service fields, it is very hard to give a complete description of it. However, we describe it in order to establish a unified environment for customer and provider to choose and publish. So we only need to focus on the information that is used frequently in establishing the relationship between service demand and supply. A comparison with product behavior will help us to understand it more clearly.

The most important information is the function of behavior. It means what this behavior can do and what is the result of it. Though we can still use a binary group < input, output > as we describe the function of production behavior, the input and output of service behavior are mostly not physical things. For example, transformation behavior changes customers' position, so the input and output are the start points and end points respectively. In addition to input and output, quality index is another important part of service behavior function. Sometimes, for the same input and output, different performance of quality may means different services. In precedent, for the same transformation, the service may either be a normal one or an express one according to different time that should be spent on it.

Though we talk about service behavior and emphasize the difference from production, resource is also an important factor that impacts customers' choice. This resource is a kind of support for service behavior rather than the result. That means this resource is the thing that will be used during the service and impact customer's service experience.

In the field of manufacturing, customers just concern about the final product while don't care who made it. But because interaction is a major feature of service behavior, there could be some special request on the behavior actor. For example, a pretty miss front may reduce customers' complaints and an automatic response system could offer make them angry. Customers often put forward a number of requests to the actor to guarantee the behavior can be implemented smoothly.

3.2 Definition of Service Component

Based on the above analysis, we have gotten all the key elements that are needed for the establishment of the relationship between service demand and supply, and we use service component to depict it. Service component is a package of service actor and a behavior performed by this actor including all the supportive elements in conceptual level. It is the basic unit of service system offering a predefined functionality and able to communicate with other components. Reusability and independent execution are two important characteristics of service component and it can be plugged into different service systems. The structure is shown in Figure 2.

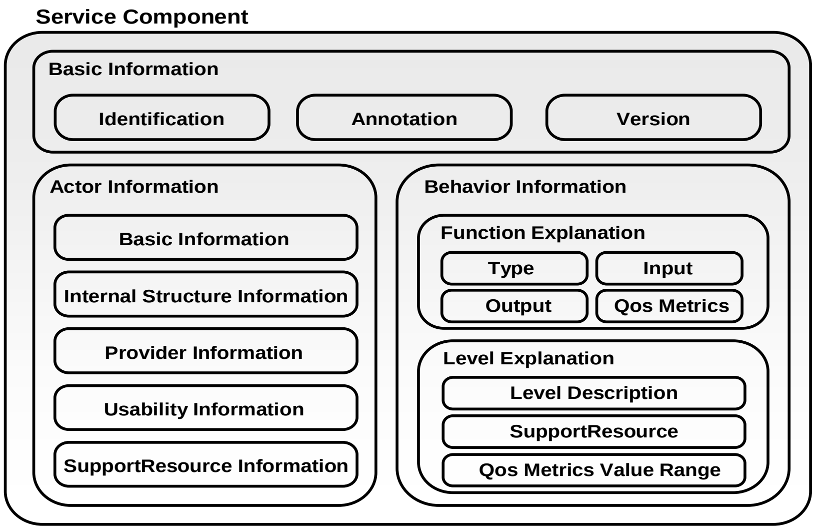


Figure 2. The structure of service component

As it is shown in Figure 2, service component gives us a unified format to describe the behavior that is needed in service by both customer and provider. It can be easily described by formal description language and here we use XML as a description language. The details are shown as follows:

• Basic Information: It involves the most basic information of a component, such as identification, annotation and version. Identification includes ID No. and name. Annotation is a text of narration. Version often includes Version No., creator and creating date.

• Actor Information: Actor is one of the two cores of service component. It describes the main body that provides service function in this component. In software domain, the main body of a traditional web service is a section of code and don't need to be further introduced. In service domain, the main may be human, software, hardware or a group of these things. It includes the following ingredients:

★ Basic Information. Such as ID No., name, etc.

★ Internal Structure Information. Sometimes, the actor of a behavior may be not executed by one entity but by a team or a group of entities. So the composition of actors is shown in this subsection.

★ Provider Information. Provider is the organization that the actor belongs to.

★ Usability Information. It shows the time when the actor can be used.

★ *SupportResource* Information. *SupportResource* is the thing that is needed and used by actor during the execution of service behavior. Sometimes, it is a little hard to distinguish a resource to be a *SupportResource* or a part of actor. Here we use an example to illustrate the nuance. In logistics service, a transport function is provided by a component. In this component, there is a truck driver and a truck that belongs to a cargo. In this cargo, if the truck driver just drives that truck, then they can be looked as a group, the truck is a part of actor. If the truck driver and truck don't have a binding relation and just many-to-many, the truck can be seen as a *SupportResource* for the truck driver.

• Behavior Information: Behavior is the other core of service component. As the same with traditional software behavior description; it also focus on the function description but adds one unique service character-service level.

Table 1. A case of service component package real service element

Natural language description of service element	*Service component description*
The name of truck driver is ZhangSan. He is a self-employed truck driver. His cell phone No. is 13936831568.	<tns:actor> <tns:provider> <tns:orgnizationName>ZhangSan</tns:orgnizationName> <tns:description>Self employed</tns:description> <tns:telephone>13936831568</tns:telephone> <tns:contactPerson>ZhangSan</tns:contactPerson> </tns:provider> </tns:actor>
He works from 9:00 AM to 7:00 PM everyday.	<tns:actor> <tns:availableTime> <tns:type>Everyday</tns:type> <tns:startTime>9:00</tns:startTime> <tns:endTime>19:00</tns:endTime> </tns:availableTime> </tns:actor>
His business is truck transformation	<tns:behavior> <tns:functionType>transormation_truck</tns:functionType> </tns:behavior>
The normal speed of delivery is an average of 80km/h and carrying capacity is 5t.	<tns:behavior> <tns:levels> <tns:levelInfo> <tns:levelName>normal</tns:levelName> <tns:levelDescription/> </tns:levelInfo> <tns:QoSMetrics> <tns:parameterName>speed</tns:parameterName> <tns:minimumValue/> <tns:maximumValue>80</tns:maximumValue> <tns:parameterUnit>km/h</tns:parameterUnit> </tns:QoSMetrics> <tns:QoSMetrics> <tns:parameterName>carrying_capacity</tns:parameterName> <tns:minimumValue/> <tns:maximumValue>5</tns:maximumValue> <tns:parameterUnit>t</tns:parameterUnit> </tns:QoSMetrics> </tns:levels> </tns:behavior>

★ Function Explanation.

◎ Type. It is a quote to a concept of service domain ontology. The subjectivity of service naming has brought a lot of inconvenience to service query. So the ontology will give us a standardized solution. Unfortunately, it is very hard to build industry recognized domain ontology, so we often use a conventional concept for substitution.

◎ Input & output (I/O). The same with software, the I/O is the main reflection of function. Sometimes, the I/O is information, but the type may not only be an electronic data, but also some papery stuff such as documents, graph, table, etc. In most cases of service behaviors, resource is another kind of I/O. For example, patients may get some drugs from a medical care; a repair service is another typical case.

◎ QoS Metrics. A set of quality parameters are used to evaluate the result of service behavior. Some of them can be obtained directly from the basis data. Some of them are gained via further computing, statistics or analysis. Some typical parameters are time, cost, customer satisfaction, etc. For each parameter, it needs to show the related data, data's acquiring method and the formula.

★ Level Explanation. In traditional web service, each function of a software component has only one performance. But in service domain, with the same terms of be havior and the actor, the service component may have different performance. This distinction is reflected by different *SupportResource* and the value of QoS parameters. For example, in logistics service, a transport component's actor is a person and the behavior is transport. For the same task, if the actor does it by a car, the time will be less than an hour and the cost will be 100 RMB; on the contrast, if the actor does it on foot, the time will be more but the cost will be less. So we use service level with different *SupportResource* and QoS metrics value range to make a distinction. It is a unique character of service component and what makes it most different from software component.

◎ Level Description. Including identifying information and a text of narration.

◎ SupportResource. The *SupportResource* will be used in this level. It is a subset of the *SupportResource* in actor information.

◎ QoS Metrics Value Range. This is the value bound of quality parameters.

A case of service component is shown in table 1, and it shows a service component of transformation. The natural language description of service is in the left and the right is the XML based service component description follows the above structure. For example, the normal speed of delivery is an average of 80km/h and weight is 5t. So it has a service level named normal and has two QoS parameters: speed and weight. Of course, it may have other levels, such as a high-speed which may load less. As a complete service component description is too long to show, we just excerpt some critical segments to this paper.

4. Virtualization Method of Service Component

Service component is only a standard document for provider to publish his service and for customer to query easily. In contrast, the software virtualized version provides a user interface to show how to use the real service element packaged by component and record the running information. It is similar as the relationship between web service and the functional software system. According to the difference of actor and business level, the service component can be divided into human component, software component, hardware component and composite component.

4.1 Virtualization Method for Human Component

Human component represents the behavior that is executed by people or mainly depends on people. It is the dominant component and the difficulty of service description. But software human behavior is not a new topic and there has been already a well-formed specification: WS-HumanTask [7]. It is used with another specification: BPEL4People [8] which will also be referred in sector 4.4 to define the process of human-machine interaction. Though WS-HumanTask focuses on software domain, it provides us with fairly fine basis to draw on.

In our opinion, the software of human component is just like a software client of the real people. Component user can use that as software to assign work to service executants just like what we do by cell phone, email or even verbal order. This software client includes three main parts: *tasklist*, log and interface.

In the view of the idea of WS-humanTask, a job assigned to the actor of a component is treated as a task and the *tasklist* is a task schedule for the component. It records when, where, to whom that the component need to provide service. It is a kind of continuously information.

Log is the utilizing history of the component. It includes serial number, date and time, source, type, event and user.

The two parts above are used frequently to record the call of components. All the functions of software service component are invoked through the interface. The outside world can call it just like calling general software interface. According to different uses, the interfaces are divided into common interface and service function interface.

Common interface corresponds to the general functions of all software service components, for example, query component's information, amend component's state, manage task list and log, etc. This kind of interface doesn't relate to specific service and is just used to query and manage its own information.

Service function interface represents a typical service function of component, such as transporting, packing, checking container, etc. This kind of function is reflected in the form of service task and service interface is the interface of a task.

The steps of making an XML file of service component to be software are as follows:

Step1. Building new software which is named same with the service component and adding an empty *tasklist* and log file.

Step2. Initializing the software and generating the common function of it: *tasklist* management and log management.

Step3. Adding the query function of software and linking with the XML file of service component.

Step4. Building the service function call interface and the parameters is the input of service behavior.

Software interfaces of a component are shown in Table 2. The left is the XML based service component description, and the right is the corresponding interfaces. For example, this component has an actor, and the provider of actor is named ZhangSan, so we can get this information by call the interface hasProvider (actor).

4.2 Virtualization Method for Software Component and Hardware Component

SaaS is a model of software deployment where an application is hosted as a service provided to customers across the Internet [9]. So customer doesn't have to take care about software maintenance, ongoing operation, and can on-demand pricing. It is gaining a great deal of attractions today and more and more businesses are adopting SaaS for cost-effective software management solutions as well as business structure and process transformations [10]. These well-packaged software services can be called directly and don't need any form of repackaging [11].

Of course, there are also many legacy systems that is developed using the traditional way. If we want to use the function of these old systems, we need to package some function interface as a web service. The research of web service packaging technology is fairly mature so we won't go details in this paper.

In fact, there is no pure hardware component, because a behavior can hardly be implemented by hardware alone. This kind of component usually packages two types of behavior.

One can be carried out by big machine which is independent and has a high degree of automation, such as Computerized Numerical Control (CNC) machine. But it still needs a command from the outside world. For this type of component, we need to issue a task order to the operator, and the call mode as well as virtualized method is the same as the human component in Sebsection 4.1. The only difference is that the hardware takes place to be the main body of the actor.

The other is the behavior of the automation equipment which is controlled by its own software system. Packaging this kind of component involves developing a special interface for the system. A typical application is the Global Position System (GPS).

Table 2. The result of a component that is software virtualized

Service component		*Function of software virtualized comonent*	

Schema of component	<tns:actor>
	<tns:provider>
	<tns:orgnizationName>ZhangSan</tns:orgnizationName>
	<tns:description>Self employed</tns:description>
	<tns:telephone>13936831568</tns:telephone>
	<tns:contactPerson>ZhangSan</tns:contactPerson>
	</tns:provider>
	<tns:availableTime>
	<tns:type>Everyday</tns:type>
	<tns:startTime>9:00</tns:startTime>
	<tns:endTime>19:00</tns:endTime>
	</tns:availableTime>
	</tns:actor>
	…
		Query function	hasActor(component)
			hasProvider(actor)
			hasAvailableTime(actor)
			…
			The return values of these functions are a string of the results.

Behavior of component	<tns:behavior>
	<tns:functionType>transormation_truck</tns:functionType>
	<tns:input>
	<tns:inputInformation>
	<tns:billObject>
	<tns:name>PaicheBill</tns:name>
	<tns:parameters>
	<tns:address/>
	<tns:arrivetime_plan/>
	</tns:parameters>
	</tns:billObject>
	</tns:inputInformation>
	</tns:input>
	</tns:behavior>
		Service function	callService(component, level, inputstring) // The value of inputstring is a string that is composed by the input of the behavior and separator. In this case, the inputstring is “Harbin Westgreat Street 207# @@@ 2008-11-21 11:00”. The reurn value of this function is a taskid.
			getResult(component, taskid) // Get the result of the service task. The return value is a result object.

		Task and log function	taskQuery(condition)
			logQuery(condition)
			…

4.3 Virtualization Method for Composite Component

The three types of service components mentioned above are basic components executed by single actor. But in the actual process of service, many acts of service process are scheduled following the relatively fixed mode, and some of the components are ordered with the regular emergence of high-frequency. In order to simplify the selection of components and service matching, reduce the computational complexity, and improve the efficiency of the service schedule, we combine these components a larger one for further reusability. It is similar with the establishment of the large size software component and web service in the software domain.

Because the three basic components are all packaged as software, we can directly draw on the existing standard used to build a large size web service to build composite service component. The common industry practice is using Web Services Business Process Execution Language (WS-BPEL) [12] and WS-BPEL Extension for People (BPEL4People) [8] to specify the information interactions between software service, and orchestrate them as a process. At the beginning of a BPEL process, <partnerLink> and <variable> is used to define the link of service and declare variables. The process is combined by <sequence> which is a group of activities that is called one by one, and <flow> which is a group of parallel activities. In the main body of process, <invoke> is used to call a service and <receive> is used to wait the service to callback. The control of the implementation logic includes <switch>, <while> and <pick>. The other details of BPEL and BPEL4People can be found in documents 8 and 12.

5. Computer-Aided Service Selection and Matching

Service component and virtualization is a kind of computer understandable way for service elements describing, saving and calling. So the computer can pre-select and match the service and provide us options, when customers make a complex service needs. And it can release us from the heavy work of comparison, calculation and communication, when there are more and more service options.

The needs of customers' are a lot of constraints on total or local quality indicators, such as function, time, cost, credit, etc. Service selection and matching is choosing appropriate service components and organizing them correctly to satisfy the customers. It equals to a multi-objective optimization problem and we use a genetic algorithm (GA) based service matching algorithm (SMA) to achieve it.

• Input

The input of SMA includes four parts: QoS parameter trees, service flow model, constraints sets, and service components sets.

QoS parameter trees are a set of key performance indicators of services and are used to evaluate the quality. The name and calculation method of these parameters are relatively stable and are divided by industry.

Service flow model is a XML file that is parsed from customers' voice. On one hand, it shows which simple parts the complex service includes; on the other hand, it describes the sequential logic of these simple parts and is used as the basis for some types of parameters such as time.

Constraints sets are the other part of customers' voice. They are divided into three types: local constraint is the constraint for one simple part; partial constraint is for two or more simple parts and global is for all the simple parts. All of the constraints are described according two formats: The first one is simple condition which is a triple form <qp, mo, value>; qp is quality parameter, mo is mathematical operator and value is the number and unit. For example, "time < 5 hour" is a simple condition. The other one is complex condition which is a logical expression of simple condition. For example, "if time < 5 hour then money = 100 RMB else money = 50 RMB". All the constraints have a weight to show the importance for customers.

Service components sets are the sets of service components classified according to service domain.

• Output

The output of SMA includes four parts: service components set, result of evaluation, and scheduling plan. Service components set are the components selected for each simple parts of the complex service. Result of evaluation is the sufficiency of the service components set to the customers' voice. Scheduling plan is a set of information for calling the service components, including time, place, and other useful information.

• Implementation logic

The implementation logic of SMA is the same with GA, and the principle of algorithm is shown as follows:

```
1. Initial population (first generation)
2. Determine the fitness of each chromosome/ individual
3. Repeat:
Perform selection
Perform crossover
Perform mutation
Determine the fitness of each individual
Until the stopping criterion applies
4. Return last population
```

The key operations of SMA are follows:

★ Chromosome coding

A chromosome can be seen as a result of service selection and matching. It is a set of gene with a certain order and each gene represent a service component. For example, in a travel services, a chromosome is a complete tourist routes and genes are attractions, transport, accommodation, catering, etc. A population is a set of the chromosomes that in the domain of this complex service.

★ Genetic operation

Generating: The population is generated following a random way and the population size is fifty. Each gene in one chromosome is generated by the service component that has the same function type. Here we use a pre-selection technique to limit the scope of the service components. We filter out those components which do not meet the local constraints before the SMA. So the service composition is under the condition of the components that meet the local constraints and if some of the gene can't find eligible components, the components that are closest to the demand are reserved as a substitute.

Selection/Reproduction: Here we use a mechanism for the survival of the fittest and the worst five chromosomes fall into disuse. Another mechanism in SMA selection is differential reproduction. The chromosomes are put into the breeding pond according to a [0,2] probability which is calculated by its fitness.

Crossover: New chromosomes are generated by exchanging the genes in the same location of two randomly selected chromosomes. Considering the characteristics of service composition, we crossover the chromosomes under niche protection. Niche presents a common feature of the same species in biology. In service composition, some simple parts of a complex service are also required to have some same features. For example, some parts should be provided by the same provider. So the niche protection is used to avoid the broken of these same features through the crossover.

Mutation: Mutation changes some genes of a chromosome and brings some new genes to the population. In SMA, the chromosome and the position of genes are selected randomly.

★ Fitness determination

In SMA, we use customer satisfaction to be the fitness of a complex service and use penalty function method to determine the satisfaction.

$$f(c) = P(c) - N(c)$$

The *f(c)* is fitness of a chromosome and it equals to the positive satisfaction *P(c)* minus the negative impact *N(c)* of those constraints that are not bound to meet.

The satisfaction of the chromosome *P(c)* is expressed as

$$P(c)=\sum_{i=1}^{n} w_i \prod_{j=1}^{m} S_j(i_j)^{wj} \sum \lambda_{as}$$

$$\left(\sum_{i=1}^{n} w_i = 1, \sum_{j=1}^{m} w_j = 1\right)$$

There are n genes in a chromosome and w_i is the weight of gene i. For each gene, the number of QoS parameters is m, and w_i is the weight of parameter *j*. $S_j(i_j)$ is the satisfaction function of the QoS parameters *j* of gene *i*. λ_{as} is the additional contributions to the satisfaction of those genes that are intrinsic satisfaction associated.

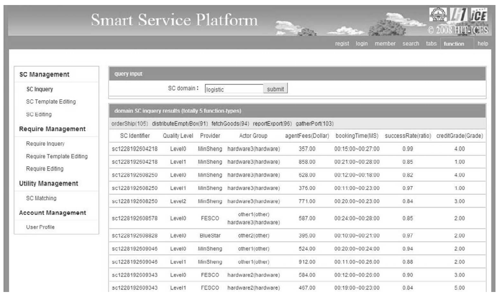
6. Case Study

The full container load (FCL) export business is one of the core businesses in ocean logistics service domain. It is a deep-division-business and the business chain can be broken down into a number of coupling parts of lower level. Because of the high degree of specialization, one participator is usually responsible for one single-part in the business chain. So in a FCL export business, it often involves many organizations and in most cases the number is greater than five. This situation leads extremely large cost for establishing business relations such as time, money and energy. At this stage in the industry, this problem is mainly solved through a long-term cooperative relationship. However, it makes the service to be a low-optional one and very likely to run another way counter to the trend of "On-demand". The contradiction is a typical problem exists in many service fields that have similar characteristics. So we choose it as case background and try to verify the virtualized service ecosystem's convenience and improvement on efficiency of system building.

A typical FCL export business mainly includes five parts: booking cabins, container yard choice and container allocation, loading goods, export declarations, and container gate-in. According to business needs, these parts can be further divided into different levels. For example, the part of loading goods includes container preparing, truck distributing, goods transformation etc. All of these can be seen as different sized behaviors and can have the providers. Firstly, they are packaged into service components according to the rules above. The main components for the five business links are shown in Figure 3. Figure 4 shows a matching result according to a customer's requirement. As soon as the customer confirms a solution from the result set, the platform will send the task message to the real actor through the virtualized software, and start the service

7. Conclusions

Nowadays, though service develops rapidly, the system for behavior service is still rudimentary and lacks coordination. The results of this paper try to make up for this. Although the virtualized system is still in the simulating and testing stage, it opens a brilliant prospect that we can orchestrate behavior service just like what we have done in web service domain.

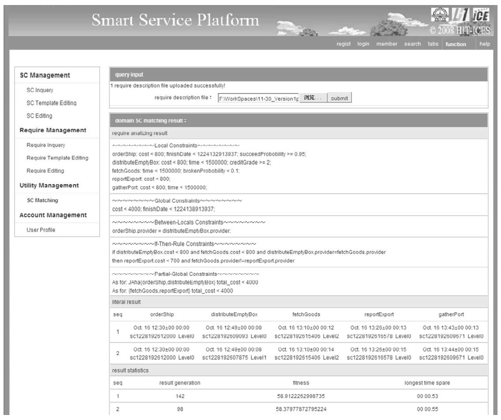


domain SC inquery results (totally 5 function-types)

orderShip(105) distributeEmptyBox(91) fetchGoods(94) reportExport(96) gatherPort(103)

SC Identifier	Quality Level	Provider	Actor Group	agentFees(Dollar)	bookingTime(MS)	successRate(ratio)	creditGrade(Grade)
sc1228192604218	Level0	MinSheng	hardware3(hardware)	357.00	00:15:00~00:27:00	0.99	4.00
sc1228192604218	Level1	MinSheng	hardware3(hardware)	858.00	00:21:00~00:28:00	0.85	1.00
sc1228192608250	Level0	MinSheng	hardware3(hardware)	628.00	00:12:00~00:18:00	0.82	4.00
sc1228192608250	Level1	MinSheng	hardware3(hardware)	376.00	00:11:00~00:23:00	0.97	1.00
sc1228192608250	Level2	MinSheng	hardware3(hardware)	771.00	00:20:00~00:23:00	0.84	3.00
sc1228192608578	Level0	FESCO	other1(other) hardware3(hardware)	587.00	00:24:00~00:28:00	0.85	2.00
sc1228192608828	Level0	BlueStar	other2(other)	395.00	00:10:00~00:21:00	0.97	2.00
sc1228192609046	Level0	MinSheng	other1(other)	524.00	00:20:00~00:24:00	0.94	2.00
sc1228192609046	Level1	MinSheng	other1(other)	912.00	00:11:00~00:26:00	0.88	2.00
sc1228192609343	Level0	FESCO	hardware2(hardware)	584.00	00:12:00~00:26:00	0.90	3.00
sc1228192609343	Level1	FESCO	hardware2(hardware)	467.00	00:19:00~00:23:00	0.84	5.00

Figure 3. Service components deployed on the platform



query input

1 require description file uploaded successfully!

require description file : F:\WorkSpaces\11-30_Version1p

domain SC matching result :

require analizing result

```
~~~~~~~~~~Local Constraints~~~~~~~~~~
orderShip: cost < 800; finishDate < 1224132913937; succeedProbobility >= 0.95;
distributeEmptyBox: cost < 800; time < 1500000; creditGrade >= 2;
fetchGoods: time < 1500000; brokenProbobility < 0.1;
reportExport: cost < 800;
gatherPort: cost < 800; time < 1500000;

~~~~~~~~~~Global Constraints~~~~~~~~~~
cost < 4000; finishDate < 1224138913937;

~~~~~~~~~~Between-Locals Constraints~~~~~~~~~~
orderShip.provider = distributeEmptyBox.provider;

~~~~~~~~~~If-Then-Rule Constraints~~~~~~~~~~
if distributeEmptyBox.cost < 800 and fetchGoods.cost < 800 and distributeEmptyBox.provider=fetchGoods.provider
then reportExport.cost < 700 and fetchGoods.provider!=reportExport.provider

~~~~~~~~~~Partial-Global Constraints~~~~~~~~~~
As for: JAha{orderShip,distributeEmptyBox} total_cost < 4000
As for: {fetchGoods,reportExport} total_cost < 4000
```

literal result

seq	orderShip	distributeEmptyBox	fetchGoods	reportExport	gatherPort
1	Oct. 16 12:30±00 00:00 sc1228192612000 Level0	Oct. 16 12:49±00 00:08 sc1228192609093 Level0	Oct. 16 13:10±00 00:12 sc1228192615406 Level2	Oct. 16 13:26±00 00:13 sc1228192616578 Level0	Oct. 16 13:43±00 00:13 sc1228192609671 Level0
2	Oct. 16 12:30±00 00:00 sc1228192612000 Level0	Oct. 16 12:49±00 00:08 sc1228192607875 Level1	Oct. 16 13:10±00 00:14 sc1228192615406 Level2	Oct. 16 13:26±00 00:15 sc1228192616578 Level0	Oct. 16 13:44±00 00:15 sc1228192609671 Level0

result statistics

seq	result generation	fitness	longest time spare
1	142	58.9122262998735	00 00:53
2	98	58.37977872795224	00 00:55

Figure 4. Matching result

From the case test we can find that, though the initialization of platform and add service component may cost a lot, for each using, it just needs less cost to edit customer's demand. A prominent advantage of virtualized system building is that the cost (time, money, and energy) is less impacted by the number of provider. In contrast to this, the cost in traditional system increases dramatically when the provider number grows. Though they use stabile service chain to improve the running condition, the custom degree is also limited. Sometimes, getting a full accordance with the wishes of customer solution from a large number of candidates by manpower can be an unpractical mission. Because the service matching is auto-calculated in virtualized system building, so it can release us from the complex computation and compare work, and we become able to pay attention to work more meaningful. At the micro level, we can use statistical methods and data mining to find which component and which combination is often used and then further explore the reasons and the law. At the macro level, based on statistics for a period of time, we may find the evolution of each part in a service business chain from an ecosystem angle.

Future work includes: establishing a sound interface system, applying this system into real service, and refining the component matching algorithm.

8. Acknowledgement

Research works in this paper are partial supported by the National High-Tech Research and Development Plan of China (2006AA01Z167, 2006AA04Z165) and the National Natural Science Foundation (NSF) of China (60673025).

REFERENCES

[1] J. Spohrer and P. Maglio, "Emergence of service science: Services sciences, management, engineering (SSME) as the next frontier in innovation," Nordic Service Innovation Workshop, Oslo, Norway, 2005.

[2] P. Maglio, S. Srinivasan, J. T. Kreulen, and J. Spohrer, "Service systems, service scientists, SSME, and innovation," Communications of the ACM, New York, Vol. 49(7): pp. 81–85, 2006.

[3] J. Spohrer and P. Maglio, "Emergence of service science: Services sciences, management, engineering (SSME) as the next frontier in innovation," Nordic Service Innovation Workshop, Oslo, Norway, 2005.

[4] http://www.Amazon.cn.

[5] http://www.Taobao.com.

[6] T. Mo, J. M. Xu, Z. J. Wang, Y. F. Ma, H. Y. Huang, Y.Wang, A. Liu, J. Zhu, and X. F. Xu, "SCS: A case study on service composition in automobile supply chain," Journal of Harbin Institute of Technology, Harbin, China 2008 Supplement 1. pp. 323–329, 2008.

[7] http://www.oasis-open.org/committees/download.php/27549/ws-humantask-1.1-spec-wd-02.doc.

[8] http://xml.coverpages.org/bpel4people.html.

[9] http://en.wikipedia.org/wiki/SaaS.

[10] SaaS 2.0: Software-as-a-Service as Next-Gen Business Platform, Saugatuck Technology.

[11] SaaS Integration Platforms: The Looming SaaS Deployment and Support Dilemma, Saugatuck Technology.

[12] http://www.oasis-open.org/committees/wsbpel.

(Edited by Vivian and Ann)