Rule-Based Multi-criteria Framework for SaaS Application Architecture Selection

Falak Nawaz, Ahmad Mohsin, Syda Fatima, Naeem Janjua

Rule-based multi-criteria framework for SaaS application architecture selection

Falak Nawaz[1], Ahmad Mohsin[1], Syda Fatima[1], Naeem Khalid Janjua[2]

[1] Department of Computer Science and Engineering, Air University, Multan, Pakistan
{fnn, ahmad, fatima}@aumc.edu.pk
2 School of Business, University of New South Wales, Canberra, Australia
n.janjua@unsw.edu.au

Abstract. Software-as-a-service (SaaS) is a very successful model for providing cloud-based services over the internet. However, due to the dynamic nature of SaaS services, it becomes very challenging to ensure provision of scalability, applying frequent maintenance and functionality updates to SaaS Services. SOAP and REST are the two mostly used software architectural styles for accessing and consuming SaaS services in cloud environment and each have its distinct advantages. Therefore, to address above mentioned challenges, it is critical to choose the suitable architectural style because the success of a SaaS is strongly coupled with its architecture style. Choosing the right software architecture for a system is a multi-criteria decision making problem and it takes into consideration the architectural style characteristics, non-functional requirements and working domain requirements. In this paper, we propose a rule-based multi-criteria decision support system (DSS) for a SaaS application architecture selection. Our proposed DSS uses weighted sum model (WSM) that take into account the architectural style characteristics, non-functional and domain specific requirements.

Keywords: cloud computing; software as a service; software architecture; SOA; REST; SOAP; architectural style; decision support system

1 Introduction

Cloud computing provides ubiquitous access to shared pool of configurable computing resources, software and data services hosted over the internet. The five important characteristics of cloud computing model are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [2]. The cloud computing service models are software-as-a-service (SaaS), infrastructure-as-a-service (IaaS), and platform-as-a-service (PaaS).

Cloud computing is built on top of virtualization that consists of compute, storage and network components in servers. Virtualization provides a solid foundation for cloud computing to expand. However, cloud computing service models are not mature enough. These service models require proven software design and architecture in order to provide scalable cloud computing platforms [4]. Therefore, selecting a suita-

ble architecture for SaaS service provisioning is a critical decision and it contributes towards success of the software. Software architecture describes the complete design of any system. This includes identifying system components, their interactions, granularity of communication needed for interactions, placement of system components in different subsystems, and interface protocols used for communication. Hence, selection of the correct software architecture is critical for the successful implementation of cloud computing platform.

In literature, different architectural styles have been used for delivery of service over the Internet. Service-oriented Architecture (SOA), in broader context, is an evolution of software architecture that provides software application functionality as a service to other applications or services. SOA provides software architectural guidelines for non-functional requirements such as reusability, modularity, extensibility and flexibility. Cloud computing platforms have not fully adopted SOA to make them more reusable, flexible and extensible [3]. To build scalable cloud computing platform, we need to leverage SOA to build reusable components, standardized interfaces, and extensible solution.

A Web service provides basic building block to develop a SOA based application. Web service is a method of communications between two computers over a network. There are two very famous types of web service architectural styles that are SOAP and REST. SOAP based implementations are also termed as Service oriented Architecture (SOA) while REST based implementations are known as Resource Oriented Architecture (ROA). We will explore these two architectural styles in next section.

Due to the complexity of cloud computing environment and the software systems running on it, success of the system strongly depends on their architecture. Software architecture has been a key element in software development process for last two decades [14]. Moreover, software architecture selection is a multi-criteria decision-making problem in which software architect try to analyze different goals and objectives for the system under consideration. Some architectural styles may have good effects on a particular problem domain but may not be suitable for another problem domain. Similarly, both REST and SOAP have their own quality attributes and limitations that may have good or bad effect(s) on different problem domains. Although comprehensive list of attributes have been listed for each style in different texts, but we cannot understand the extent to what advantages and disadvantages of quality and quantity attributes of architecture are considered [14]. Therefore, comparing capabilities, characteristics and benefits of software architecture is somehow difficult task. In this paper, a rule based decision support system (DSS) has been developed which tries to support the decision making process by considering different related criteria for web service architecture including quality attributes, domain requirements and architectural style characteristics.

In the next section, a comparison of REST and SOAP architectural styles is presented. In Section 3 we will describe the motivation for selection of architectural style in cloud-based environment. Section 4 will present factors and criteria for architectural styles, non-functional requirements and domain requirements for multi-criteria decision making. Section 5 will present proposed DSS for Web service architectural selection and finally Section 6 will conclude this paper.

2 REST vs. SOAP

A Web service provides software functionality online that is accessible through network endpoints. As discussed above, Web service can be technically distinguished between SOAP and RESTful Web services.

SOAP Web services use XML for message format and description of interfaces (WSDL). A SOAP-based architectural style is suitable for application domain with following characteristics [5].

- A formal contract is required between service provider and consumer. Web Services Description Language (WSDL) takes care of all the information required about the web service operations, parameters, types and message format. This is useful in situations where service composition is required in complex business process models.
- Architecture needs to handle complex non-functional requirements such as security, trust and coordination.
- Architecture needs to handle asynchronous invocation and processing of service requests.

RESTful web services are usually considered suitable for lightweight and ad hoc services. RESTful web services use existing W3C standards, e.g. HTTP. A REST-based architectural style is suitable for application domain with following characteristics.

- There is no formal description of service endpoints (e.g. no WSDL) between service producer and consumer. The service producers usually provide some additional toolkits describing how to invoke REST web services.
- Architecture needs to handle non-functional requirements such as scalability, network bandwidth, and integration.
- Architectures where web services are completely stateless just like HTTP requests on a website. For example, existing web applications can expose their functionality easily through CRUD operations using RESTful web services.

Many comparative studies have been carried out to identify strengths and weaknesses of SOAP and RESTful Web services with respect to architecture, technology, security, performance, and scalability. The conceptual, principal and technological level comparison of RESTful Web services and SOAP based “big” web services is performed in [6]. The author says, “On the principle level, two approaches have similar quantitative characteristics. On the conceptual level, less architectural decisions must be made when deciding for SOAP-based Web services. On the technology level, the same number of decisions must be made but fewer alternatives have to be considered when building RESTful Web services”. SOAP-based Web services produce considerable network traffic and high latency due to large message size [8]. But payload size in SOAP can be reduced using different compression techniques [7]. RESTful Web services have better performance than SOAP-based Web services in wired and wire-

less communication networks. The RESTful web services are lightweight, easy and self-descriptive with higher flexibility and lower overhead.

Another main difference between REST and SOAP Web services [9] is that SOAP is a tightly coupled design similar to RPC (Remote Procedure Call), and REST is a loosely coupled design similar to navigating Web links. As a study of two opposing standards based on SOAP and REST, SOAP has the advantage of tight coupling of operations, while REST has the advantage of scalability and lightweight access to its operations. REST has the disadvantage of managing the name space with a large number of objects, and a disadvantage of SOAP is the need for dedicated client ports for different types of notification. REST principles also played an important role in standardizing SOAP 1.2.

Internet of Things (IoTs) and Mobile devices has taken this discussion of selecting REST or SOAP style architecture to the next level. A comparative study of SOAP vs REST for provisioning Web services on Mobile phone is performed in [7]. A benchmarking evaluation of SOAP and RESTful Web services is performed in [10] for mobile devices. Benchmarking includes string concatenation and float number addition web services. The performance evaluation results show the advantages of using RESTful web services over conventional web services for mobile devices. Authors in [11] compared REST and SOAP Web services with respect to performance. They concluded that RESTful Web services are now emerging as an alternative to SOAP-based Web services and might be a more suitable choice in some cases.

The more recent work on SOAP and REST highlights that Web Service performance is becoming an important factor. This work "concluded that RESTful web service is a better alternative for SOAP based web services. SOAP based web services are produces considerable network traffic, high latency and the message size is also large this is not in the case of RESTful. The RESTful web services have better performance than SOAP based web services in wired and wireless communication network. The RESTful web services are lightweight, easy and self-descriptive with higher flexibility and lower overhead [12].

3 Motivation

Evolution of Cloud services, Internet of Things (IoTs) and Web mobile apps are shaping the future of internet. This will determine how smartphones and physical devices will interact with each other. In this paper, we have focused on why choosing the right architecture for cloud based software-as-a-service (SaaS) is important and what factors drive architecture selection.

Each architectural style chosen has trade-offs related to Non-functional requirements (NFRs) and Architectural characteristics (ACs). Domain requirements (DRs) are equally important while choosing the right architecture. Currently software architect give less preference to the domain requirements while choosing the architecture. Within each of above mentioned area, there are number factors that need to be taken care of while choosing the right architecture. But there is no such automated tool that makes this multi-criteria decision making easy. Very rare work has been done in this

domain. Existing systems focus on ACs and DRs only [14] and does not take NFRs into account. Our proposed DSS takes all three areas into consideration and uses a knowledge base that has the ability to update its knowledge and suggest suitable choices to the software architect.

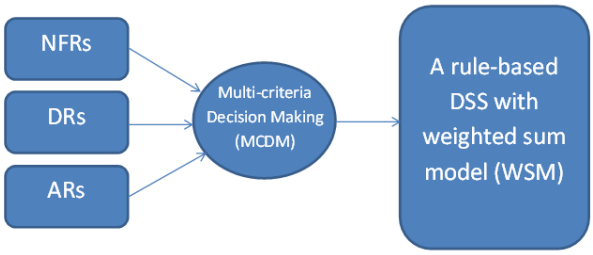

Fig. 1. Pictorial representation of the problem

Architecture must be chosen according to the nature of the applications, its domain requirements, non-functional requirements and required characteristics of architecture. Architecture style chosen by considering these requirements will be best suited for cloud based software-as-a-services (SaaS). Therefore a need for decision support systems in this domain is appreciated.

4 Multi-criteria decision maker for architecture selection

A multi-criteria decision maker will give suitable option to software architect for solving above mentioned problem. A decision support system (DSS) provides support for all phases of the decision making process. It should be easy to use meanwhile providing support for users at all levels to make decision [14]. We have used CLIPS as a Rule-Based DSS in which the knowledge component includes procedural and inferential rules. We found number of ACs and NFRs for various domains and tried to analyze cloud based SaaS web services. We then prepared a comparative analysis of these ACs and NFRs for both SOAP and REST implementations.

4.1 Domain Requirements (DRs)

We have selected some domain characteristics (or high level functional requirements) according to type of application. The selected cloud-based SaaS web services fall in the following domains: E-commerce, Logistics, Telecommunications and Health-care.

4.2 Architectural Characteristics (ACs)

We explored the architectural characteristics for SOAP and REST that important for selection of architecture. The characteristics that are considered in our DSS are given below in Table 1.

Architectural Style Characteris-	SOA	REST
Heterogeneity	✓	✓
Protocol layering	✓	✓
Loose coupling	✓	✓
Integration style	✓	✓
Resource identification	×	✓
URI design	×	✓
Resource interaction semantic	×	✓
Resource relationship	×	✓
Contract design	✓	✓
Data representation	✓	✓
Message exchange pattern	✓	✓
Traffic monitoring	×	×
Traffic determination	×	×
Traffic transformation	×	×
Service description	✓	✓
Service identification	✓	✓
Service composition	✓	✓

Table 1. Architectural Characteristics

4.3 Non-functional Requirements (NFRs)

There are a number of non-functional requirements and their sub-factors but we only selected security, reliability and performance as key NFRs for our DSS for architecture selection process. The reason for selecting these NFRs is that these are the most common NFRs required for applications domains mentioned above.

NFR	Sub-factors	SOAP	REST
Security	Encryption	✓	✓
	Integrity	✓	✓
	Authentication	✓	✓
	Authorization	✓	✓
	Non-repudiation	✓	✓
	Confidentiality	✓	✓
Reliability	Point-to-Point	✓	✓
	Ordered delivery of message	✓	✓
	Delivery status	✓	✓
	Elimination of duplicate message	✓	✓

	Resending message	✓	×
	Reliable delivery of message	✓	✓
Performance	Caching	×	✓
	Load balancing	×	✓
	Throughput	✓	✓
	Response time	✓	✓
	Latency	✓	✓
	Execution time	✓	✓

Table 2. Common NFRs required for applications domains

5 Working of proposed rule-based DSS

In order to select architecture style correctly and precisely, all existing information related to the application are considered. The proposed DSS uses characteristics of Web service architectural styles, characteristics of domain of application being developed and non-functional requirements. We assigned weightage to these characteristics and requirements against both REST and SOAP architectures and used this weighted criteria for inference in DSS. The complete design of proposed DSS is depicted in figure 2 in detail. The proposed DSS has five essential components that help in decision making process.

- Repository
- Tools
- Rule base
- Decision maker
- User interface

Responsibilities of each component and what they contribute to decision making process is given below.

5.1 Repository

We have three types of repositories which are DRs (Domain Requirements), NFRs (Non-Functional Requirements) and ACs (Architecture Characteristics). In DRs we have characteristics regarding requirements for different domains mentioned in the previous section. NFRs contain requirements provided by different quality attributes and also information regarding the number of sub-attributes of quality attributes provided by specific web service architectural style. ACs has information of all the characteristics of web service architectural styles SOAP and REST so that architect can select according to the nature of application being developed.

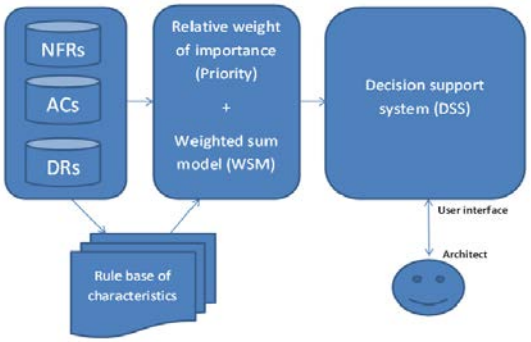

Fig. 2. Rule-based DSS for Web service architectural style selection

5.2 Tools

Domain requirements and architectural styles characteristics would be prioritized on the basis of number of characteristics selected.

When the necessary information is gathered and importance of levels of quality attributes are obtained, the DSS applies Weighted Sum Model (WSM) for NFRs and number of characteristics required for specific Web service architectural style and domain requirements would already be counted while gathering information according to the need of app being developed as shown in figure 2. WSM is the simplest MCDA (multi-criteria decision analysis/making) method for calculating a single result from a number of alternatives [15].

We can generalize an MCDA problem by assuming M alternatives and N decision criteria [16]. We further assume that all criteria/factors mentioned in Table 1 and Table 2 are our benefit criteria which means that higher the values, the better it is. A tick mark in the table is assumed to have a value of 1 and cross mark is assigned 0. Next suppose that w_j denotes the relative weight of importance (priority) of the criterion c_j and a_{ij} is the performance value of alternative A_i when it is evaluated in terms of criterion c_j. Then, the total (i.e., when all the criteria are considered simultaneously) importance of alternative A_i, denoted as $A_i^{\text{WSM-score}}$, is defined as follows:

$$A_i^{\text{WSM-Score}} = \sum_{j=1} w_j \, a_{ij}, \text{ for } i = 1, 2, \ldots, m$$

In this case, SOAP and REST are alternatives and quality attributes (such as security, reliability, performance) and architectural style characteristics are criteria. The rules that are extracted from architecture characteristics, Non-functional requirements, domain requirements and the priorities as defined by the user of the DSS will become inputs to the system.

5.3 Rule Base

In order to make decisions, we need some rules which indicate the interaction among DRs, NFRs and ACs with respect to the nature of Web services being developed. In other words, the rules determine which domain characteristics are required, which characteristics of architecture are required, what is the importance level of each quality attribute. These rules are kept in the rule base and are extracted from repository.

5.4 Decision maker

Decision maker provides a rule-based engine to allow a wide range of knowledge to be represented as heuristics or rule of thumb. These rules specify a set of actions to be performed for a given situation. This component is responsible of receiving information about the priorities ci of every domain requirements, architecture characteristics and non-functional requirements from software architect via user interface. These priorities ci are considered as inputs for the weighted sum model for NFRs and ACs, the decision maker determines a particular architectural style suitable for the cloud-based services being developed. This result is then displayed to the user through user interface. If there is more than one architectural style suitable for a particular domain, the system recommends all of them, leaving the final choice to the software architect. The system also has the learning capability of storing the final selected architecture by the architect in knowledge base and uses it in future for similar situations to recommend right architecture.

5.5 User interface

The user interface is responsible for receiving the information from user regarding domain requirements, architectural style characteristics, non-functional requirements and also the priorities for NFRs and ACs. The suggested architecture style(s) is/are represented by user interface as a suggestion to the architect by their priority.

6 Conclusion

REST and SOAP are the most used architectural styles for implementing cloud-based software-as-a-services (SaaS). Due to complexity, frequent maintenance and updates of the implemented services, it is important to choose the right architectural style based on NFRs, architectural style characteristics and domain requirements. However, choosing the right architectural style is a multi-criteria decision making problem that requires all the factors to be considered along with their priorities by the software architect. Our proposed system solves this problem by using the weighted sum model (WSM) and rule-based DSS. By defining ACs and NFRs criteria in the form of rules and using relative weight of importance (priority) to WSM, we get a suggested architectural style that is suitable for the given domain.

References

1. R.T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," doctoral dissertation, Dept. of Information and Computer Science, Univ. of California, Irvine, 2000.
2. Mell P, Grance T: The NIST definition of Cloud Computing. Gaithersburg, MD: NIST, Special Publication 800–145; 2011.
3. L. J. Zhang and Q. Zhou "CCOA: Cloud Computing Open Architecture," In the proceedings of IEEE International conference of Web Services, 2009.
4. A B. Zhang, L. Su, Y. Sun, M. Lu, "Research on Cloud Computing Technology Serving Space TT&C Applications". Lecture Notes in Electrical Engineering, Proceedings of the 27th Conference of Spacecraft TT&C Technology, Guangzhou, China 2015.
5. http://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html
6. C. Pautasso, O. Zimmeraman, and F. Leyman, "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision", WWW 2008, April 21–25, Beijing, China. 2008.
7. K. Wagh and R. Thool, "A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host", Journal of Information Engineering and Applications, 2012.
8. S. Mumbaikar, P. Padiya, "Web Services Based On SOAP and REST Principles", International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013.
9. M. Z. Muehlen, J. V. Nickerson, and K. D. Swenson, "Developing Web Services Choreography Standards - The Case of REST vs. SOAP," Elsevier B.V. 2004.
10. H. Hamad, M. Saad, and R. Abed "Performance Evaluation of RESTful Web Services for Mobile Devices" Computer Engineering Department, Islamic University of Gaza, Palestine International Arab Journal of e-Technology, Vol. 1, No. 3, January 2010.
11. F. Belqasmi, J. Singh, S. Y. B. Melhem, R. H. Glitho, "SOAP-Based vs. RESTful Web Services: A Case Study for Multimedia Conferencing", *IEEE Internet Computing*, vol.16, no. 4, pp. 54-63, July-Aug. 2012, doi:10.1109/MIC.2012.
12. S. Mumbaikar and P. Padiya, "Web Services Based On SOAP and REST Principles", International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013.
13. D. Guinard, "A Web of Things Application Architecture - Integrating the Real-world into the Web", PhD thesis No. 19891, ETH Zurich, Zurich, Switzerland, August 2011.
14. S. Moaven, J. Habibi, H. Ahmadi and A. kamandi, "A Decision Support System for Software Architecture-Style Selection", Sixth International Conference on Software Engineering Research, Management and Application, 2008.
15. E. Triantaphyllou "Multi-Criteria Decision Making: A Comparative Study", Dordrecht, The Netherlands: Kluwer Academic Publishers (now Springer). p. 320. ISBN 0-7923-6607-7. 2000.
16. F. Nawaz, K. Qadir, H. Farooq Ahmad, "SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish Subscribe Model". In the fourth International Conference on Semantics, Knowledge and Grid. IEEE, China 2008.
17. M. Svahnberg, "Supporting Software Architecture Evolution - Architecture Selection and Variability", Ph.D. Thesis, Blekinge Institute of Technology, 2003.
18. Janjua, Naeem Khalid, et al. "Digital health care ecosystem: SOA compliant HL7 based health care information interchange." Digital Ecosystems and Technologies, 2009. DEST'09. 3rd IEEE International Conference on. IEEE, 2009.