A Survey of Service Oriented Architecture Systems Maintenance Approaches

Hamza Naji[1] Mohammad Mikki[2]

[1]Department of Computer Engineering, Islamic University, Gaza, Palestine
[2]Department of Computer Engineering, Islamic University, Gaza, Palestine

ABSTRACT

Maintenance of Service Oriented Architectures (SOA) plays an important role in guaranteeing and previews a successful deployment in any enterprise. The SOA "development and maintenance" process demands, must apply the traditional system evolution and maintenance rules. The conditions in SOA are different from the traditional software developing and maintenance, so we present in this survey paper the roles of SOA system developers and the different approaches for SOA maintenance systems as a problem's solution of using the traditional approaches for object oriented system.

KEY WORDS:

SOA, Service, Maintenance, Development, Oriented, Architecture.

1. INTRODUCTION

The small software projects such as calculator program or something else can develop just by coding it from the start to the end. But when we talk about big software projects we need the software life-cycle which comprises different phases that are followed in order to outsource a complete software product. The complete software life-cycle we talked about starts from requirements then we do the analysis and design phase finally we conclude with testing for the whole system. The remaining stage, which comes after the delivery is the maintenance. One type of the software architectures is the service oriented architecture (SOA); this type provides the ability to call the service instead of objects. Services are the basic architectural elements of SOA, in addition to reusable components that represent business or tasks, such as customer lookup, credit card validation, weather lookup, or line-of sight calculation[, ,]. Reusability could be a key part of this definition because it's what allows the creation of recent business and operational processes supported from these services. The maintenance is an important phase in service oriented architecture (SOA) and it demands rethinking of the traditional system evolution and maintenance roles. In this survey paper we examine the traditional use of the fundamental design principles in SOA maintenance scenarios after examining the definition of SOA and its elements in order. We analyze how the nature of maintenance differentiates in SOA with respect to the traditional software. We focus on the approaches and techniques that address the maintenance issues in SOA. The rest of the paper is organized as follows: Section 2 establishes the SOA definition and developers roles. Section 3 shows the SOA maintenance issues and Literature review. Finally, we tend to conclude the paper in Section 4.

2. SERVICE ORIENTED ARCHITECTURE (SOA)

The SOA system consists of three elements: services (the basic of the SOA system), applications (that import services and use it), and an SOA infrastructure (put the roles about how the applications and services are communicate after connecting) as shown in the SOA process in Figure number 1 below.

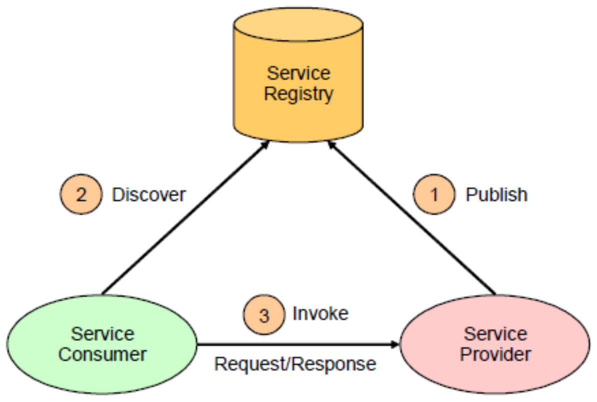


Figure 1: SOA components and processes

Figure 1 as shown presents the SOA components and progress of the operation in the whole system that built by services.

2.1 SOA ACTIVITIES

There are some SOA's activates that we can talk about in this subsection as the following:

2.1.1 STOCKHOLDERS

The evaluation process should show all members of the architecture to express activities of each of them and see how their concerns are addressed, those members, we called them the stakeholders. When the dependability of the stakeholders of SOA decreases the risk of overlooking important architectural concerns. One of the challenges of eliciting (QOS) needs for a system is that it perhaps not to be possible to know all the stakeholders. This is right for SOA systems because it consist of public services and the stakeholders search for services as their needs. We will present some common roles below for the traditional systems architecture and some special roles for SOA . The specific stakeholders chosen for an evaluation will depend on the needs of the organization. Also, we present the following stakeholders who should be invited to subscribe in the system evaluation architecture:

2.1.1.1 SYSTEM INSTRUCTION

1. **Software Architects**. The activities are including experimenting and deciding between several architectural techniques, providing the interface, and architecture validity and

verification against the functional and nonfunctional requirements quality attribute requirements (QAR). The developers of the architecture produce the documentation files that construct the architectural view for different stakeholders, documenting the risks and tradeoffs of the architectural design as well as the rationales for design choices. Architects additionally make sure that the implementation conforms to the design.
2. **Developers**. Their main activities contained implementing the architectural elements of the system consistent with the design specification, giving experience throughout detailed
3. design processes, and conducting experiments or making prototypes to validate an architectural approach.
4. **Service Usage Regulators**. Their main activities include making rules for service utilization, like give the specification that services should adapt to certain standards, and probably putting constraints on the services which will be used.
5. **Testers**. Their main activities include the plan test for the systems, to excite all of the planned tests, then save the results of all planned tests, and reporting faults.
6. **Integrators**. Their main activities are to bind that the architecture and implementation conform to open and standards are fully accepted, also to consider architectural approaches that simplify service integration, upgrades, and replacements.
7. **Maintenance Developers**. This is the main aim of our paper and we will present and explain their activities in the next section. Their main responsibilities include modifying the software system to correct defects and adapting the software once environmental changes occur (e.g., hardware or software system changes).
8. **Project Managers**. Their main activities include managing the development effort, creating the project plan, and tracing the progress of the project.
9. **Chief Information Officers (CIOs)**. The CIO activities are to trot out the architects, developers and business analysts to confirm that a solution can integrate well with existing systems, applications, and infrastructure.

2.1.1.2 SYSTEM CONSUMERS

1. **Chief Security Officers (CSOs)**. The CSO works with the architectural engineers, business analysts, and developers to confirm that each one data security policies are followed and ensured.
2. **Business Managers**. Their primary role is to make sure that the appliance supports the organization's business goals which the architects perceive all legal and restrictive implications.
3. **Business Analysts for Customers**. Their primary roles and responsibilities are to accumulate and transmit to developers the data of the business domain and functional requirements and quality attribute needs of the system.
4. **End Users**. Their main responsibilities include learning to work the system, making ready and coming into inputs, and deciphering the output from the system. They also generate system requirements.
5. **Developers of Service Users**. If the system provides services to external service user applications, the architects or developers who are liable for these external purchasers ought to even are invited. These external developers may provide input on application program interface (API) design and desired quality of service (e.g., availability).
6. **Maintenance Developers**. They are responsible for general maintenance duties with the delicate distinction that they'd most possible not be able to modify services and would

usually be forced to change alternative components of the system. The inability to modify services would be similar to buying off-the-shelf software.

2.1.1.3 INFRASTRUCTURE PROVIDERS

The infrastructure side contains a lot of providers as the following:

1. **System Administrators**.
   Their responsibilities include attaining a decent understanding of the system operation for troubleshooting issues that arise throughout and when deployment. They typically assume most duties associated with pc security in a corporation (i.e., repairs of firewall and intrusion detection systems, management of access rights, and applying patches to software system and operational systems).
2. **Network Administrator.**
   The network administrator responsible for the network infrastructure maintaining and troubleshooting issues with routers, switches, and computers on the network.
3. **Database Administrators.**
   They produce and maintain databases, making certain information integrity and consistent performance of the datacenters management systems.

2.2. NONFUNCTIONAL REQUIREMENTS OF MAINTENANCE PROCESS

The nonfunctional requirements or the quality of services is a common measurement for the SOA which we can judge of the end service by the nonfunctional requirements. The SOA nonfunctional requirement is some requirements in the specification of the system that includes the ability to use services as description documentation provided by the infrastructure providers we talked about above. Quality of service's can includes these requirements such as (Reliability, Availability, Usability, Security, Performance, Scalability, Extensibility, Adaptability, and Test ability). When the previous requirements are taken on consideration in the maintenance process of SOA Systems. According to the above explanation of quality of services we can now present the SOA maintenance developers as follows. The SOA components developers can do some tasks to prepare each of them to work in the system correctly. We will identify each developer and the particular tasks of them as follows:

2.2.1 INFRASTRUCTURE DEVELOPERS

Focus on providing a stable infrastructure that has standards, infrastructure services, and development tools. The infrastructure supports the protocol and data formats of the service's current and potential clients. Tasks for infrastructure developers include:

- Providing standards to implement as part of SOA infrastructure.
- Identifying discovery, communication, and security services.
- Identifying and developing obligated techniques to satisfy the most important set of potential service users.
- Providing tools for application and service developers.
- Documenting and supporting the infrastructure.

2.2.2 APPLICATION DEVELOPERS

Focus on the invention, composition, and invocation of services, either statically at design time or dynamically at run time. Key tasks for application developers are:

- Give an understanding to the concept of SOA infrastructure.
- Make a discovery process for services that incorporated into applications.
- Reviewing service description documentation.
- Get the appendix to invoke the identified services in applications, including any data conversions, error handling and availability handling.
- Give a testing process for the whole system after integrate the service in the context of the application being developed.

2.2.3 SERVICE PROVIDERS

Focus on the outline and roughness of services, in order that applications will simply find and use them with acceptable Quality of Service. Tasks include:

- Understanding requirements of potential service users.
- Understanding SOA infrastructure.
- Evolving code that receives the service request interprets it into calls into new or existing systems and produces a response.
- Describing and publishing the service.
- Developing service initialization code and operational procedures.

After we define the SOA activities and components and also define the maintenance process for both the traditional and SOA architectures now we need to present a quick review of the literatures to gather some of the approaches used in SOA based systems maintenance and development in the literature review section.

3. SOA MAINTENANCE ISSUES AND LITERATURE REVIEW

In this section we will review the previous approaches of the SOA maintenance and give the main ideas for each one. Then we will present the most efficient on of them which do well with our topic.

The software projects life-cycle start from identifying the idea of the product and then end with the end of the shelf life of the product. In fact there is no real shelf life for software but it begins to suffer towards the substantial development of the out of its environment by the high requirements needs. The architecture of the software must present in models, and these models contain the life-cycle of any product through abstract descriptions . The final phase in the life-cycle model is the maintenance and it's what we talking about in this paper. All phases preceding the maintenance conceded as a per- delivered phases i.e. the developing process of the software but the maintenance considered as a post delivered phase i.e. the phase generate after the product been delivered to the costumer. So, the software product is never been delivered as a complete version because of the requests of changes and violated requirements as the [Lehman Law] rules. Such requests for changing in requirements originate from the users of the software system, and may have a form of bug reports or may be requests for additional functionalities . The notion of the maintenance is not confined in fixing bug but it consists of the improvement of some of the aforementioned nonfunctional requirements or quality of services. It is a definition that closes to the IEEE Standard for software maintenance: Software Maintenance is the process of modifying a software system or component after delivery to correct faults, improve performances or other attributes, or adapt to a changed environment.

3.1 SOA MAINTENANCE IMPORTANCE

We can say that services have a service contract with an interface and the evolution aspect seems to be hidden from the SOA applications users. But in real the providers can change the functional attribute of the one service (without concern about who will use it) as the requirements without exposing these changes in their interfaces. Changes in SOA systems may be triggered not only by the service providers but also by the service consumers themselves, since they may desire arrangement with another competitor service meeting their new requirements or having better performance. This phenomenon is called independent service evolution .

3.1.1 RELATED ISSUES

When we talk about traditional software the case is somewhat easy but when we start re-developing the SOA with the concerned of the all QOS or nonfunctional requirements as much as possible by its providers without knowing the end user or which applications will use it in the future.

In this context we need to discuss some issues as follows:

The first issue: After developing the previous changes how we can estimate the influence on these changes in the whole SOA system according to the system functional and nonfunctional requirements because it is the importance factor in evolving the software systems . By return to service -the base element in our paper- when we make the developing process we have to take into consideration that the evolving service returns its interface and nonfunctional requirements or dependencies and that mostly conceder to be in complete information about the analysis of the service. The problem here is how to make the evolution in service that provide or deal just by interface and what is the influence of the whole system after the final integration with the developed service. The first approach in this context is **[Basuet. et.al.[13]]** Give a technique to deal with dynamic dependencies between services. Building of one dependency is related of the other identified dependencies between two messages by taking into consideration the appearing of services to other applications. This experiment was applied on HP business data, SOA based system consist of several services. **[Bertolino et al. [14]]** Give a model depends on black-box approach effect to specify the quality attributes including business requirements just taking advantage of service interface only. That can be happen by the invocations of its operations. A deferent system perturbation had been used by **[Romano, et al. [15]]** to explain the active dependencies. The use the previous technique to monitor the service work in the system. An operational dependency graph for a specific combo of system and workload was created by the active dependency approach while requiring very few details of the internal implementation of the system**. [Ryu et al.[16]]** Give a technique to deal with dependencies issues between services by something called completed conversations. The approach analyzes the strategies of combining the service nonfunctional changes and the SOA system. These conversations can produces by the system check of its executions by decision tree model. This approach will determine the business protocol dependencies between the system and also the developing service dividing them into forward and backward dependencies. **[Novonty et al.[17]]** Give a technique to deal with dependencies issues between services in a web application. Based on proposes a dependency, and the deep analysis for the text feature of hyperlink, a regular expression-based linkage information extraction method is presented. Other techniques are based on the formal or algorithmic approaches. For more practicability this approach uses mathematical foundations to record the behavior of the SOA to achieve the dependency handling purposes**. [Alda [18]]** is one of these approaches based on the previous technique which have a way for service handling dependences purposes. She distributes the approach into two steps. The first one is the ability of the costumer to use group of maintained services relies on public service produced by provider. That's mean the applying of the generalization of one service to contain a list on inheritances services. **[Liu, Ma and Zhao [19]]** is another type of the previous technique which presents the a conversations dependency between business processes to enhance the evolution with dynamic dependencies of the SOA based systems this approach is used to define the order of activities in the process and by this we can produce the conversation dependency. **The second issue** we want to talk about is the comprehension of the service which asked by "how" question. The question in this case is how we can identify the behavior of the service when it evolved.

Now we need to collect the approaches that help the maintenance developers to understand the evolved service operation and behavior for example the functional and nonfunctional requirements. We knew that the service can just implemented just by its interface and that make the comprehension process very hard because we can't access the data required for this task. Now we want to present some approaches that work well in the previous issue.**[Bertolino et al. 2009 [20]]** is an example of this issue which explained above in the first issue.

The third issue is to provide an approach that helps the maintenance developers to finish the functional and nonfunctional development attributes of the service and end these process with the testing of those attributes and making sure if the evolve service performs previous attribute requirements? Or not. **[Bertolino et al. 2009]** also provide an approach in this context to test the functional requirement attribute just based on its interface. The system's maintainer sends to the intermediate provider some conversations between the system and the service, which have been recorded in logs. Following, the intermediate provider acquires coverage data from the service provider, which show in what extent the conversations cover the functionality of the service. The coverage data can help SOA system's maintainer to: a. produce further test cases; b. become aware of when an adequacy criterion of its test cases is reached; b. update its test cases by adding tests which cover untested behavior; c. update its test cases by dropping tests that are exercising the same case; d. update its test cases by collecting coverage data on successive versions of services.

CONCLUSION

In this paper explain the notion of the SOA based systems software, the successes software factors, the components of it and the rules of each of the component developers. Then we defined the functional and nonfunctional requirements attributes and the importance of them in the maintenance process. Finally we present the importance of maintenance process in the SOA based systems and give some approaches in three issues in maintenance (the analysis influencing in the whole system, the understanding of services attributes, and the testing of services.) and explained each other. Finally the SOA maintenance topic still need other efforts to enhance the services maintenance process and it still a big space for researchers to support this area of research with new effective and creative approaches.

REFRENCES

[1] Grace Lewis, 2010“ Getting Started with Service Oriented Architecture (SOA)Terminology “ http://www.sei.cmu.edu/training/p81.cfm .

[2] Bromberg, Y.D., Issarny, V.(2005): INDISS: interoperable discovery system for networked services. In: Middleware 2005, ACM/IFIP/USENIX, 6th International Middleware Conference, Grenoble, France, November 28 - December 2, 2005, Proceedings. 164–183.

[3] Pistore, M., Traverso,(2007) P.: Assumption-based composition and monitoring of web services. In: Test and Analysis of web Services. Springer. 307–335.

[4] Phil Bianco, Rick Kotermanski,, Paulo Merson (2007). Evaluating a Service - Oriented Architecture , Software Engineering Institute .

[5] O’Brien, Liam; Bass, Len; & Merson, Paulo. (2005) "Quality Attributes and Service-Oriented Architectures". Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[6] Xiao, H., Guo, J., and Zou, Y. 2007. Supporting change impact analysis for service oriented business applications. In Proceedings of the 1st ACM International Workshop on Systems Development in SOA Environments (SDSOA).

[7] John McDermid, 1992, "Software Engineer's Reference Book", Butterworth-Heinemann,(November 2,3),P 170.

[8] Lehman, M. M. 1996. Laws of Software Evolution Revisited. In Proceedings of the 5th European Workshop on Software Process Technology (EWSPT).

[9] Bennett, K. H. and Rajlich, V. 2000. Software maintenance and evolution: a roadmap. In Proceedings of the 22th ACM International Conference on Software Engineering (ICSE).

[10] IEEE. 1998. IEEE Standard for Software Maintenance. Tech. rep., IEEE. International Standards Organization. 1998. Software Engineering-Software Maintenance.

[11] Canfora, G., Penta, M. D. 2006. SOA: Testing and self-checking. In Proceedings of the 1st International Workshop on Web Services-Modeling and Testing (WS-MaTE).

[12] de Souza, C.R.B 2005. On the Relationship between Software Dependencies and Coordination: Field Studies and Tool Support. Ph.D. dissertation, Donald Bren School of Information and Computer Sciences, University of California, Irvine.

[13] S. Basu, F. Casati, F. Daniel, 2008 “Toward web service dependency discovery for SOA management,” In: Proceedings of IEEE International Conference on Services Computing.

[14] Bertolino, A., Inverardi, P., Pelliccione, P., and Tivoli, M. 2009. Automatic synthesis of behavior protocols for compostable web-services. In Proceedings of the 7th joint meeting of the European Software Engineering Conference and the 17th ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[15] D. Romano, M. Pinzger, E. Bouwers, 2011 “Extracting dynamic dependencies between web services using vector clocks,” In: Proceeding of IEEE International Conference on Service-Oriented Computing and Applications.

[16] Ryu, S. H., Casati, F., Skogsrud, H., Benatallah, B., and Saint-Paul, R. 2008. Supporting the dynamic evolution of Web service protocols in service-oriented architectures. ACM Transactions on the Web.

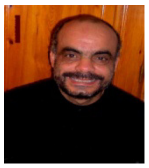
[17] P. Novotny, A. L. Wolf, B. J.Ko, S Lee, 2013 “Discovering service dependencies in mobile ad hoc networks”, Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management .

[18] S. Alda, 2005 "Peer group – based dependency management peer-to-peer architectures," Proceedings of the 2005 International Conference on Databases, information systems, and peer-to-peer computing.

[19] M. Liu, D. Ma, and Y. Zhao, 2009 "An approach to identifying conversation dependency in service oriented system during dynamic evolution," In: Proceedings of the 2009 ACM symposium on Applied Computing.

[20] Bertolino, A., Inverardi, P., Pelliccione, P., and Tivoli, M. 2009. Automatic synthesis of behavior protocols for compostable web-services. In Proceedings of the 7th joint meeting of the European Software Engineering Conference and the 17th ACM SIGSOFT International Symposium on Foundations of Software Engineering.

Authors



Mohammad A. Mikki: is a professor of parallel and distributed computing in the computer engineering Department at IUG with about twenty years of research, teaching, and consulting experience in various computer engineering disciplines. Dr. Mikki was the first chairman of the electrical and computer engineering (ECE) department at IUG in the academic year of 1995-1996. He taught both graduate and undergraduate courses at the ECE department at IUG. In addition he taught several undergraduate courses at the College of Science and Technology, College of Education (currently Al-Aqsa University), Al-Quds Open University, and University of Palestine. He was a visiting Professor at the Department of Electrical and Computer Engineering at University of Arizona in Tucson, Arizona (USA) during the academic year of 1999-2000. He was granted DAAD Study Visit Scholarship to Paderborn University in Paderborn in Germany from July 2002 to August 2002 from DAAD (German Academic Exchange Service). Dr. Mikki published about thirty publications in both international journals and conferences.

Hamza Naji: has graduated in 2013 with B.Sc. in Software Engineering from University of Palestine Gaza, then start studding M.Sc. Of Computer Engineering in the Islamic University of Gaza in 2014/2015. His research interests in the field of "Data Mining" and "Software Engineering".