J. Basic. Appl. Sci. Res., 2(4)3432-3436, 2012
© 2012, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Conceptual Overview of Service-Oriented Software Systems Development

Saeid Kamari

Computer Department, Sahneh Branch, Islamic Azad University, Kermanshah, Iran

ABSTRACT

Service-oriented architecture is a paradigm for design and development of distributed and heterogeneous software systems. Service-oriented systems design is one of the major issues in software engineering. One of the main concerns in service-oriented systems design is to generate services with greater reuse capability, so that they can be reused in other systems. In this paper, according to the design process in a service-oriented systems life cycle, different system design approaches have been studied and the problems and benefits of each of them, the common methodologies to develop service-oriented systems are discussed. Moreover, according to the importance of the services identification in the service-oriented architecture and its performance as a key and vital operation in the development process, available patterns in this field are discussed and then will be evaluated from the functionality perspective. Obviously getting familiar with the strengths and weaknesses of different methods of systems design and service identification in widely used service-oriented architecture help designers to select the best solution in design and produce their desired systems.

KEY WORDS: Service-oriented architecture (SOA), Web service, Software system life cycle, Development methodologies, Service identification.

I. INTRODUCTION

Service-Oriented Architecture (SOA) as one of the most prominent architectures in the past decade, with creating a layered architecture and the introduction of basic entities called services try to implement distributed and heterogeneous software systems [1, 2, 3]. Service-oriented architecture is a paradigm for creating, realization and maintenance of business processes distributing in big hetereogenous systems [7].

Web services technology is an effective mechanism to share application logic between different machines with different operating systems and development environments. Web service is a software system that is determined by a URL and its interfaces and bindings are defined and described by XML. This definition can be discovered by other software systems. These software systems can then connect to other Web services using XML-based messaging protocols over Interne. Web services are the most successful example of practical implementation of SOA [8].

With acceptance of service-oriented architecture as a new approach in software engineering as a model for the production of software systems, and in order to practical use of SOA - similar to other existing methods – some SOA specific methodologies are required for system development with emphasis on service-oriented approach. According to dependability of SOA to services, one of the key issues in service-oriented architecture is to identify the services needed in the organization processes and generate their interfaces. By generating the service interfaces, it will be possible to search required organization services or produce new services in design phase in development methodology [4, 10].

This paper is organized as follows: In Section 2 the basic concepts used in the paper are presented. In Section 3 the proposed service-oriented systems life cycle is discussed. Section 4 describes available SOA-based systems development approaches. Section 5 also provides a common service-oriented systems development methodology. Different approaches of services identification particpating in business processes are described in Section 6 and finally in section 7 the conclusion is offered.

II. BASIC CONCEPTS

In this section the basic concepts and technologies needed in this paper are introduced.

A. Service Oriented Architecture

Service-oriented architecture provides a layered architecture trying to create architecture with different abstraction levels. It separates various concepts in each system software development such as issues related to user interfaces, synchronization, services and resources. According to the multi-layer architecture and using open standards at every level, service-oriented architecture can be an appropriate option for the implementation of software systems in a heterogeneous distributed environment. Service-oriented architecture various layers are shown in Figure1 [2]. The SOA resource layer comprises of all available information sources such as the previous programs, databases, systems management and even people and knowledge in the organization. In the higher layer services are located on the organization resources based on standards and specifications outlined in the service-oriented architecture as a coating or are generated from scratch. This layer creates an integrated and extensible infrastructure for interaction between the services by building independent services with well-defined interfaces. Due to the importance of synchronization in distributed systems and the separation of collaboration from computational problems, a separate layer for synchronization is considered in SOA. In this layer, all the standards and mechanisms of service coordination and combination in terms of business processes are implement. The task for the last layer is to provide an integrated interface to access and manage the entire system.

***Corresponding Author:** Saeid Kamari, Computer Department, Sahneh Branch, Islamic Azad University, Kermanshah, Iran.
Email: s.kmari@shbu.ac.ir

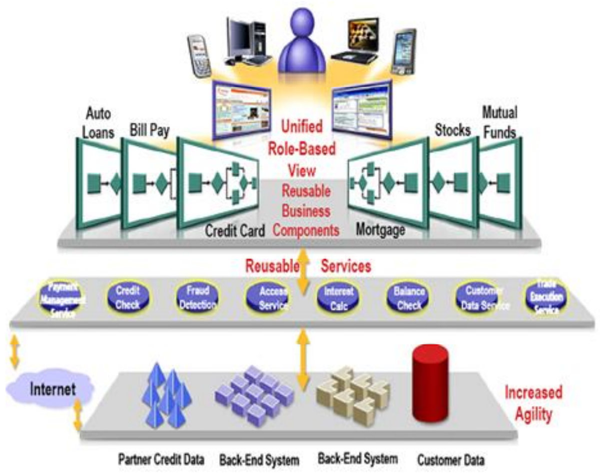

Figure1. SOA layers

B. Web services

Web services are a set of stateless, autonomous, coarse grained, platform-independent and loosely-coupled software components, which are implemented under a specific namespace. Web services and SOA-based practical systems using the internet as the connection media. In most definitions of SOA, Web services are introduced as a successful implementation of this architecture, whereas SOA can be implemented with any service based technology [8].

The web service model is composed of three basic entities: service provider, service registry and service consumer [9]. Service provider creates the service with a well-defined and standard interface and then publishes it in a centeral registry. Service registry includes information such as address and the way to contact the supplier and technical information about the service. Service consumer obtains the needed information from the registry and then binds to it using service description to invoke its methods. To enable communication between different applications, with different languages and platforms, for each of the publication, some searching and binding appropriate standards are adopted. The Web service model is shown in Figure2.

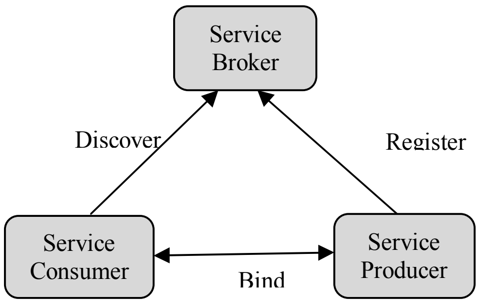

Figure2. Web service model

C. Types of Services in SOA

Services are basic elements of service-oriented architecture. Due to the large variety of business functions, services are employed for different purposes and also play different roles. Services are parts of the business process. Therefore, to identify and extract the service, it is needed to view organization's business processes. Business process is a task that must be done in an organization. Moreover, the set of activities needed to do the work is characterized in process [7].

From a technical perspective, services are divided into three distinct classes:

1- Basic services- that provide only a specific business function and don't seem to decompose into smaller services. These are the most fine-grained services.
2- Composed services- that represent a fine workflow within a business process. From the granularity perspective, these services are located in a tier above the basic services. These are built from available and basic services.
3- Process services- that are the most coarse-grained web services. From a business perspective, a process service shows a large workflow of activities or services.

III. Service-oriented Systems Life Cycle

The life cycle of software systems based on service-oriented architecture, includes four different phases: Identification, analysis and design, implementation and testing. At the first phase, different users' requirements and expectations are extracted. At the analysis and design phase, available services in organization processes according to users and owners concerns are produced. In implementation phase, identified services are combined to create higher-level services to meet customer needs. In the test phase, the service functions and processes created by the users' requirements and criteria's are tested. This phase can be done repeatedly till to cause the users and owners satisfaction and completely match to their needs. Different phases of the SOA-based software systems production and their inter-relationships are shown in Figure3 [10].

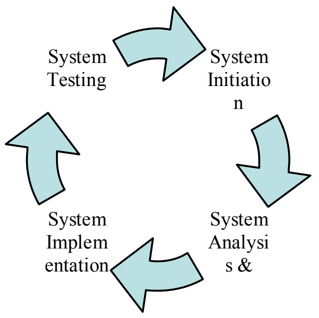

Figure3. Service-oriented systems life cycle

IV. SERVICE-ORIENTED SYSTEMS DESIGN AND DEVELOPMENT APPROACHES

There are numerous questions and problems in design and development of Service-oriented software systems: How to identify services which are parts of business processes? How to decompose a system into smaller parts (services), so that they can be implemented? How to design and build services with most reuse capability, so that they can be used in more scenarios and collaboration protocols? To solve these problems and to answer these questions, there are generally three approaches [10]:

- *Top-down approach*, in which the problem, system or process is broken down into smaller pieces till to reach basic services. Because of top-down view, this approach helps designer to realize what services are needed in the system and understand the services and activities distinction.
- *Bottom-up approach*, in which smaller services are combined to create larger parts and processes. In other words, this method tries to use software assets and services to build processes and the system.
- *Agile approach*, which is a combination of two previous methods. At first, candidate services are identified by top-down analysis and then tries to select services from available software assets by balancing the extracted functionalities and existing services and processes. It causes to reduce system development costs.

Each of the first two approaches are assumed to have specific problems. In the first method, the designer developes the system by describing the abstract services which are later discovered and replaced by real ones. The major difficulty of this method is to find suitable concrete services that most consistent with candidate ones. In second approach, designer is aware of existing services and tries to develop a system which is able to interact and collaborate with such services. To do so, it is also required to know complete information about the service such as the service location in registry at first. Morever, because of pre-defined features and functinality of existing services, there may be less flexibility in service combination. So, it may be possible to generate servcies which are not completely fit to users' expectations and requirements.

V. SERVICE-ORIENTED SYSTEMS DEVELOPMENT METHODOLOGY

Given the problems that both top-down and bottom-up approaches in the design and development of SOA-based systems have, the third approach seems to combine these two methods together with the advantages of each of them to solve the existing problems. Thus, an intermediate design methodology to produce SOA-based software systems is needed. In this methodology, the design process needs stakeholders' requirements and objectives analyzing at a level of abstraction. Different conceptual SOA-based software systems analysis and design steps and their relationship are shown in Fig.4.

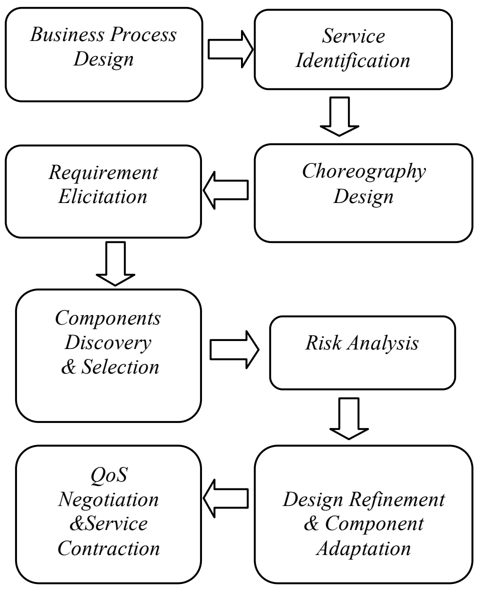
Business Process Design
Service Identification
Requirement Elicitation
Choreography Design
Components Discovery & Selection
Risk Analysis
QoS Negotiation &Service Contraction
Design Refinement & Component Adaptation

Fig4. Service-oriented system development methodology

Generally, the given development framework has the following steps are provided [4]:

1. *Business Process Design.* The overall process begins with the business process modeling phase. Thus, stakeholders are defined with their high-level goals and basic Key Performance Indicators (KPIs), which measure the global performance from the viewpoints of each stakeholder.

2. *Service Identification.* This step is focused on separating loosely coupled functional parts of the collaborative business process into standalone service components. Considering that business processes evolve in time in order to adapt to business strategy which changes following market rules (i.e. new products, new services, new business models), we may observe that candidate services can be identified as:
   - repeated blocks of activities inside a business process;
   - similar blocks of activities among various business processes or different applications;
   - time-invariant blocks of activities in a time-variant business process.

   An accurate analysis of the activity diagrams produced at the first step is essential to identify these blocks of activities and to make them eligible to become Web services. The analysis process can be partially automated by searching for similar graphical patterns in the UML activity diagram representing abstract business processes.
3. *Choreography Design.* Desired communication patterns among identified abstract services are defined using UML sequence or collaboration diagrams based on the interaction patterns among stakeholders, Web services and the system. Choreography model can be used to define collaboration protocols.
4. *Requirement Elicitation.* Requirements for discovery of existing software components and Web services (inputs/outputs, pre-conditions/effects, behavior patters and desired quality levels) are defined based on the identified abstract services.
5. *Components Discovery and Selection.* Software components that can be used to compose the system are of two essential types:

- *Modifiable*, i.e. available as documented source-code (e.g., open source projects) or coming from organizations available to customize it (e.g., software houses);
- *Unmodifiable*, i.e. coming from providers not particularly interested in customization or not customizable at all (e.g., Web services).

  This step covers discovery of Web services (interfaces, behavioral specifications and coordination patterns specified at the previous stages). Then, the QoS information about Web services and their providers must be collected (using data published by service providers, authorized agencies or other service clients), in order to select the best candidate Web services.

6. *Risk Analysis.* This step is needed to assess risk related to use of external Web services (loss of service, loss of data, security/privacy concerns, etc.). On the basis of this assessment, the existing risks can be mitigated through selection of more appropriate services, use of alternative services for critical tasks, system re-design, data replication, and so on.
7. *Design Refinement and Component Adaptation.* At this step the prepared models can be refined to allow for the seamless integration of the external Web services. Found Web services and existing legacy systems are tested and analyzed in order to decide how to introduce them to the system. It may happen that adaptors or wrappers are needed. Providers of chosen services become stakeholders of the system. If no services with required functionalities are found, the organization should think about implementing them, thus increasing code reusability both in its own future projects and in the projects of third parties.
8. *QoS Negotiation and Service Contraction.* If quality parameters of discovered Web services do not correspond to identified KPIs, they can be negotiated with service providers. At this step identified KPIs should be mapped into direct requirements for QoS further resulting in Service Level Agreements (SLAs).

VI. SERVICE IDENTIFICATION APPROACHES

Based on the presented methodologies for SOA-based system analysis and design, it can be said that the process of generating concrete services according to users' specifications includes four distinct steps. During the process, concrete services are produced by reducing services abstraction levels. Different system analysis and design steps of SOA-based systems are shown in Figure5 [5, 6, 11].

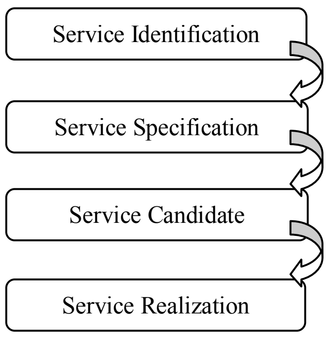

Figure5. Service-oriented systems analysis and design steps

Each step focuses on some aspects of service generation:

1. *Service Identification*: In this step services participating in organization processes and collaborations are identified based on users needs and without mention of low-level behavioral and functional details.
2. *Service specification*: In this step, the service abstraction level is reduced according to collaboration protocols, recognized relation patterns using choreography model. Behavioral details and interfaces of services are then extracted based on recognized features.
3. *Service candidate*: By defining service interfaces, it is now possible to find candidate services. It is time to search

service providers for recognized services to find the best suitable and consistent services based on extracted requirements and interfaces. The goal of this step is to find and select services which are most compatible with customer needs and service requirements.

4. *Service realization*: This is the last step of service generation process in which identified services are created in the form of real and concrete entities.

Some authors believe in service identification as the heart and core of service-oriented architecture. However, there is no agreement on service concept, its identity and goal. So, different approaches for service identification by focusing on special aspects of services have been proposed. Table 1 summarizes some of the significant service identification methods [12].

Table1. Different service identification approaches

Method	Description
Business Processes	Business processes is decomposed into sub-processes or modular activities. The most low-level tasks contain small and concrete logical working units which are supported by functionalities presenting separate services.
Business Functions	In this method, the relatively stable business activity model acts as the basis for service identification.
Components	In this method, IT-functionalities are divided to the components with most concrete and worst loosely-coupled. These components interact with each other by calling other services.
Non-Functional	In this method, non-functional requirements define the basis of services separation and their boundaries.
Responsibilities	It's not a real and practical method. This introduces an owner for each service which is responsible for it.
Infrastructure	In this method technical infrastructure forms the basis of service separation.
Existing Supply	In this method, services are generated by existing tools and system wizards.
Business Goals	In this method, business goals are decomposed to reach supporting services.

VII. CONCLUSION

Widespread use of service-oriented architecture and Web services represents their high acceptance and efficiency. This has caused that owners of information technology and software engineers do increasing efforts to simplify and automate various stages of design and development of software systems based on service-oriented architecture. In this paper, by pointing to exact position of the design stage in the life cycle of service-oriented systems, a methodology for designing systems based on service-oriented architecture was presented.

According to importance of service identification in service-oriented systems design, various approaches were described for this purpose. Services identification in the service-oriented architecture and extract them from the organization processes is vital and important. In order to generate interface of services participating in a process, it is necessary to model process executive logic and the way of their collaboration primarily. As a future work, process description models such as orchestration and choreography can be used to produce service participants in processes [13, 14]. Orchestration refers to an executable business process that interacts with internal and possibly external web services of organization. It describes how services interact in message level, how business process logic flows and also shows order of interactions execution. In choreography model, each participant states its contribution in process interactions. This standard traces sequence of message exchange between the various partners and resources. In other words, the model defines how a set of services collaborates in order to achieve a common goal [9].

REFERENCES

[1] T.Erl, "Service Oriented Architecture: A field Guide to Integrating XML and Web Services", Prentice Hall PTR, 2004

[2] Newcomer E, Lomow G,"Understanding SOA with Web Services", Addison Wesley Professional, December 14, 2004.

[3] Erl T, SOA: Principles of Service Design, Prentice Hall, 2008.

[4] H.Rob, S.Kinder, S.Graham, “IBM’s SOA Foundation: An Architectural Introduction and Overview”, November 2005.

[5] M. Bell, "Service-Oriented Modeling: Service Analysis, Design and Architecture”, John Wiley& Sons, ltd, 2008.

[6] M.Bochicchio, V.D'Andrea, N.Kokash, F.Longo, "Conceptual Modeling of Service-Oriented Systems", University of Salento, University of Trento., Italy, 2007.

[7] N.Josuttis, "SOA in practice", O'REILLY, 2007.

[8] E.Cavanaugh, “Web Services: Benefits, challenges, and a unique, visual development solution”, Altova white paper, 2004.

[9] S.Dustdar, W.Schreiner, “A Survey on Web Services Composition”, Int, J. Web and Grid Services, Vol. 1, No. 1, 2005.

[10] M.Popozoglou, W.Van der Heuvel, “Service-oriented Design and Development Methodology”, Int.J. Web Engineering and Technology, Vol.2, No.4, 2006.

[11] R.Boerner, M.Goeken, “Service Identification in SOA Governance”, Third IEEE International Conference on Digital Ecosystems and Technologies, 2009.

[12] L.I.Terlouw, A.Albani, “Identifying Services in SOA”, ICIRS consulting & research, 2009.

[13] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., "Web Services Choreography Description Language Version 1.0.", W3C Working Draft 17 Decem- ber 2004, World Wide Web Consortium ,2004.

[14] Peltz C., "Web services orchestration-a review of emerging technologies, tools and standards", Technical report., Hewlett Packard Co., January 2003.