Service Oriented Architecture Essentiality as a Best-Practice for the Development of Large Software Projects

Atefeh Khosravi
Islamic Azad University-Tehran Northern Branch, Tehran, Iran
Email: ati.khosravi@gmail.com

Nasser Modiri
Islamic Azad University of Zanjan, Zanjan, Iran
Email: nassermodiri@yahoo.com

***Abstract*—The success of a software development process always has been an obsession. There are loads of efforts to design and develop applications according to customers' requirements and their business process functionality. As technologies and frameworks for software production grow, choosing the most appropriate method becomes one of the critical decisions to lead a project to success. Experts are always looking for solutions to produce a software with high quality, in associated with customers desires, and with acceptable flexibility and logical cost which has the maximum efficiency. Software maintenance, extendibility, competency and changeability are some of important factors which master decision makers consider them to choose the most convenient method in software development. Service oriented architecture is an architecture which helps to provide these key features of software products. In this paper we will examine the benefits which lead us to utilize service oriented architecture in large application development.**

***Index Terms*—SOA, service oriented applications, software development, large software projects**

I. Introduction

The main idea of SOA is based on composition of object oriented architecture and component base architecture [1].

In this architecture developers are divided into three independent but cooperative groups: Application builders or services clients, service brokers and service providers.

Service providers' task is to provide independent and loosely coupled services. Service brokers' duty is service introduction and marketing. Application builders find their required services for constructing their applications via service brokers.

SOA can be consumer centric. In this way application builders announce their requirements and service providers supply them by services [2]. In fact the most important goal of SOA is to provide services which are exactly what consumers want and are as much independent and reusable as possible. And without depending on specific platforms they can make development process faster so that the cost of development decreases. Services should support activities in business process. It is possible that designer decide to design several services for an activity because some part of this activity may be used in other business processes. Or it is possible to design one service for several activities. Despite the designer approach in designing services, it is important for him or her to keep major feature of services (independency and reusability) in him or her mind.

Fig. 1 demonstrates the position of service consumers and service providers, and the relation between business processes and services.

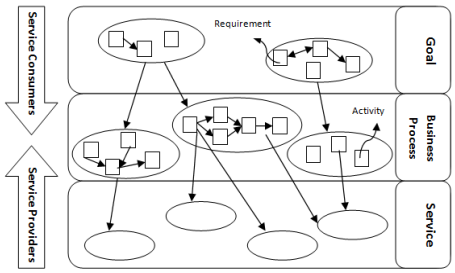

Figure 1. Relation between business processes and services

In this paper we will examine five important factors which are vital for developing large software projects and we will discuss how service oriented architecture can assist us to achieve them easier. Next section focuses on project division benefits and the gentle effect of SOA on it. Section 3 demonstrates how SOA eases maintainability of the large projects. After that we will show the importance of business and UI separation in large projects, and the effect of SOA to achieve it. Two last sections will focus on customization and competency which SOA brings and their benefits in the large software productions.

II. Project Division

Manuscript received October 15, 2012; revised December 24, 2012.

doi: 10.12720/joace.1.2.170-172

SOA is a way of organizing software applications and support infrastructure into an interconnected set of services, each is accessible through standard interfaces and messaging protocols. Once all the elements of an enterprise architecture are in place, existing and future application can access these services as necessary without the need of convoluted point-to-point solutions. This architectural approach is particularly fruitful when multiple applications running on varied technology and platforms needs to communicate with each other [3].

According to this feature, we can divide projects to some small and independent applications, so that we can delegate the responsibility of each application to a specific team having special experience in the business domain of that part. For instance to produce bank applications which are often complicated, we can divide the application to retail banking application, modern banking application, swift application, card and switch application and … . As a result by reducing the complexity of it, not only the agility will increase but also project management and maintenance will become easier.

In addition, as SOA is service based and services are supposed to be as independent as possible, we can outsource services. So that in some domains which the organization has not enough experience, it can utilize the knowledge of other organizations who have implemented similar projects. This will end to shorter development time and also less future malfunctionalities. For example it is possible to outsource the exchange system as its functionality is so different from routine tasks such as account and deposit management. We can even connect different existing functionalities by using some wrappers and decorators to expose them as standard services, so that we can integrate them as if they were one application.

III. Maintenance

Maintenance is much easier in SOA as services are designed loosely-coupled. Since the services have the least or no dependency, change management is simpler. Because changes won't have unpredictable side effects on other parts and services.

On the other hand services are designed reusable and each provides specific functionality. So that to change functionality in a business process it is just needed to apply changes in corresponding service and by doing so, our desired change will be observed in every single part of system which need to use that service functionality. For instance issuing voucher is a common functionality among different activities of banking. In the SOA we consider issuing voucher as a service so that if we need some changes, say extra wage to issue a voucher, we can simply apply changes just to this service, and the changes will be seen in all parts like deposit settlement, deposit withdraw, direct transfers, and which have utilized this service because of its reusability.

Service orientation facilitates monitoring and controlling functionalities. As all of functionalities are as services, any request is consider as a service call, so that traceability and service accounting and any information about service calls can be monitored. So it is possible to recognize which service is called by who and when, and what are the service inputs and outputs. In addition it is possible to control availability and authority to permit a service call. For instance displaying the balance sheet in a banking system can be considered as a service. But in banking just the branch manager needs to view it, so we can put an authority check on this service, and before going through its functionality it is possible to check the access permissions of the requester. If the requester has the permission to run this service, the request will be served and the result will be sent to requester.

IV. Business and UI Separation

One of the most essential issues in application development is to consider separation of business and user interface (UI). This will have a very considerable effect on the quality and maintenance of end products. Because when some changes happen to the business, this separation enables the system to continue its functionality without any changes to the UI.

Lack of business and UI separation forces us to scan the whole UI, to detect any effect of business changes in, and also to detect the side effects of changed UIs!

In addition it decreases readability and flexibility of codes which ends in maintenance cost increment. For example imagine that for some security reasons we decide to control some permission before accessing to data base. To do so, we need to add a method before connecting to DB, in a mixed business and UI development there will be no choice other than scanning the whole codes to detect places where have direct access to DB. However, if we separate business and UI, any changes can be handled much more easily by applying changes just to the components supplying that business.

SOA is an architecture which provides the ability of business and UI separation, so it helps improving quality and maintenance. So applying some changes or optimizations in a corresponding service of a business will be reflected through the UI of every single part of a system which uses that service.

V. Competency

Being distinctive to overcome rivals has been always one of organizations' desires. They always try to find new solutions to leave their rival behind. Decreasing the price of products is the first option that comes to mind. It will definitely absorb more clients. However, they put themselves in a detrimental competition since it leads to erosion and the loss of gained profit of service providing, so that it is not applicable.

Another solution to overcome the rivals is to provide new and innovative services. According to its architecture, SOA facilitate it. For instance in an internet-banking system if we want to add new features like currency exchange, we can simply implement the corresponding services. SOA increases agility in business process because of it separates interface and implementation [4]. So that whenever we need to provide a new functionality, we can append it to the system by implementing the

interface of corresponding service without involving in its implementation or details.

Furthermore sometimes it is necessary to provide more vast services. For example, to create a portal for a customer to collect all his financial history, we require the intra-banking cooperation. As SOA provides service interface, we can connect different applications using it. It is noticeable that connected applications' nature is not necessarily the same. For example it is possible to connect SCM[5] and CRM[6] using SOA to attract more customers and detect their preference so that ends in purchasing pattern prediction recognition which causes cost reduction in goods maintenance and shipping.

VI. Customization

Another reason to choose SOA for massive projects is the customize-ability which is supported by this architecture.

Services just represent their interfaces to the consumers and hide their implementation and details, consumers can develop their application regardless of the platform they use, and it is just needed to utilize the services according to their interfaces. Even it is possible to use different platforms to develop different part of an application because of some security or technical issues. For instance it is possible to develop the basic information definition part of a banking application in a web-based manner because just the central branch should have access to this part, and for critical situations it better to have access to this part remotely. Whereas other parts can be developed as desktop application as they are used in tellers.

On the other hand by using SOA, we can utilize some middlewares to configure the service requested processing methods. For instance according to our facility, we can process requested services by multiprocessing or multithreading techniques or even their combination. In addition the independency of services helps us to be able to utilize grid computing and cloud computing.

SOA enables plug-in based programming and producing flexible coexisted applications [7]. For instance we can develop a general banking application using SOA, and for each specific customer (bank), it is just needed to replace current plug-in by that customer specific plug-in. for example in some banks which don't offer loan facilities, we can simply unload or remove the corresponding extension, without having any side effect on the rest of application.

VII. Summaries

In this paper we introduced the key feature of SOA and the benefits of using it to develop large software projects.

Maintenance, competency, customizability and changeability are some of important factors of software products which we examined different aspects of SOA programming and explained how SOA can assist us reaching them.

Project division and business and UI separation are also necessary for large software development and service oriented architecture is one of the best options to supply them.

References

[1] W. T. Tsai, "Service-Oriented System Engineering: A New Paradigm," in *Proc. IEEE International Workshop Service-Oriented System Engineering*, 2005, pp. 3-6.

[2] W. T. Tsai, Z. Jin, P. Wang, and B. Wu, "Requirement Engineering in Service-Oriented System Engineering," in *Proc. IEEE International Conference on e-Business Engineering*, 2007, pp. 661-668.

[3] M. P. Papazoglou, "Service-oriented Computing: Concepts Characteristics and Directions," in *Proc. Fourth International Conference on Web Information Systems Engineering*, 2003, pp. 3-12.

[4] Ali Arsanjani. (October 2012)"Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA." [Online]. Available: http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IBM/I041109A.pdf

[5] S. E. Fawcett, G. M. Magnan, and M. W. McCarter, "Benefits, barriers, and bridges to effective supply chain management," *Supply Chain Management: An International Journal*, vol. 13, no. 1, pp. 35–48, 2008.

[6] H. Ernst, W. D. Hoyer, M. Krafft, and K. Krieger, "Customer relationship management and company performance—the mediating role of new product performance," *Journal of the Academy of Marketing Science*, vol. 39, pp. 290-306, 2011.

[7] M. N. Huhns and M. P. Singh, "Service-oriented computing: key concepts and principles, Internet Computing," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75–81, 2005.

Atefeh Khosravi, 1987M.Sc student of software engineering in Islamic Azad University-Tehran Northern Branch (Tehran/Iran). Received her B.Sc in software engineering from Islamic Azad University-Tehran Northern Branch (Tehran/Iran). Currently she is software analyzer and developer in Tosan LTD in Tehran, developing CoreBanking systems. She is interested in requirement engineering, business process analysis and service oriented computing.

Nasser Modiri, 1962, Received his M.Sc and PhD in Electronics engineering from the University of Southampton (UK) and the University of Sussex (UK). He is currently, Assistant Professor of Department of Computer Engineering Islamic Azad University (Zanjan/Iran). Research interests include Network Operation Centres, Framework for Securing Networks and virtual organizations.