A Framework for Service Categorization and Requirement Gathering along with Discovery and Negotiation Threads of new Services in Service Oriented Architecture (SOA)

Ashutosh
Gautam Buddha University, Greater Noida

ABSTRACT

Service oriented architecture is an architectural principle that positions IT services as the primary means through which business services are offered by the organization to its ecosystem. Therefore, SOA offers the prospect of better alignment of Academic and Administrative goals and Information Technology (IT) solutions in Education Organizations.

The aim of this paper is to describe SOA in detail with considering all the approaches, concepts and methodologies that surrounds architectural model of SOA. Service based application development, frameworks and other related requirements are discussed in this paper in order to have a complete and accurate figure of SOA and be competent in utilizing service orientation concepts in enterprise application development. Service-Oriented Architecture (SOA) is a method for publishing services hosted by computer systems for the use of other computer systems. This method can be used to integrate applications and is therefore called Service-Oriented Integration (SOI). Integration brokers are a traditional method of integrating different kind of systems by sending messages from one system to another. This paper gathers requirements for an integration broker in Service-Oriented Architecture and presents framework that can be used to build SOI architecture.

1. INTRODUCTION

Service Oriented Architecture (SOA) is an architectural style and a combination of methodologies that aims to achieve interoperability of remotely or locally located homogeneous and heterogeneous applications by utilizing reusable service logic. Service orientation shows variation in adopting technology for implementation, rather than focusing on the technology itself, as SOA considers the description of the problem domain before concentrating on the usage of a specific execution environment.

This paper will approach to SOA from the following perspectives:

- Service oriented applications have a different approach to its layered architectural model. It is needed to describe and clarify what SOA is, the relationships of SOA with other architectures and development approaches, and the requirements of SOA in order to demonstrate how the architecture can be applied to software application development.
- Service design and development is a significant concept of service orientation. To investigate basic service design considerations, model-driven service development, and functional and operational aspects of services is important to have successful SOA implementation.

1.1 Categories all the services according to the requirement.

As per the requirements of the user the services are categorized. Service Categorization is very much important part for the service development. Services are categorized according to the type of service and user requirement.

1.2: Make specifications for all requirements.

Now when the service categorization is done the second objective is to specify the requirements these requirements are specified by the service broker and service user.

1.3: Design a model for describing these requirements and service categories for service provider.

Requirement categorization is done because when software is developed with wrong requirement the cost of software is increase rapidly while this requirement is corrected. Now a Framework Model is proposed for SOA for Service Categorization and requirement specifications.

SOA contains 6 entities in its conceptual model, described as follows [1]:

i **Service Consumer:** It is the entity in SOA that looks for a service to execute a required function. The consumer can be an application, another service, or some other type of software module that needs the service. The location of the service is discovered either by looking up the registry, or if it is known, the consumer may directly interact with the service provider.

ii. **Service Provider:** It is the network addressable entity that accepts and executes requests from consumers. It provides the definite service description and the implementation of the service. The service provider can be a component, or other type of software system that fulfills the service consumer's requirements.

iii **Service Registry:** It is a directory which can be accessible through network and contains available services. Its main function is to store and publish service descriptions from providers and deliver these descriptions to interested service consumers.

iv **Service Contract:** A service contract is the description that clarifies the way of interaction between the service consumer and provider. It contains information about request-response message format, the conditions in which the service should be executed, and quality aspects of the service.

v. **Service Proxy:** It assists the interaction between service provider and service consumer by providing an API written in the local language of the consumer. It is supplied by the service provider and a convenience entity for the consumer. As well as it can enhance

performance and provides caching facilities. Service proxy is an optional entity in SOA.

vi. **Service Lease:** It specifies the amount of time that a service contract is valid. It is managed by registry and determines the executive well-defined timeframes of binding to the services. Usage of service lease supports loose coupling between service provider and consumer and maintenance of state information for the service.

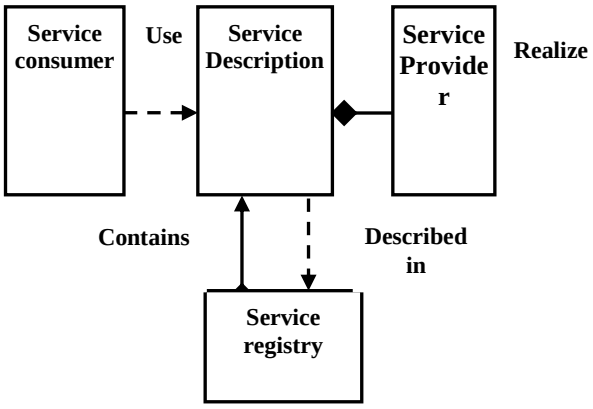

Fig. 1: Service Oriented Architecture Conceptual Model

Table 1: Comparison of Architectural Development Models

Structure Development	Object Oriented Development	Component Based Development	Service Based Development
Very fine system structuring	Small – grain system structuring	Medium – grain system structuring	Coarse – grain system structuring
Low reusability	Low reusability	Medium reusability	High reusability
Tight coupling	Tight coupling	Loose coupling	Loose coupling
Have compile time dependencies	Have compile time dependencies	Have compile time dependencies	Have only run time dependencies
Intra-application communication scope	Building blocks are individual classes	Building blocks consist of several classes (components)	Building blocks consist of components
	Encapsulation, Inheritance, Polymorphism	Interactivity, connectivity, and exchangeability of components	Published interface definition
	Functionality is described by class declarations	Functionality is described by interface declarations	Dynamically discoverable distributed services
	Dynamic but large number of connected object		Functionality is described by network addressable component interface declarations
			Inter-enterprise communication scope

2. REQUIREMENTS FOR SOA

It should be clear that it's important to develop an architecture that meets all of your requirements. These requirements should include the ability to:

i) Leverage existing assets. This is your most important requirement. Existing systems can rarely be thrown away, and probably contain within them data that is of great value to your enterprise. Strategically, the objective is to build a new architecture that will yield all the value that you hope for, but tactically, your existing

systems must be integrated so that, over time, they can be componentized or replaced in manageable, incremental projects.

ii) Support all required types of integration. These include user interaction (to provide a single, interactive user experience), application connectivity (to deliver a communications layer that underlies the entire architecture), process integration (to choreograph applications and services), information integration (to federate and move your enterprise data) and build to integrate (to build and deploy new applications and services).

iii) Allow for incremental implementations and migration of assets. Fulfilling this requirement will enable one of the most critical aspects of developing the architecture: the ability to produce incremental ROI. Countless integration projects have failed because of their complexity, cost and unworkable implementation schedules.

iv) Build around a standard component framework. You must include a development environment that is built around a standard component framework to promote better reuse of modules and systems, allow legacy assets to be migrated to the framework and allow for the timely implementation of new technologies.

v) Allow implementation of new computing models. Specific examples of this requirement include new, portal-based client models, grid computing and on demand computing.

3. FRAMEWORK FOR REQUIREMENT GATHERING AND SERVICE SPECIFICATION:

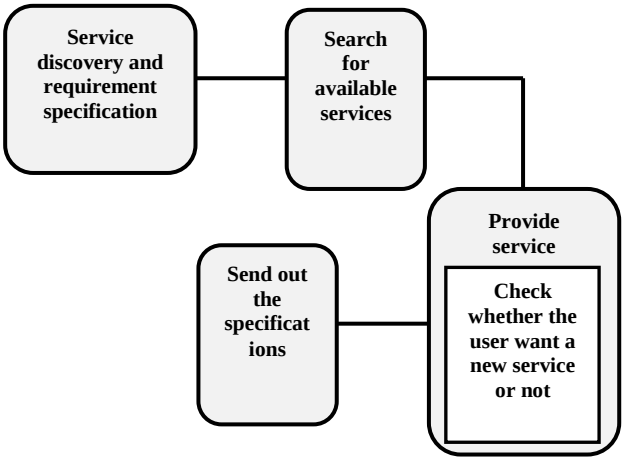

Fig. 2: framework for requirement gathering and service specification

Service oriented architecture is an emerging approach that address the requirement of loosely coupled, standards based and protocol independent distributing computing. To build an SOA a highly distributable communications and integration backbone is required. This paper presents the unified framework for discovery and negotiation requirement for new services in Service Oriented Architecture. Specially, this paper 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.

4. MODEL FOR DISCOVERY AND NEGOTIATION OF THE NEW SERVICES

This paper proposes a combined model for service discovery and negotiation of new service. In the paper [18] the discovery method and new service negotiation method is discussed but as a different different processes. 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.

Step 1: In this model first of all service and negotiation thread is discovered, and specifications are searched.

Step 2: Now search for these specifications, if specifications are exactly found and available service is searched then got the service. And if no service is found then reject the request. And search that the new services are posted or not. And with this check for interface with matching parts and these are available then request for the interface.

Step 3: After searching for the available service ask to the customer that he got the service or not. If service is taken then contract document is prepared and if service is not taken then go to the new search.

Step 4: Now in the new search process first of all check whether a new service posted or not then a contract is prepared and if service is not taken then negotiate. And if not negotiable then there is an exit a new search option which is taken by customer based on his choice. If services are taken from available services then documentation for requestor is also done.

Step 5: In this model modification also possible in the new service after negotiation any modification can be done if want to. And then bind and execute.

Step 6: After contract document sent to the requestor and service bind and execute service specifications are sent out.

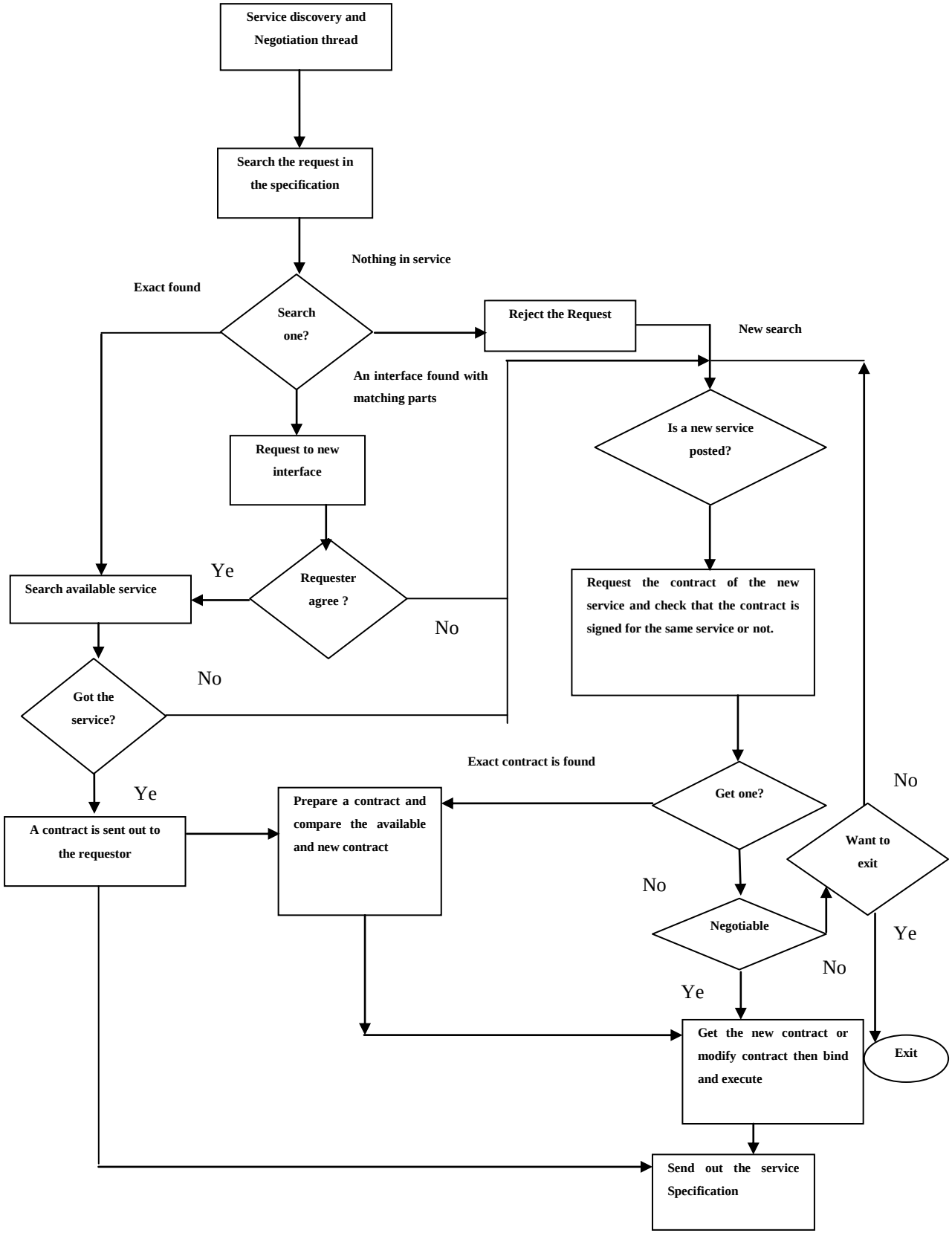

Fig. 3: Flowchart for Model for Discovery and Negotiation of the New Services

5. SERVICE CATEGORIZATION

a. ***Utility services (Infrastructure services)*** – usually defined by grouping together infrastructural capabilities with a common purpose, for example a Logging Service, and thus granularity is directly defined by the utility functions supported;

b. ***Entity services –*** these have their functional context scoped by the entities that they manage. For example,

granularity of a Student Management Service is defined by the Student entity. In this case Student Management Service would include all the capabilities that are necessary to maintain the Student entity;

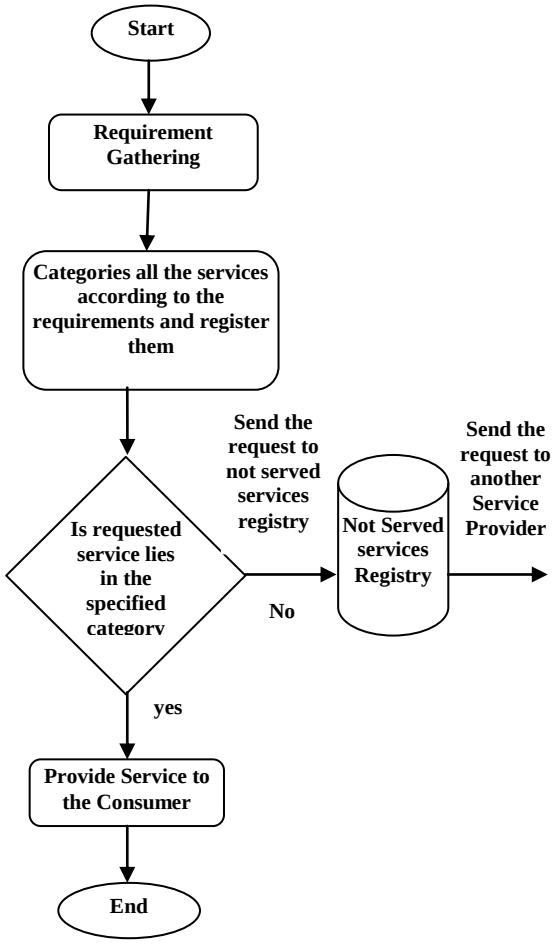

Fig.4: Flow chart for requirement gathering and service categorization

c. ***Task services*** **–** these typically contain a group of capabilities related to the same business task, for example Tax Calculation Service. Task services tend to be fairly granular;

d. ***Process services*** **–** these usually deal with a larger functional context defined by the encapsulated business process. For example, Financial Aid Management Service would include all the capabilities necessary to manage financial aid.

e. ***Composite services*** **–** these are the services which are defined by grouping the task service and process service. Composite service include whole process with task like if a user want to calculate only the amount of tax which he/she is paying through the financial management process then user can easily get the service.

6. CONCLUSION

Service oriented architecture is an emerging approach that address the requirement of loosely coupled, standards based and protocol independent distributing computing. To build an SOA a highly distributable communications and integration backbone is required. This paper presents the unified framework for discovery and negotiation requirement for new services in Service Oriented Architecture. Specially, this paper 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.

The purpose of this paper was to find requirements and present a Framework in Service-Oriented Architecture. And to described different integration architectures and technologies that can be used to build Service-Oriented Integration architecture. This paper also concentrated on requirement specification because if requirements are not matched with the specifications then the service is not developed correctly.

7. REFERENCES

[1] Thomas Erl "*Service-Oriented Architecture*" Pearson Education (Third Publication) 2007.

[2] Mohammad Kazem HAKI, Maia Wentland Forte (2010) "*Service Oriented Enterprise Architecture Framework"* IEEE.

[3] Farzaneh Mahdian, Vahid Rafe, (2010) "*Different Models of Dependable Services in Service-Oriented Architecture*", 978-1-4244-6542-2/,IEEE

[4] Broy, Diernhofer, Grünbauer, Meisinger, Rappl, Rittmann, Schätz, Schoenmakers, Spanfelner, "*Service-Oriented Development*" white paper.

[5] Olaf Zimmermann, Niklas Schlimm, Günter Waller, Marc Pestel, "*Analysis and Design Techniques for Service-Oriented Development and Integration"*.

[6] Gregor Hohpe (2005) "*Developing Software in a Service-Oriented World*" white paper.

[7] Werner Vogels, (2003) "*Web Services are not Distributed Objects*"IEEE Internet Computing.

[8] Xin Yu, Yu Bai (2009) "*A Method for Accessing Trusted Services Based on Service-Oriented Architecture*", Fifth International Conference on Information Assurance and Security.

[9] Yukyong Kim, Hongran Yun, (2006) "*An Approach to Modeling Service-Oriented Development Process*"IEEE International Conference on Services Computing (SCC'06)0-7695-2670-5/06.

[10] Martin Kuba,Ondrej Krajicek (2007) "*Literature search on SOA, Web Services, OGSA and WSRF"* IEEE.

[11] Zhiyi Ma, Xiao He, Lianghuan Kang in the paper (2009) "*A Model Driven Development Platform for Service-Oriented Applications*" 978-0-7695-3812-9/09 IEEE.

[12] Fahmideh, Mohsen Sharifi, Pooyan Jamshidi, Fereidoon Shams, Hassan Haghighi "*Process Patterns for Service-Oriented Software Development*" IEEE.

[13] Jiaqing Yu, Jianzhong Cha, Yiping Lu, Shasha Yao (2008) "*A Service-Oriented Architecture Framework for the Distributed Concurrent and Collaborative Design*" 978-1-4244-2013-1/08/ IEEE.

[14] Tao Zhang, Shi Ying, Sheng Cao, and Xiangyang Jia (2006) "*A Modeling Framework for Service-Oriented Architecture*" Proceedings of the Sixth International Conference on Quality Software (QSIC'06)0-7695-2718-3/06 IEEE.

[15] Adil kenzi, Bouchra El Asri, Mahmoud Nassar, Abdelaziz Kriouile (2009) "*A model driven framework for multiview service oriented system development*" 978-1-4244-3806-8/09/ IEEE.

[16] Francoise Baude, Virgine Legrand (2011)"*A component-based orchestration management framework for Multidomain SOA*" 978-1-4244-9221-31111 IEEE.

[17] Nestor Riba, Humberto Cervantes (2007) "*A MDA tool for the development of service-oriented component-based applications*" Eighth Mexican International Conference on Current Trends in Computer Science 0-7695-2899-6/07 IEEE.

[18] Alberto Gonz´alez, Eric Piel, Hans-Gerhard Gross (2009) "*A Model for the Measurement of the Runtime Testability of Component-based Systems*" IEEE International Conference on Software Testing Verification and Validation 978-0-7695-3671-2/09.

[19] Gerald Kotonya, John Hutchinson (2007) "*A Service-Oriented Approach for Specifying Component-Based Systems*" Sixth International IEEE Conference on Commercial-off-the-shelf (COTS)-Based Software Systems (ICCBSS'07) 0-7695-2785-WO7.

[20] Michael Jiang, Allan Willey (2005) "*Architecting Systems with Components and Services*" 0-7803-9093-8/05/ IEEE.

[21] Georg Buchgeher, Rainer Weinreich (2009) "*Tool Support for Component-Based Software Architectures*" 16th Asia-Pacific Software Engineering Conference 1530-1362/09 IEEE.

[22] Eun-Ju Park, Haeng-Kon Kim, Roger Y. Lee (2007) "*Web Service Security model Using CBD Architecture*" Fifth International Conference on Software Engineering Research, Management and Applications 0-7695-2867-8/07 IEEE.

[23] Liam O'Brien, Paulo Merson, Len Bass (2007) "*Quality Attributes for Service-Oriented Architecturess*" International Workshop on Systems Development in SOA Environments (SDSOA'07) 0-7695-2960-7/07 IEEE.

[24] Julie Street, Hassan Gomaa (2008) "*Software Architectural Reuse Issues in Service-Oriented Architectures*" Proceedings of the 41st Hawaii International Conference on System Sciences.

[25] Hassan Gomaa, Koji Hashimoto, Minseong Kim, Sam Malek, Daniel A. Menascé (2010) "*Software Adaptation Patterns for Service-Oriented zsgfdfdgrt5ch65Architectures*" 978-1-60558-638-0/10/03 ACM.