International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 3 Issue: 1
ISSN: 2454-4248
09 – 13

Development of Unified Framework for Discovery and Negotiation Requirement for new services in Service Oriented Architecture

Mohd Ashraf
Associate Professor, Computer Science & Engineering
Maulana Azad National Urdu University, Hyderabad
E Mail: ashraf.saifee@gmail.com

Abstract: Service oriented architecture is a style of software design where services are provided to the other component through a communication protocol over a network. It is an emerging approach that addresses the requirement of loosely coupled, standards based and protocol independent distributing computing**.** To Build an SOA a highly distributable communication and integration backbone is required. In this paper, Authors presents the unified framework for discovery and negotiation requirement for new services in Service Oriented Architecture. Specially, this paper also address the issue to negotiate and search the new services, differentiating between several old services and the new services that are similar but not identical based on specification.

Keyword: *Software Engineering, Service Oriented Architecture, Software Requirement, Software Reuse specification Requirement.*

*****

I. INTRODUCTION

Software engineering is a systematic approach of development, operation and maintenance of software. In software engineering, Service-Oriented Architecture, different method which is used for software development with the help of services and these services may communicate with other services also. Service-Oriented Architectures (SOA) is becoming increasingly widespread in a variety of computing domains such as enterprise and e-commerce systems, which continue to grow in size and complexity. These systems are expected to adapt not only to the fluctuating execution environments, but also to changes in their operational requirements. SOA is a collection of different services and these services can communicate to each other using message passing (message passing include simple data passing or coordination of different activates). Software architecture describes the system's components and their interaction at the high level. These components are not distributed objects and work as a module which is deployed onto a server as a single unit along with other components and the interaction between the components is called "connecters".

Using Service-oriented Architecture, Software quality can be improved as well as cost will reduce with more reusable component in software engineering. Reusable components are designed to perform specific functions. These are independent and pre-built pieces of programming code. Therefore, it is important and productive to conduct research on how to develop software with service –oriented computing technology.

There is not enough research and practices to implement "register, find, bind and execute" paradigm and make practical and cost-effective. So we need to analyze this process deeply to provide practical architectures and methodologies for reusable services. The impact of reusability in SOA is innovative. The component development for providing various services is a difficult task for a Service Oriented System. It is not efficient to develop a new component for a new service every time as it would not be economical and also it is difficult to integrate it with the Legacy System.

The structure of the paper is as follows: Section 2 describes related work. Section 3 describes different architectural practices for integrating reusable services and architecture. Section 4, describes unified model for discovery and negotiation for new services in SOA. Finally, Section 5, summarizes the main conclusions from this paper.

II. RELATED WORK

Service-Oriented Architecture (SOA) is a software architecture style that has recently gained in popularity because of its potential to maximize reuse, operability, and flexibility. SOA achieves the aforementioned benefits by dividing the software architecture into three main components: **(a)** service providers **(b)** service consumers, and **(c)** service brokers.

To achieve a SOA, the players must interact in a specification, as shown in *Figure 1*.

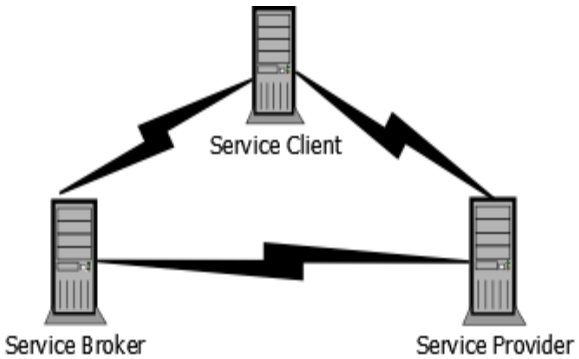


Figure1: Service-Oriented Architecture

Tai, et al. [1], address the problem of transactional coordination in service-oriented computing. The authors introduce the concept of system support for transaction coupling modes as the policy-based contracts guiding transactional business process execution. An SOA requires

 ISSN: 2454-4248
09 – 13

that developers discover at development time service descriptions in repository systems and, by reading these descriptions they are able to code client applications that can (at run time) bind to and interact with services of a specific type i.e., compliant to a certain interface and protocol). To address this problem, *Deora, et al[2].*, propose a quality of service management framework based on user expectations. This framework collects expectations as well as ratings from the users of a service and then the quality of the service is calculated only at the time a request for the service is made and only by using the ratings that have similar expectations. Similar research efforts are reported in *Patil, A.A. and Meteor-S[3]*. The AI and semantic Web community has concentrated their efforts in giving richer semantic descriptions of Web services that describe the properties and capabilities of Web services in a computer-interpretable form. For this purpose,
DAML-S ,*Rouvellou, et al.[4]* language has been proposed to facilitate the automation of Web service tasks including better means of Web service discovery, execution, "automatic" composition, verification, and execution monitoring. In addition, in *Jaeger, et al. [5]*, an approach is described that builds on top of existing Web services technologies and combines them with some concepts borrowed from the semantic web to leverage web service discovery and composition. This approach is captured by the METEOR-S Web Service Annotation Framework (MWSAF). In the basic SOA provides a simple browsing-by-business-category mechanism for developers to review and select published services. In [15], a hybrid matching approach is suggested, combining semantic and syntactic comparison algorithms of WSDL documents. Comparable research efforts have been reported in [6].

Reusability Problem of a component in service and different Architectures for Services

It has been discussed enough that why it is difficult to develop a new service for every time but along with the development of a new service it is similar difficult to reuse an existing component for new services also. A traditional software component cannot be designed and applied in the same way as the reusable component. The main difficulty for the reusable component is to meet the all specifications with the implementation logic. We required two kinds of effort, for making services reusable, one is that we provide the services whose development is difficult and needed an advanced development environment which is difficult or expensive for application developers to build or purchase. The other one is that we need to make the service easily applicable and required cheep service application tools that can be installed easily.
Krueger proposed a framework in 1992 to evaluate software reusability with the following four aspects [7] abstraction: a reusable component should have a specific level of abstraction that avoids the software developers to sift the details of it; selection: a reusable component should be easy to locate, compare, and select; specialization: a group of reusable components should be generalized and the generalized form should be easy to be specialized to an application; and integration: There must be an integration framework to help reusable components be installed and integrated into newly developed application, *Sarukkai, S.[8]*. From these four criteria, abstraction and specialization for service are definite.
A service as a good abstraction should be affordable, attractive, necessary, professional and trustworthy. Affordable means purchasing a service should be cheaper than developing by developers themselves. Attractive means a service should provide exciting performance. Necessary means a service should be necessary for developing specific applications; Professional means the providers should have strong experiences and background for the services and trustworthy means it is safe to use the services provided. If we want to achieve these goals it is important to select and integrate the service and a good architecture help for achieving these goals [9].

• ***Architecture for Centralized service center***

This architecture makes the Selection easy because the number of service centers is limited and definite. For Integration, a service is applied into and application is restricted by service center. In this architecture application providers are not so much flexible. This architecture is not appropriate because some time large quantities of cheap workstations and desktop computers are available. This is a practical solution for many complicated services such as Global Positioning System (GPS) services, Coordinated Universal Time (UTC) services, weather forecast services and bioinformatics services.
This architecture is used in National Aeronautics and Space Administration (NASA) and the Europe Space Agency. This center might need to install powerful super computers that are normally not afforded to application developers. It also needs to provide large databases that normally application developers are unable to be managed. The volume of database should be greatly superior to any database managed by an application developer. It is also installed for specific expensive systems for specific services.

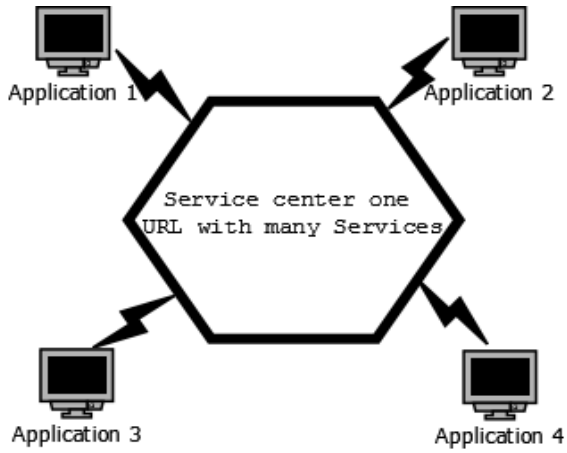


Figure 2: Architecture for Centralized service center

Architecture for Distributed Service Center

This architecture works same as centralize service center for both process selection and integration. The only difference is the construction of the center but not the service provision. It provides large quantities of services and equipped with high speed networks. The bandwidth should

 ISSN: 2454-4248
09 – 13

support the rush service hours without significant delay for satisfactory services. This architecture is appropriate to large number of service requests with small number of data to be processed. This architecture can be found in VSM [2].

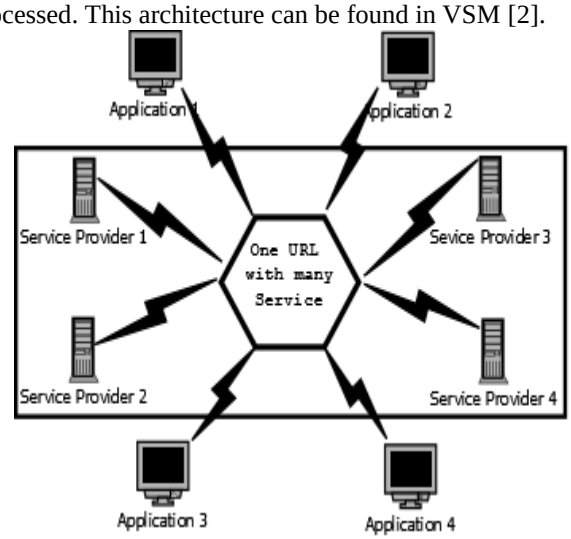


Figure 3: Architecture for distributed Service Center

• Architecture for Distributed service providers with Centralized service center

In this architecture Selection is easy as application developer only contact to service registry. This service registry only stores the service specification not the real services. So this architecture makes the Integration very flexible. This architecture helps for making a service registry and many small service providers. And each service provider only provides small number of services.

In this architecture only a single computer is responsible for accepting and replying the request and only service provider will accept and reply for the service request.

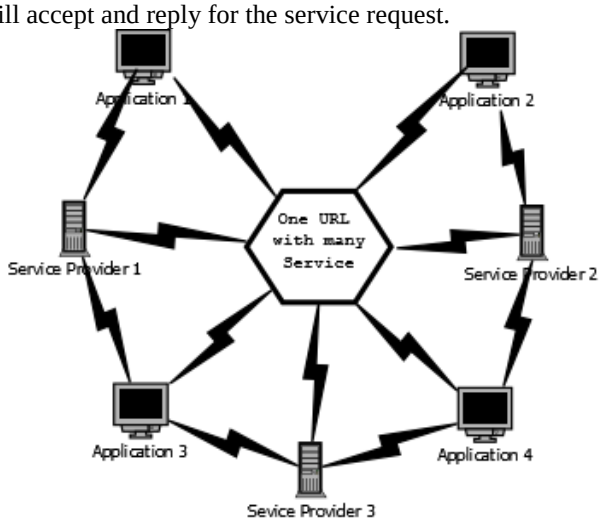


Figure 4: Distributed service providers with centralized Service center

Architecture for Distributed service providers with Distributed service registry

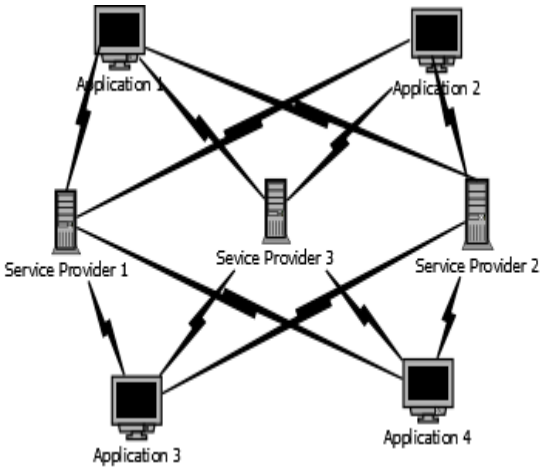


Figure 5: Distributed Service providers with Distributed Service Registry

In this architecture the Selection and Integration is more flexible, but this architecture is also very complex. It requires special consideration of service registry, service negotiation and service provisions. Much of the literature in service computing is trying to contribute within these types of architectures *(Sarukkai, S.,2005).* With this architecture, service providers are located at different computers with different URLs. We call them independent service providers. The service registries are also distributed on different computers.

All of the service providers form a service network. The application developers will broadcast or multicast their requests to all the networked computers, collect the replies, compare and select one service provider, sign a contract and send out the service request. Handorean and Roman [8], proposed a similar architecture. With this architecture, both the service providers and the application developers may specify the services. It needs a lot of negotiation for the two parties of the service to make an agreement [6].

Proposed Model for Discovery and Negotiation of the New Services

This paper proposes a combined model for service discovery and negotiation of new service. Author proposed the discovery method and new service negotiation method. In this process, Service requesters can be either arbitrary application developers or other service providers. A service provider needs to register its services with a service registry and provide services directly to interested parties. Each service may have multiple service interfaces to meet the needs of different requesters, and requesters can dynamically discover the interfaces they require. Making discovery-based service abstraction is challenging.

Following are the steps involved in the proposed model for discovery and negotiation of new Service in Service Oriented Architecture

Step 1: Firstly, all service and negotiation threads are discovered, and specifications are searched.

Step 2: If specifications are exactly matched

a) Then available service is searched and provides the service.

 ISSN: 2454-4248
09 – 13

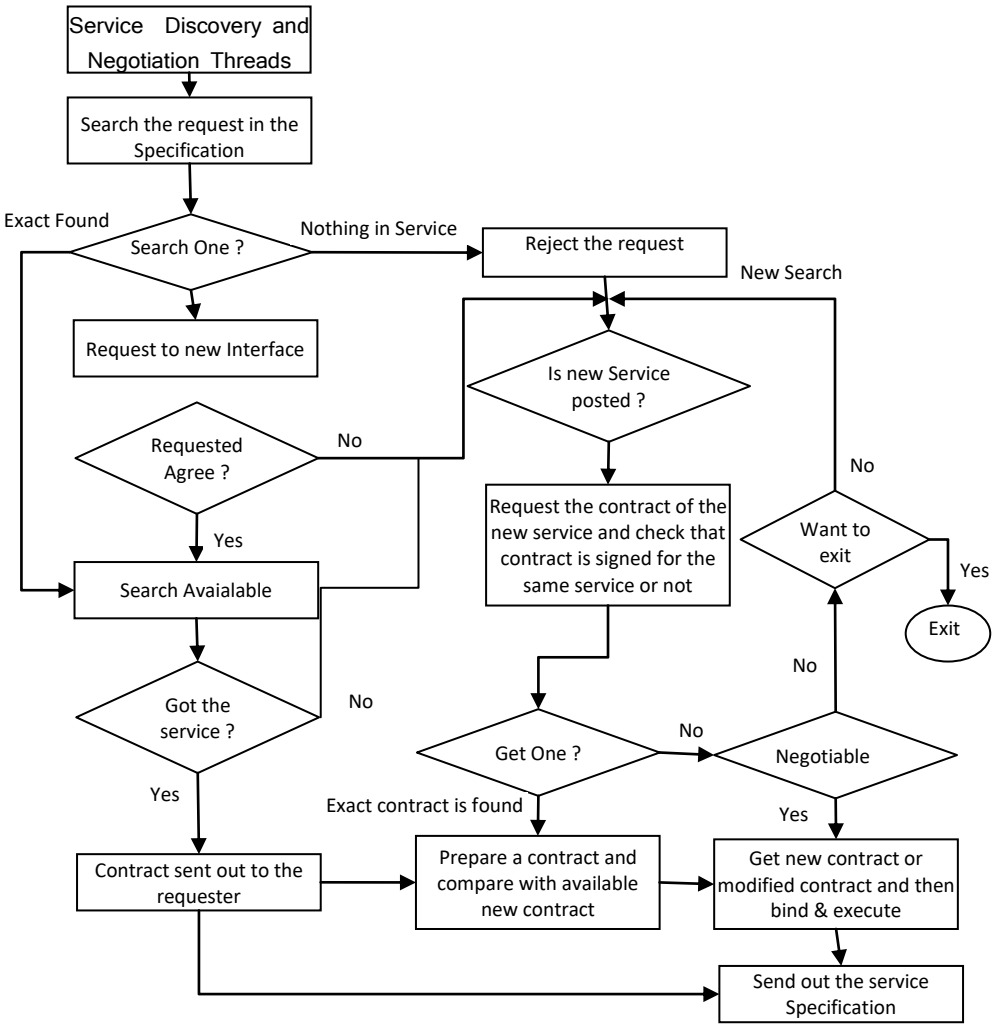
b) If specification is not matched then request gets rejected and will search for new service. Now it will check the interface with matching parts and send request for the same.

Step 3: If request is served then prepare the contract document otherwise search for new one.

Step 4: The new search checks whether the new service is posted and prepare the contract otherwise negotiation is done for the provided service. If service is not negotiable then customer can exit from the process.

Step 5: In this model modification can be done in the new service before or after negotiation, then bind and execute.

Step 6: After contract documents are sent to the request or the service is bind and executed, and specifications are sent to the registry.



III. CONCLUSION

The SOA paradigm has the potential to offer significant benefits to software systems development, maintenance, and reuse. However, many of SOA's benefits are not guarantee just by implementing a SOA. Therefore it is necessary for researchers and developers to address the important software architecture and reuse issues prior to creating a SOA. This paper has discussed the better framework for discovery and negotiation requirement for the new services and the architecture issue for adding new service to the repository. In summary, good architectural design practices and related issue are necessary for composing discovery and negotiation threads for new service.

REFERENCES

[1]. Tai, S., Mikalsen, T., Wohlstadter, E., Desai, N., Rouvellou, I., (2004). "Transaction policies for service-oriented computing" Data Knowl. Eng. 51(1), pp. 59-79.

 ISSN: 2454-4248
09 – 13

[2]. Deora, V., (2003). "A quality of service management framework based on user expectations" Proceedings of the First International Conference on Service Oriented Computing (ICSOC03), Springer, Heidelberg.

[3]. Patil, A.A.,Meteor-S, (2004). "Web service annotation framework" WWW '04: Proceedings of the 13th international conference on World Wide Web, pp. 553-562. ACM Press, New York.

[4]. Brown, A., Johnston, S., and Kelly, K., (2002). "Using Service- Oriented Architecture and Component-Based Development to Build Web Service Applications", A Rational Software White Paper, Rational Software Corporation.

[5]. Jaeger, M.C., Tang, S., (2004). "Ranked matching for service descriptions using DAML-S" Grundspenkis, J., Kirikova, M., (eds.), Proceedings of CAiSE'04 Workshops, pp. 217-228. Riga Technical University Riga, Latvia,

[6]. Ding, X., (2004). "Similarity search for Web services" Proceedings of the 30th VLDB Conference, pp. 372-383,

[7]. Deora, V., (2004). "Incorporating QoS specifications in service discovery" Proceedings of WISE Workshops, Lecture Notes of Springer Verlag.

[8]. Sarukkai, S., (2005). "Identity-Based Service-Oriented Architecture", Web Service Journal, Feb., http://www.syscon. com/story/?storyid=48038&DE=1

[9]. • Haibin Zhu, (2005). "Building Reusable Components with Service-Oriented Architectures" 0-7803-9093-8/05 IEEE.

[10]. Budd T., (2002). An Introduction to Object-Oriented Programming (3rd Ed.), Addison-Wesley.

[11]. Handorean, R. and Roman, G. C., (2002). "Service Provision in Ad Hoc Networks", Proc. of the 5th International Conference on Coordination Models and Languages, York, UK, April 8-11, pp. 207-219.

[12]. Jeng, J.J., (2001). "An Approach to Designing Reusable Service Framework via Virtual Service Machine", SSR'01, May, Toronto, Ontario, Canada, pp. 58-66.

[13]. Krueger, C. W., (1992). "Software Reuse", ACM Computing Survey, vol. 24, no. 2, June, pp. 131-183.

[14]. Ran, S, (2003). "A Model for Web Services Discovery with QoS" SIGecom Exch. 4(1), 1-10

[15]. Richard, G.G., (2000). "Service Advertisement and Discovery: Enabling Universal Device Cooperation", IEEE Internet Computing, vol. 4, no. 5, Sep/Oct, pp. 18-26.

[16]. TheDAML-S Coalition.:DAML-S, (2002). "Web Service Description for the semantic Web" Horrocks, I., Hendler, J.A., (eds.) The Semantic Web - ISWC2002, First International Semantic Web Conference. Lecture Notes in Computer Science.

[17]. Van den Heuvel,W.J., (2007). "Integrating Modern Business Applications with Legacy Systems: A Software Component Perspective". MIT Press, Cambridge.

[18]. Wang, Y., Stroulia, E, (2003). "Semantic structure matching for assessing Web-service similarity"Proceedings of First International Conference on Service Oriented Computing (ICSOC03), pp. 194-207. Springer, Berlin.