ISSN:0975-9646

A Survey Report on Requirement Traceability of Service Oriented Architecture

Saini Subhadarshini, Abhishek Ray

School of Computer Engineering
KIIT University, Bhubaneswar, India

Abstract-Requirements of the stakeholders keeps on changing during the lifecycle of a software product development which is almost inevitable. Requirement traceability provides measures to identify the changes that happen during various phases of the software lifecycle, like during requirements analysis, coding, testing and the delivery of the product. In this paper our aim is to study the current use of requirement traceability in the context of service oriented architecture (SOA) and to make a systematic review of various traceability frameworks already available for SOA. Service oriented architecture in recent years have been very widely adapted as it bridges the gap between computer science and the business world. The distributed nature of SOA combines many services to make a system, so it is usually difficult to track the changes of the different service artifacts at the development lifecycle.

Keywords: requirement traceability, service oriented architecture, change analysis, service artifacts, system development lifecycle.

I. Introduction

The IEEE[1] define traceability as : " The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor–successor or master-subordinate relationship to one another, for example, the degree to which the requirements and design of given software component match". In other words requirement traceability maps the relationship between various components of a software system like requirements, design, code implementation and delivery of a system. Traceability is mainly accepted in software systems due to following factors: (i) that traceability relates each requirement to its design decisions, (ii) that it gives guidelines to the designers about the current status of the development, to trace and track the errors and to reconsider new designs decisions on the making, (iii) to remove the communication gap between the developer and the stakeholder and (iv) to validate the customer about the products maps to requirements.

SOA: Service-oriented Architecture is a software design pattern based on discrete pieces of software called services, providing application functionality to other applications. The services are combined to provide a complete functionality to a software application. The major characteristics [13] which makes SOA widely accepted are that they are (i) loosely coupled i.e. dependency between services is less, (ii) contractual i.e. services adhered to the services agreement, (iii) autonomous and discoverable i.e. services are self described so they can be easily found and accessed , (iv) the services are reusable, (v) composable i.e. they facilitate the assembly of composite services.

Requirement traceability for SOA: Requirement traceability for a software system is very much different from traceability of a service oriented architecture reason being (i) SOA is build comprising number of services.(ii) A SOA is usually built and maintained over more than a single system and finally, (iii) There may be many stakeholders responsible for a single SOA. We can say that SOA is a collective effort. So the relation between the business world and software needs to be maintained properly. There is always a probability of change in both the business and software. Again any change initiated in any of the both fields is a change for the SOA, as they are closely related to each other. So an efficient traceability mechanism is required to record these changes. At the instance of a business process change or a software change in the architecture, it must be immediately reflected in the system. Thus traceability for service-oriented architecture is a very challenging task. It is usually very difficult to develop traceability links between various services artifacts of a business. So there are many proposed frameworks which makes the task of traceability over SOA easy. In this study we will be reviewing some papers on traceability frameworks over SOA.

The rest of the papers is organized as follows : section-II gives the review of literature survey and a brief description of some papers of traceability over SOA. Section-III gives discussion of importance advantages and disadvantages of the traceability frameworks studied in section-II. Finally conclusion and further work are presented in section-IV.

II. Literature Review

Requirement traceability is gaining a very fast acceptance in the software development[2][3]. Many researchers have been made on requirement traceability. Its wide application on the development and maintenance of various types of software make it very useful. But so far there are a very few research on traceability applications on service oriented architecture. In this section we would be reviewing some major works on requirements traceability of SOA.

M.A Hirzalla et al.[9] proposes an Intellitrace framework. This intelligent framework acts across various layers of the SOA based system. This framework analyses the impact of changes of the service artifacts over the key performance indicators (KPIs) using traceability links.

The framework introduces service model which gives all the external and internal information of SOA based system. The Intellitracer tool used in the framework can be a standalone tool or it can be integrated with any commercially available tools. A trace is automatically initiated. The SOA change listener initiates the trace by analyzing different types of events with respect to the changes made. The framework overseas small changes, those that can be tolerated. Thus unnecessary actions can easily be avoided. The change of events are quickly analyzed from the dashboards provided by the tool and automates the process of making decision in different situations. This saves a lot of time and the quality of overall situation handling is improved..

S.Seedporf et al. [10] proposed a framework for semantic traceability for service oriented architecture. The key feature of the framework being that it can be individually adjusted for supporting traceability in any SOA environment. It uses web ontology language (OWL) which is a formal language to define a semantically rich SOA information model.

The framework is divided into two layers, the artifact layer and the knowledge layer. In the artifact layer all artifacts are maintained by the stakeholders using their own database or repository. No central repository is available in the framework. The knowledge layer contains an overall representation of the artifacts. This layer is automatically created extracting information from the artifacts layer, by the use of crawler, which is an extraction tool. Here the traceability links can be automatically generated or can be captured using extraction tool. It supports Ontobrowser Tool[6] to navigate in the knowledge base and ontology. We can say that STraS is a very flexible and light weight approach which can be incorporated in any SOA environment. It holds the stakeholders together by providing tools for notification and reporting.

P.Valaderas et al., 2008 [7] proposes a work for traceability of requirements of model-driven development of web applications. In this paper a model to model transformation has been defined that allows requirements traceability of navigational model from web application. Web engineering methods OOWS allows the abstraction of navigation model. Graph transformation techniques are used for mapping requirement which is done by a AGG tool. This mapping is automatically applied to the graph transformations for developing a tool support. A tool called TaskTracer is implemented in the framework. The navigation model derived is analyzed by the tool. And a HTML traceability report is generated that helps the stakeholder to analyze how requirements are supported by the conceptual elements. The information collected from the traceability report helps to check the validity and accuracy of the model to model transformation. This report helps to analyze how the requirements are supported in the navigation model, the effect of any change in requirement of the system and the modeling of the requirement. An ecommerce application is used to run on the task tracer framework to check the validity of the system.

N.Lungu et al., [8] in their paper focuses on the development of a complete requirement traceability solution system for housing the best traceability methodologies and solutions which are based on web service framework. The functionality of this system works on 6 modules within the limitations of traceability of web services. All the 6 modules of the system are independent web services and the communication between them is done through message passing mechanism. Thus a complete traceability is achieved with the successful collaboration of the web services. Here traceability is implemented as a traceability tree where a Root node is there to maintain the structure. A parent-child relation is developed which makes the user easy to manage the requirements. Testing is done on individual modules to check the validity of the system. Many commercially available tools are also well described in the paper.

III. Discussion

In paper [9,10] the authors have proposed some very extensive frameworks for traceability of service oriented architecture. There are meta models considered to convenient the process. The traceability done is either through the tool proposed or can be done through any commercially available tool. In paper [10] a framework is discussed to generate a traceability report of model-to-model transformation of web applications. Here the transformation are made by the AGG tool and trace links are automatically generated. In [8] the service oriented system is divided into some modules and each module acts as an individual and independent web service. The collaboration of these web services helps for the generation of traceability.

The major disadvantage that we came across all the papers studied is that the service artifacts considered for traceability does not comply the whole SOA. So probably the whole architecture is not available for traceability. Again the service modules given as input are not very well described and clear in the studied frameworks. In some papers the traceability tool used are commercially available tools that are available in the market. There is a flexibility in choosing the tools. Most of them are extraction tools rather than verification tools.

Some major advantages being that the characteristics of a SOA based system like flexibility, loose coupling, reusability, interoperability helps to apply traceability easily on it. Again traceability is vital for a SOA based system in order to record the frequent changes of the multiple services of the system.

IV. Conclusion

Thus from the above study it is clear that there are both advantages and disadvantages in applying requirements traceability to a service oriented system. The currently available traceability frameworks lack focus on incorporating the complete service-oriented architecture so it is difficult to generate a successful traceability of the system. Again due to the lack of automated support for generating and maintaining trace links between service artifacts makes it very time taking and tedious task.

So in future we would like to implement a traceability framework which deals both technically and systematically with the service oriented architecture for performing traceability. It should accept the service artifacts of any business format specified and analyze the changes in the system through traceability measures and then implement the changes back to the system, rectifying all the changes made.

REFERENCES

[1] IEEE standard glossary of Software Engineering Terminology, IEEE Std. 610.1990, 1990, p1.

[2] B.Ramesh and M.Jarke, "*Toward Reference Models for Requirements Traceability*" , IEEE Transactions on Software Engineering, vol.27, no1, pp- 58-93, 2001.

[3] V Lieno, *"Documenting Requirements Traceability Information: A Case Study",* Helsinki University of Technology , 2001

[4] M Hokkanen , *"Requirement Traceability",* Lapperenrate University of Technology, 2001.

[5] O.C.Z. Gotel and C.Finkelstein,*"An analysis of the Requirement Traceability Problem",* Proceedings of the First International Conference on Requirement Engineering, pp 94-101, April 1994.

[6] H.J.Happel and S.Seedof, "*Ontobrowser : A Semantic wiki for sharing knowledge about Software Architectures*", Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE), Boston, USA 2007, pp. 506-512.

[7] P Valderas and V Pelechano, *"Introducing Requirement Traceability support in Model-driven Development of Web Applications",* Journal Information and Software Technology 51, pg 749-768, April 2009.

[8] Ntheye Lungu and Fadzai Muvuti, "*Service-oriented Architecture for Software Traceability System".*

[9] M.A Hirizala, A Zisman and J Cland Huang, "*Using Traceabilty to Support SOA Impact Analysis"*, IEEE World Congress on Services, pg 145-151, 2011.

[10] S Seedorf,, K Nordheimer and S Krug*, "STraS a Framework for Semantic Traceability in Enterprise-wide SOA Lifecycle Management"*, pg 212-219, IEEE 2009.

[11] B.Ramesh and E.Edwards, "*Issues in the Development of Requirement Traceability model*", IEEE 1990

[12] http://en.wikipedia.org/wiki/Service-oriented_architecture