Journal of
Information Technology & Software Engineering

Mohsin et al., J Inform Tech Softw Eng 2016, 6:2
DOI: 10.4173/2165-7866.1000176

Research Article Open Access

An Automated Approach for Web Services Architectural Style Selection

Mohsin A*, Fatima S, Khan AU and Nawaz F

Department of Computer Science and Engineering, Air University Multan, Pakistan

Abstract

Selection of an appropriate architectural style is vital to the success a web service. The nature of architecture design and selection for service oriented computing applications is quite complex as compared to traditional software architecture. Web Services have complex and rigorous architectural styles with their own underlying architectural characteristics. Due to this, selection for accurate architectural style for web services development has become more complex decision to be made by architects. Architectural style selection is a multi-criteria decision and demands lots of experience in service oriented computing. There is a huge gap for automated selection of web services architectural styles. Decision support systems are good solution to simplify the selection process of a particular architectural style. Our research suggests an automated approach using DSS for selection of architectural styles while developing a web service to cater FRs & NFRs (Functional & Non Functional Requirements). Our proposed mechanism helps architects to select right web service architectural pattern according to domain, and non-functional requirements without compromising quality. In this paper a rule base DSS has been developed using CLIPS (C Language Integrated Production System) to support decision process in multi-criteria requirements. To select suitable web service, system takes architectural characteristics, domain requirements and software architect preferences for NFRs as input by applying rule base approach. Next Weighted Sum Model has been applied to prioritize quality attributes and domain requirements. Scores are calculated using multiple criterions to choose the final architecture style.

Keywords: Web services; Architectural styles; Rule based; DSS; Multi-criteria requirements; Quality attributes; Automated decisions

Introduction

Software architecture controls how system elements are recognised and assigned, how the elements interrelate to form a system, the amount of communication needed for interaction. Therefore, selection of the suitable architectural style(s) for use in construction of software is of importance. A good Architecture can create a difference between success and failure of web and mobile applications in SOC (Service Oriented Computing) domain.

Service Oriented computing has emerged as top choice for software developers, utilizing the integration of cloud computing and IOTs (Internet of Things). The basic components of SOC are services over distributed networks allowing various devices and software to exchange information. SOC, initially emerged from Service Oriented Architecture(s) an architectural style, it has now become a larger knowledge area consisting of other architectural styles. Because of operational constraints in different environments Quality factors are largely dependent on particular architectural style in use.

A typical Web service is an interface that defines a collection of operations that are network accessible through standardized XML messaging. A Web service is described using a standard, format XML or JSON notion, called its service description [1]. At present we have various architectural styles to choose from to develop a web service each with its own pros and cons.

Software architecture has been a key component in software development in past two decades. Therefore, choosing the correct architecture is a basic task in software engineering phases, concerning quality attributes of a web service. Software Architecture provides abstractions and defines relationships among those abstractions while Architectural Styles impact largely on performance, security, reliability and many others. When we talk about Service oriented computing applications, they are more complex and heterogeneous in nature with respect to different architectural styles. A few traits have been recorded for each style in distinctive writings, however we can't comprehend the degree to what benefits and drawbacks of quality and quality attributes of each architecture are considered [1]; thusly, contrasting capabilities, attributes and benefits of software architecture is a by one means or another difficult task. Apart from Quality requirements other set of requirements make it a tough decision for architects for selection of a particular style to develop a web service.

Figure 1 shows N- tier architectural framework of Web Services. In this figure basic components of a typical web service application are depicted by keeping in mind FRs and NFRs required to support any architectural style, typical web service architecture has been mapped into three layers.

At present when we are conducing this research there are multiple architectural styles available for developing a particular web service like SOAP, RESTFU etc. Each one of these style(s) allows system architects to develop web service to complete a specific functionality with similar and conflicting quality attributes. The general concern of users is performance, security, reliability and related quality attributes. Other most important factor is self-characteristics of architecture to be selected. So, software architect(s) have to consider the self-characteristics as well as domain requirements in addition to NFRs in order to select appropriate architecture according to need of the app being developed. This decision becomes critical to meet different and varying requirements.

At the moment there are various web services styles suitable for different types of web services to design. But there is a lack of work to make distinction which web service style is better for a particular set of requirements. Requirements are of various types i.e. Functional, Non Functional and Domain specific. There is not a single Architectural

***Corresponding author:** Mohsin A, Department of Computer Science and Engineering, Air University Multan, Pakistan, Tel: 92 300 0760708; E-mail: ahmad@aumc.edu.pk

Received March 12, 2016; **Accepted** April 24, 2016; **Published** April 30, 2016

Citation: Mohsin A, Fatima S, Khan AU, Nawaz F (2016) An Automated Approach for Web Services Architectural Style Selection. J Inform Tech Softw Eng 6: 176. doi:10.4173/2165-7866.1000176

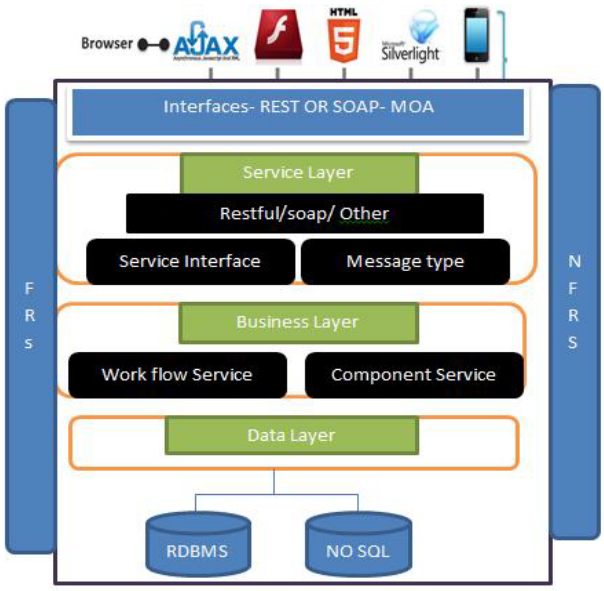

Figure 1: Web Service N- tier Architecture style.

style for a web service which can full fill are the criterion for a given problem set. Some Style are good in terms of Performance but lacks security and vice versa. Architects usually choose a web service style based on its performance in pervious projects, word of mouth or its ease in development. There is no proper framework or any automated system which can recommend for usage of a particular web service style.

Service oriented computing has transformed modern web and cloud computing paradigms. Though there are different architectural styles for developing a web service but here we have considered three different types of web-service architectures which are widely used in industry today. Service Oriented Architecture (SOA), Resource Oriented Architecture (ROA) and Message Oriented Architecture (MOA). These will be explored in later section in detail.

After research we have considered three types of requirements these are Domain Requirements, NFRs and self-characteristics of a particular architectural style(s) in whole decision making process.

For developers and architects there are many architectural choices to choose for a web service. At the moment various architectual patterns exist carrying their own pros and cons for a specific type of a web service development in a particular domain. Architectural style selection is based on various aspects of the system under investigation. There are number of reasons of poor quality but we have considered two that are negligence of NFR (Non-functional requirements) and nature of applications being developed. Architectures have a number of characteristics that must be considered but manual decision and prioir expertise are not enough to make a correct choice.

In this paper, a Rule based Decision Support System (DSS) has been developed which attempts to help the decision making process by keeping in mind different related criteria for web service architecture including quality attributes weightage according to web and mobile app being developed, domain requirements and architectural style characteristics. And base on these criteria suggesting suitable solutions.

Section II presents Literature Review of the related work and Research Challenges. In Section III we shall present proposed DSS for web service architectural selection; Section IV will show Selecting architecture using proposed DSS. In Section V we will see impact on decision making process, Section VI discusses the advantages and Section VII & VIII will cover future work and conclusions respectively.

Literature Review

In this section we shall review service oriented computing, web service, web service architectural styles, approaches used for selection of architecture in past.

SOC architectural styles

Service-oriented computing is an emerging cross-disciplinary paradigm for distributed computing, which is changing the way software applications are designed, delivered and consumed. At the heart of service-oriented computing are services that provide autonomous, platform-independent, computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols to build networks of collaborating applications distributed within and across organizational boundaries. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. The general description of commonly used architectural styles is as follows:

MOA (Message-Oriented Architecture): MOM (Message-Oriented Middleware) or MOA is an alternative to the RPC (Remote Procedure Call) distribution mechanism. This mechanism called Message-Oriented Middleware or MOM provides a clean method of communication between disparate software entities. MOM is one of the foundation stone that distributed enterprise systems are built upon. MOM can be defined as any middleware infrastructure that provides messaging capabilities. A client of a MOM system can send messages to, and receive messages from, other clients of the messaging system. Each client connects to one or more servers that act as an intermediary in the sending and receiving of messages. MOM uses a model with a peer-to-peer relationship between individual clients; in this model, each peer can send and receive messages to and from other client peers. MOM platforms allow flexible cohesive systems to be created; a cohesive system is one that allows changes in one part of a system to occur without the need for changes in other parts of the system [2].

SOA (Service-Oriented Architecture): The service-oriented Architecture (SOA) uses services to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications. Services are autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways. They perform functions that range from answering simple requests to executing sophisticated business processesrequiring peer-to-peer relationships among multiple layers of service consumers and providers. Any piece of code and any application component deployed on a system can be reused and transformed into a network-available service. Services reflect a "service-oriented" approach to programming that is based on the idea of composing applications by discovering and invoking network-available services to accomplish some task. This approach is independent of specific programming languages or operating systems. It lets organizations expose their core competencies programmatically over the Internet or various networks.

ROA (Resource-oriented Architecture): Resource Oriented Architecture (ROA) or REST Oriented Architecture are used interchangeably and ROA (REST Oriented Architecture) is just a fancy name for a SOA (Service Based Architecture) using REST services. REST was proposed by Roy Fielding. REST is architecture for developing Web services. REST attempts to mimic architectures that

use HTTP or similar protocols, by constraining the interface to a set of well-known standard operations (i.e., GET, PUT, POST, and DELETE for HTTP). Here, the focus is on interacting with statefull resources, rather than messages or operations. "REST architecture is designed to show how existing HTTP is enough to build a Web service and to show its scalability". It avoids the complexity and processing overhead of the Web services protocols by using bare http. One important REST concept is a resource, which is a piece of information that has a unique identifier (e.g., a uniform resource identifier (URI)). REST web services architectural style reduces the complexity of transforming data from XML or JSON between sender and receiver.

SOAP vs. RESTfulvs. MOA: Though there is a division among the advocates of two prominent styles in web services but recently a trend has been seen which shows architects and developers favouring RESTful as it is much easy to implement. In a research case study for developing multimedia conference applications both styles were used and results show that RESTful proved itself much better in performance [3]. In another work where performance analysis was done for both styles on different platforms in cloud, RESTful outclassed SOAP style [4]. The new versions of SOAP make it easy to use. One advantage of SOAP is that it uses generic transport protocols while RESTful only uses http/https. REST is best suited in less complex scenario while SOAP is more suitable for complex systems [5]. Message oriented Model has some commonalties with SOAP but differs in architecture and internal complexities.

Architecture selection approaches: The use of Computational intelligence for decision support system in order to solve different problems is much better as opposed manual decisions [6]. A decision support system automates the process for decision making in any domain. There are various approaches for its usage, here we mention few related to our work.

Wang and Yang proposed a selection method for people who lack expertise and experience to select appropriate architecture style for their software systems [7]. The authors collected and categorized a number of common architecture styles, and used Quality Attributes as a criterion to evaluate all those architecture styles. Moreover, they provided a systematic selection process powered by AnalyticHierarchy Process (AHP). It is a widely used theory andprovides a measurement through pair wise comparisons and relies on the judgments of experts to derive priority scales. This method just considers the quality attributes but does not cater the domain and architectural requirements.

Babu et al. [8] presented a method called SSAS (Selection of Software Architecture Styles). It uses analyticnetwork process (ANP) to determine the degree ofinterdependence relationship among the alternatives (architecture styles) and criteria (Requirements). It provides a way of collecting expert group opinion along with stakeholders interests (e.g. reliability, performance) [8]. It should be noted that, the traditional AHP is applied to the problem without considering interdependence property among the criteria.

Moaven et al. [9] explained "A Decision Support System for Software Architecture-Style Selection" presented DSS which makes use of fuzzy logic to represent concepts of quality attributes more precisely and efficiently while considering interaction among them. They constructed a DSS based on knowledge-base which has the ability of updating its knowledge and provides the system architect with suitable choices to select among them. With respect to knowledge base and exploiting expertise of people that work in this domain (e.g. expert architect) some rules have been extracted that can help to surfing the style repository and offering a style or combination of some styles [10]. This DSS considered NFRs but this DSS not considered the characteristics of architecture to be asked by architect as input for making decision.

Theoretical framework for using a decision maker for cloud based web services architecture selection via rule based engine was presented by Falak et al. [2]. This work only presented framework for choosing an appropriate style among ROA and SOAP. Moreover practical implication of this framework was not provided so impact on the selection process could not be predicted [2].

Architecture selection techniques

Further we can categorize Architecture selection techniques based on manual selection and automated selection. Both techniques help Architects for selecting appropriate architecture. Being a web service a complex entity to be built with diverse requirements, it becomes difficult for system architects to select the one which is most suitable for needs. On the other hand automated selection facilitates software architect to select closely related as per requirements with very short time investment. One can say that manual process for architecture selection is economical but it depends how much domain experience the decision maker has.

The research challenge

Research Challenge: Web Service Architecture Selection in Multicriteria Requirements.

The single most difficult challenge in software development in modern era is the selection of appropriate architectural style and this decision becomes even complex and difficult when it comes to select web services styles. Multiple factors come into play in selection of architecture style for Service Oriented applications. So core Research challenges are:

- Selection of correct architecture style for web services without compromising quality attributes.
- Existence of multiple web services architectural styles.
- Multi-criteria Requirements
- Complex Decision making process
- Diverse Characteristics of different styles

Proposed DSS for Web Service Architectural Selection

Our proposed solution tries to automate the web service architecture selection process by focusing on quality attributes and functional requirements. There are some common features among web service styles but target applications always do require varying quality attributes. So as a solution we have developed a rule based decision support system in CLIPS (C language Integrated Production Shell) to automate the. For this purpose three architecture styles SOA, ROA and MOA are selected and NFRS security, reliability and performance are considered. NFR preferences are taken first along with general characteristics for architectures under question. Weighted sum model is applied with selected characteristics prioritization. This process ends with suitable web service style suggestion based on requirements. The complete process of proposed DSS is depicted in Figure 2.

DSS and its types

Due to the extremely high attention to the computer-based information systems, making use of Decision Support Systems to support and improve decision making has become of importance.

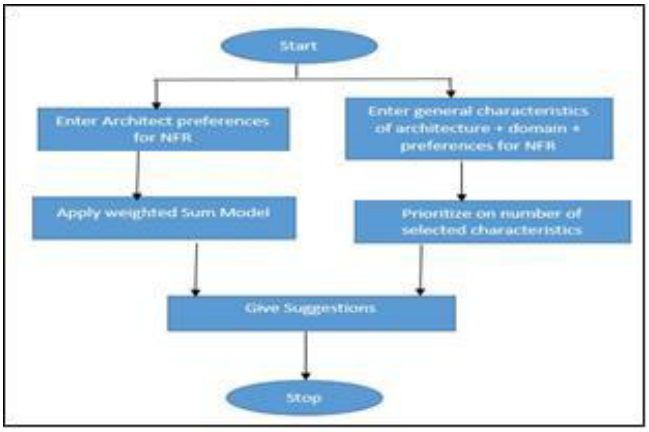

Figure 2: proposed DSS work flow.

Decision making problem is the process of finding the best option from all of the feasible alternatives. DSS is a computer-based system which supports a decision, in any way. DSS will essentially solve or give options for solving a given problem. A DSS provides support for all phases of the (semi-structured and unstructured) decision making process and a variety of decision making processes. It should be easy to use meanwhile providing support for users at all levels to make decision [11]. Decision support systems can be classified in several ways. One well-known classification is to put them into six frameworks [11]:

1) text-oriented which emphasizes on creating, revising and reviewing documents; 2) database-oriented in which the database organization is of importance and the emphasize is most on query capabilities and generating strong reports; 3) spread sheet-oriented which allows the user to develop models to execute DSS analysis; 4) solver oriented DSS which use solver for solving a particular type of problem; 5) rule-oriented DSS in which the knowledge component, which often is an expert system, includes procedural and inferential rules; 6) compound DSS which is a combination of two or more of classifications mentioned above.

Working of proposed DSS

In order to select an architectural style from given choices of architecture styles correctly and precisely, all existing information related to the application should be considered. The proposed DSS will use characteristics of web service architectural styles, domain characteristics of application being developed and NFR (Non-functional requirements) weight age as input for inference and provides appropriate decision complete design of proposed DSS is depicted in Figure 2 in detail [12].

The proposed DSS has five essential components that help in decision making process that are:

- Repositories
- Tool
- Rule-based
- Decision making (CLIPS)
- User interface

Obligations and concerns of every part alongside what they accommodate the DSS is explained as follows.

Repository: We have three types of repositories which are DC (Domain Characteristics), NFRC (Non-Functional

Requirements Characteristics) and ASC(Architectural Style Characteristics) [13]. In DC we have characteristics regarding requirements for different domains like e-commerce, banking, health care apps and others. NFRC contains characteristics provided by different quality attributes and also information regarding no. of sub-attributes of quality attribute provided by specific web service architectural style [14]. ASC have information of all the characteristics of web service architectural styles SOA, ROA and MOA respectively. The characteristics considered in repository in proposed DSS are summarized in Tables 1, 2 and 3.

Table 1 shows the characteristics that MOA, SOA and ROA possess in order to fulfill the requirements of app being developed whose nature is SO (Service-Oriented). Now comparison w.r.t NFR characteristics are given in Table 2. Three NFRs are under considerations mainly security, reliability and performance.

Table 2 showing which characteristics of NFR each architectural style possess with assigned weights to be considered as inputs for system.Now comparison w.r.t domain characteristics is summarized in Table 3.

Tools: Domain characteristic and Architectural stylescharacteristics would be prioritized on basis of no. of characteristics selected. After getting all the required information and priorities of quality attributes, the weighted sum model for NFR's would be applied to DSS and no. of characteristics required of specific web service architectural style and domain characteristics would already be counted while gathering information according to the need of app being developed as shown in Figure 3.

Architectural Characteristics	MOA	SOA	ROA
Heterogeneity	Yes	Yes	Yes
Protocol layering	No	Yes	Yes
Loose coupling	Yes	Yes	yes
Integration style	Yes	Yes	Yes
Resource	No	No	yes
identification			
URI design	No	No	Yes
Resource	No	No	yes
interaction semantic			
Resource	No	No	Yes
relationship			
Contract design	No	Yes	Yes
Data representation	Yes	Yes	Yes
Message exchange	Yes	Yes	Yes
pattern			
Traffic monitoring	Yes	No	No
Traffic	Yes	No	No
determination			
Traffic	Yes	No	No
transformation			
Service description	Yes	Yes	Yes
Service	Yes	Yes	Yes
identification			
Service discovery	Yes	Yes	Yes
Service	Yes	Yes	Yes
composition			

Table 1: Architectural characteristics comparison.

NFR	Characteristics	MOA	SOA	ROA
Security	Encryption	Yes, 2	Yes ,2	Yes, 1
	Integrity	Yes, 1	Yes, 2	Yes, 1
	Authentication	Yes, 1	Yes, 2	Yes, 1
	Authorization	Yes, 1	Yes, 2	Yes, 1
	Non-repudiation	Yes, 1	Yes, 2	Yes, 1
	Confidentiality	Yes, 1	Yes, 2	Yes, 1
Sum of weight (S)		**7**	**12**	**6**
Reliability	Point-to-Point	Yes, 1	Yes, 1	Yes, 1
	Ordered delivery of	No, 0	Yes, 2	Yes, 1
	msg			
	Delivery status	Yes, 1	Yes, 1	Yes, 1
	Elimination of	Yes, 1	Yes, 1	Yes, 1
	duplicate message			
	Resending message	Yes, 1	Yes, 1	No, 0
	Reliable delivery of	Yes, 1	Yes, 2	Yes, 1
	msg			
Sum of weight (R)		**5**	**8**	**5**
Performance	Caching	Yes, 1	No, 0	Yes, 1
	Clustering	Yes, 1	No, 0	Yes, 1
	Load balancing	Yes, 1	No, 0	Yes, 1
	Throughput	Yes, 1	Yes, 2	Yes, 3
	Response time	Yes, 1	Yes, 2	Yes, 3
	Latency	Yes, 2	Yes, 1	Yes, 3
	Execution time	Yes, 2	Yes, 1	Yes, 3
Sum of weight (P)		**9**	**6**	**15**

Table 2: NFR characteristics comparison.

Domain characteristics	MOA	SOA	ROA
Functionality	Yes	Yes	Yes
App type	Yes	Yes	Yes
Run offline	Yes	No	No

Table 3: Domain characteristics comparison.

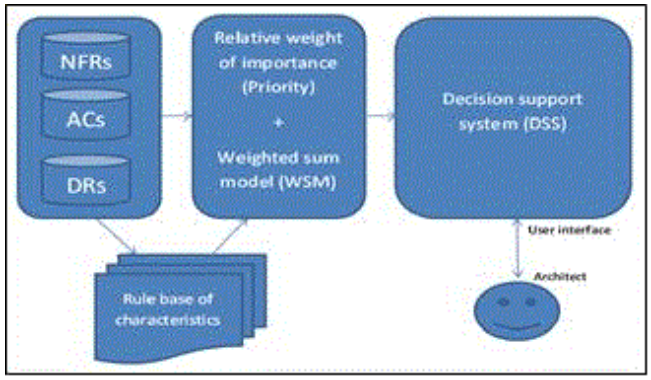

Figure 3: Architecture for proposed DSS.

Where weighted sum model (WSM) is the best known and simplest multi-criteria decision analysis (MCDA) method for evaluating a number of alternatives in terms of a number of decision criteria [2].

In general, suppose that a given MCDA problem is defined on m alternatives and n decision criteria. Furthermore, let us assume that all the criteria are benefit criteria, that is, thehigher the values are, the better it is. Next suppose that w_j denotes the relative weight of importance 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_iWSM-score, is defined as follows:

$$A_i^{\text{WSM-Score}} = \sum_{j=1} w_j a_{ij}, \text{ for } i = 1, 2, 3, \ldots, m. \quad (1)$$

In our case architectures SOA, ROA and MOA are alternatives as (A_i) and quality attributes security, reliability and performance are criteria's as (C_j).

General characteristics of architectures are rules in DSS and priority will be given to the highest number of characteristics selected of specific architecture. For example if we have selected 20 characteristics of SOA, 17 of ROA and 10 of MOA then DSS will recommend SOA [15-18].

The rules, which were extracted from architecture styles characteristics, NFRS, domain characteristics, and the priorities, and incorporated by DSS user i.e. system architect, are used as inputs of the tool as in Figure 4.

Rule Base: For extracting decision, there is need of some rules on the basis of our repositories DC, NFRC and ASC. Simply we can say, rules decide which domain characteristics are required, which characteristics of architecture are required, what is the importance level of each quality attribute. These rules would be stored in rule-based engine and then would be obtained with help of repositories. Rule is generally defined as:

P –> Q or if P then Q (2)

Where P would be characteristics of architecture, domain characteristics or any NFR and Q would be preferred architecture prioritization according to P.

Total 108 rules are implemented in DSS based on characteristics explained earlier

Decision maker (CLIPS): CLIPS has been used fordevelopment of rule based Decision Support System as CLIPS is one of the generally utilized AI (Artificial Intelligence) apparatus utilized for Rule-based DSS. CLIPS gives a firm tool to taking care of a wide mixture of information. Rule-based programming permits heuristics, otherwise "thumb rule," that tag a group of activities executed in specified circumstance [7].

CLIPS help in dealing with the obligation of this part, indeed, accepting as well as sending data to and from every segments belong to DSS. Interface is a "data communicator" possesses an obligation to get needs that the DSS user gave to system through client interface as well as give it to tool as an information that is part of DSS for utilization of it. The priorities entered by software architect become inputs for

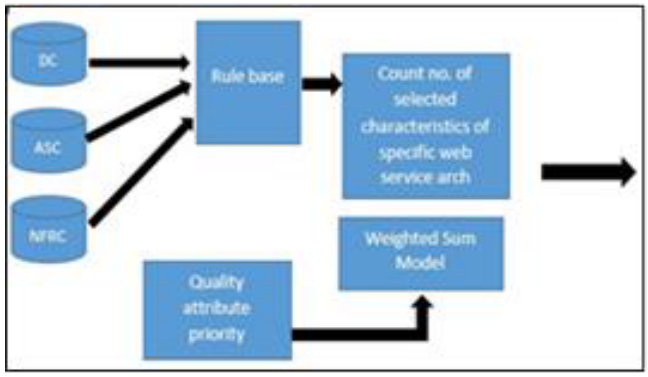

Figure 4: Tool internal Process.

weighted sum model tool that is used for NFRs. Subsequently attaining results after applying WSM, and oblige utilizing different attributes data that are given via software architect, CLIPS inference engine will do surmising on premise of guidelines chain of command. At that point architectural style or may be combination of architectural styles would be suggested for the app under development. The results would be sent to client interface for displaying. Accordingly, DSS dictate recommendations towards the software architect so as to pick most appropriate architecture by utilizing expertise [19,20].

User interface: The user interface is in charge of accepting the data from client in regards to domain qualities needed, which attributes of architectural style are obliged, what is the essentialness level of every quality characteristics as per the way of application being produced; the received data is entered to the decision making tool. In addition, speaking toessential data required by user got via decision making or retrieved via repositories is also the obligation of client interface. The proposed architectural style is spoken to software architect as a recommendation according to need via client interface. The software architect or other vital stake holders could choose the suitable architectural style or styles between proposed one's regarding his insight around the issue.

Actors for interface

The actors for system are as follows:

- Software architect
- Developer
- Technical client

All these actors interact with system by giving desired input i.e. preferred weightage for NFRs security, performance and reliability, then select required characteristics of architecture and domain. Then actors would have recommendations from DSS and can select architecture with highest weightage. The abstract level interface that may appear for actors is depicted in Figure 4. The interface showing the initial inputs.

Automated Architecture Selection

To get a better understanding of our approach we have conducted a hypothetical case study. For this purpose we have identified use case for which architects have to develop web services. In this whole process we consider following Requirements for architectural style selection:

- Business Requirements
- Functional Requirements
- Non Functional Requirements

Along with these requirements we also cater for specific architectural style characteristics and prioritization inputs.

USE CASE: Online Book Order Service

Business requirements: Consider requirements for ordering abook online. Different clients should be able to place online orders for books using this web service in various online stores.

FR01: A client logs onto the book retailer's site and rings a rundown of the obliged articles.

FR02: A request is intended to the "check accessibility" administration which supplies data to the web entrance as to the amount at present in stock.

FR03: The client submits the request for the obliged articles/ books. Order web Service then issues a quotation. The "quotation issue" service demonstrates the cost of the products requested by the individual client making due note of the client status (e.g. rebates, conditions, and so on.).

FR04: In the event that the online payment is done, the payment is acknowledged, and the merchandise are assigned for dispatch. The "send" service exchanges all the essential data to the dispatch system, including the client's conveyance and charging location.

Key Non functional Requirements are:

- NFR01- Security
- NFR02 – Performance
- NFR03 – Reliability

We have considered multicriterian requirements to take input for rule based DSS including general characteristics. A decision problem described over three alternatives A1, A2, A3 namely SOA, ROA and MOA which defined via four criteria C1, C2, and C3 namely Security, Performance plus Reliability. DSS will take weightage of NFR's from architect on basis of the preferences for given case study. The interaction between system and architect is depicted in Figure 5.

The architect would enter weightage for NFR on scale of 1 to 5 where 5 means highly preferred and 1 means less preferred. There are three styles for this use case: Service-Oriented Architecture, Resource-Oriented Architecture also called RESTful, and Message-Oriented Architecture. The satisfaction level of every NFR via different web service architectural styles is summarized in Table 4.

The weightage for each factor is entered as input into system as follows:

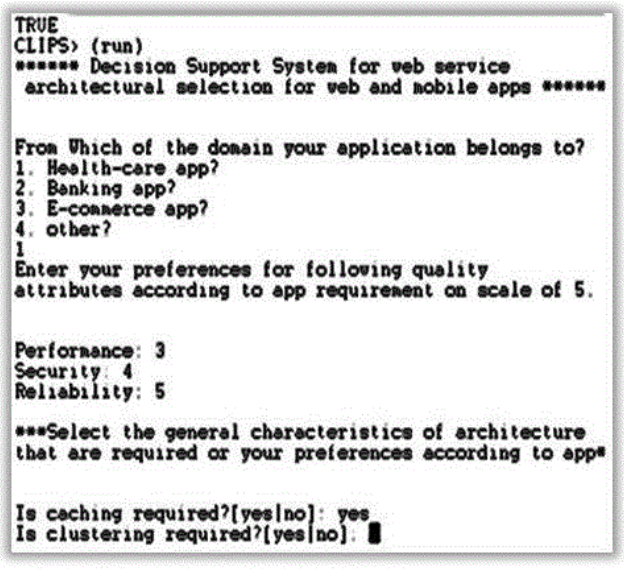

Figure 5: DSS CLIPS interface.

Criteria's	Security	Performance	Reliability
Alternative	--	--	--
SOA	6	6	8
ROA	12	15	5
MOA	7	9	5

Table 4: Satisfaction level.

(C1)Security=5, (C2) Performance= 4, (C3) Reliability= 5

Now subsisting these values in equation (1) derived earlier we get:

(SOA): A1 = w1*a11 + w2*a12 + w3*a13

W(A1) = 5*6 + 4*6 + 5*8 = 94

(ROA): A2 = w1*a21 + w2*a22 + w3*a23

W(A2) = 5*12 + 4*15+ 5*5 = 155

(MOA): A3 = w1*a31 + w2*a32 + w3*a33

W(A3) = 5*7 + 4*9+ 5*5 = 96

Here we have calculated NFR weightage for each architectural style under consideration.

By taking these requirements as input shown in Table 5 the following results are generated by DSS. As represented in Table 6.

At least 21 input for architectural characteristics were taken along with weightage for NFRs. After applying weighted sum model to NFR the final weightage for SOA is at 94, ROA is at 155 and MOA is 96. Similarly from architectural and domain characteristics the weightage is 13 for ROA and SOA and 8 for MOA respectively.

Here we get highest value for ROA in relation to Quality attributes and equal weightage for other characteristics. From these results it is very easy for developers and architects to choose the right architecture as for this use case ROA is more suitable architectural style to choose to develop online Book order web service.

Impact on Decision Making Process

By incorporating different elements involved in manual decision making process into an automated DSS we have streamlined the process for architecture selection for a web service. Decision for selection of a particular architecture for software itself is a complex way and it becomes even more difficult when we are selecting a particular architectural style for web service domain. Here we only considered 03 different types Non-functional requirements for 03 said architectural styles. Here our DSS makes sure to incorporate these NFRs to any selected architectural style by using weighted sum model. Trade-offs analysis may also be done between different NFRs at time of selection of a particular architectural style.

108 different rules were constructed as tree to support various requirements. Now if this process would have been manual or left to system architect without any automation, architecture selection could be subjective depending upon expertise leading to high costs and low quality.

NFR weightage	Architectural + domain characteristics
(SOA): A1	Minimum 21 question asked on the basis of architectural and domain repositories
(ROA): A2	
(MOA): A3	

Table 5: Input by architect.

Architectural Style	Weightage w.r.t NFR	Weightage w.r.t architectural characteristics
SOA	94	13
ROA	155	13
MOA	96	8

Table 6: Results.

Here we discuss some advantages and contribution to the domain with respect to related work.

Advantages

The proposed technique has various advantages which lead towards high quality web services development with reduced, cost and overhead for software development teams. Some of them are:

- Simplified Decision making process
- Un biased decision leading to high quality
- Multi-criterion requirements incorporated
- Automated decision making for architecture selection
- Reduced Risk

Contribution to domain

Though the architecture selection is considered totally a manual process and depends on the knowledge and personal experience of the decision makers. Web services architecture selection is trivial as there are multiple architectural patterns and rate of change in these styles is quite rapid. We believe that there is a criteria as well as a process within architects' mind when they deal with architecture style selection but it is totally based on the judgment of the selector. Our core contribution to this domain is that we present a novel way for architecture selection by incorporating multicriteria requirements. The plus point about our work is that we have considered modern styles which are in use in todays cloud and web development environment. Some researchers have worked in traditional software architecture selection [9] using multicriteria approach but this work lacks the indepth validation of fuzzy path analysis. Booth et al. [11] did work on architectural style selection focusing only on quality attributes by applying AHP technique, though it is a good direction in this domain but they did not consider self characteristics of the styles under question. Our research is quite unique in a sense that we have tried to bring all the aspects in decision making process for web services for example NFRs, domain requirements and above all the self characteristics of these styles. Wieghted Sum Model is a proven approach used in our work for calculating the quality attributes weightage. In a similar work titled "ANP-GP Approach for Selection of Software Architecture Styles" [8] focus on quality attributes and balance among architectural style where goals and objectives are considered in the decision making process. Again this work does not target specific style but applies this method in a general way. In contrast to this work we have targeted modern web services styles and gathered all their characteristics using rule base approach for a better decision making.

Future Work

Web services are everywhere in cloud, big data and IOTs so demand for development and consumption in near future shall increase. In future more architectural styles may be explored and incorporated into this DSS. We have focused key Quality attributes; it can be further enhanced to input other NFRs into system. In future a DSS should be extended to not only suggest architectural style but should be capable to generate suggested architecture style skeleton.

Conclusions

Characteristics of architectural styles are different with each other and, therefore, each one has its own strengths and weaknesses in a problem space. Our research has created new dimensions for architectural selection support. Consequently identifying the Non-

functional requirements, architectural characteristics and domain requirements.

In this paper a DSS for web service architectural selection has been developed which considers core requirements. It uses weighted sum model for NFRs and architectural and domain requirements are prioritized on basis of selected characteristics using rule based model as knowledge engine. The results obtained from this DSS would help software architects, teams and developers in making precise and efficient decisions. Subsequently it will help improve the overall quality of web service development process.

References

1. Svahnberg M, Karlskrona K (2003) Supporting Software Architecture Evolution. Blekinge Institute of Technology Dissertation Series.
2. Nawaz F, Mohsin A (2015) Rule-Based Multi-criteria Framework for SaaS Application Architecture Selection. Artificial Intelligence in Theory and Practice IV 465: 129-138.
3. Triantaphyllou E (2013) Multi-criteria decision making methods: a comparative study. Springer Science & Business Media.
4. Riley G (2015) What is CLIPS?.
5. Mumbaikar S, Padiya P (2013) Web Services Based On SOAP and REST Principles. International Journal of Scientific and Research Publications 3: 1-4.
6. Chen Z (1999) Computational intelligence for decision support. CRC Press.
7. Wang Q, Yang Z (2012) A method of selecting appropriate software architecture styles: Quality Attributes and Analytic Hierarchy Process. Bachelor of Science Thesis, University of Gothenburg.
8. Babu KD, Govindarajulu P, Ramamohana Reddy A, Aruna Kumari A (2011) ANP-GP approach for selection of software architecture styles. Int J Softw Eng 1: 91-104.
9. Moaven S, Habibi J, Ahmadi H, Kamandi A (2008) A decision support system for software architecture-style selection. Software Engineering Research, Management and Applications, 2008. SERA'08, Sixth International Conference, Prague.
10. Belqasmi F, Singh J, Melhem SYB, Glitho RH (2012) SOAP-based vs. RESTful web services: A case study for multimedia conferencing. IEEE Internet Computing 54-63.
11. Booth D, Hass H, McCab F (2004) Web Services Architecture, W3C Working Group Note.
12. Fielding RT, Taylor RN (2002) Principled design of the modern Web architecture. ACM Transactions on Internet Technology (TOIT) 2: 115-150.
13. Pautasso C, Zimmermann O, Leymann F (2008) Restful web services vs. big'web services: making the right architectural decision. Proceedings of the 17th international conference on World Wide Web.
14. Wagh K, Thool R (2012) A comparative study of soap vs rest web services provisioning techniques for mobile host. Journal of Information Engineering and Applications 2: 12-16.
15. Muehlen MZ, Nickerson JV, Swenson KD (2005) Developing web services choreography standards—the case of REST vs. SOAP. Decision Support Systems 40: 9-29.
16. Dudhe A, Sherekar SS (2014) Performance Analysis of SOAP and RESTful Mobile Web Services in Cloud Environment. IJCA Special Issue on Recent Trends in Information Security RTINFOSEC: 1-4.
17. Lofthouse H, Yates MJ, Stretch R (2004) Parlay X Web Services. BT Tech J 22: 81-86.
18. Turban E, Liang TP, Aronson JE (2006) Decision Support Systems and Intelligent Systems. Seventh Edition.
19. Curry E (2004) Message-Oriented Middleware. Middleware of Communications, John Wiley and sons, Chichester, England.
20. Fielding RT (2000) Architecture Styles and Design of Network based Software Architectures. University of California Irvine.