Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 11, Number 6 (2015), pp. 3991-4004

http://www.ripublication.com

Host Environment Focused Software Component Quality Ranking And Selection

K. R. Sekar, K. S. Ravichandran, R. Krishankumar

School of Computing, SASTRA University, Thanjavur, India.

Abstract

Selection of software component for API is accomplished based on Quality of Service (QoS) attributes. These QoS attributes are mapped based on the application which consumes it. Researches never minded of the operating systems, where the applications are hosted. Hence an attempt towards component selection for application types with respect to the host operating systems has been proposed in this paper. A breakthrough idea is attempted here, which sequences the QoS attributes based on the type of application with respect to the operating system. ***Rank correlation methodology*** is the technique devised in this paper, which works based on threshold values of QoS attributes. And, the proposed work is confined to Web application, Expert system, and embedded application. The operating systems bound to these applications are in different varieties. This work is proposed to achieve in high quality component selection, subsequently resulting in compatibility based selection for the users and providing ultimate flexibility in software development.

Keywords: Rank correlation Methodology, Application programming Interface, Quality of Service

1. Introduction

In the present software industries software Components are categorized in two parts, namely ***commercial*** and ***customized*** components. Normally, commercial components are with the third party vendor rather than the supplier. Users of such products, like software companies, buy their required component only through suppliers. An appropriate selection of the component is basically through a third party vendor description towards the component, and such a component is referred to as a commercial component. Customized components are made for the literal need of software design and their services. These are accomplished with good accuracy because of their nature. Customized components need more expertise, which is time consuming and not cost

effective. Core functionalities are available in the same location to enhance the product's reliability. On the other hand, marketable components do not provide precise services-either too many or too less. Compared to specially formulated components, saleable components are more economical and need less effort. Considering the above facts, the Company can answer the important question whether to make or buy and overcome the herculean task of identifying the most economical and reliable component from the enormous component market, thus answering the second question as well.

Software component efficiency and its compatibility are measured with respect to the combination of the applications and operating systems. For measuring the efficiency, the only yard stick available is QoS. Functional aspects are also taken as a measurable scale and the same is denoted in the below cited paper. Functional aspects, metrics and domain-specificity are common factors to identify the right component for the given architecture. Once the component satisfies the above said factors, they are more reliable and can accommodate more components [5].

The QoS of the component selection is based on the following: QoS parameters for various applications like Web, Expert and Embedded systems. ***Web Application*** employed with compatibility, manageability, deplorability, security, performance, dependency, cost, availability, reachability, training and complexity. ***Expert Application*** engaged with availability, consistency, trainability, capturability, energetic and efficiency**.** ***Embedded Application*** occupied with maintainability, expertise, efficiency, consistency, cost, dependability, robustness, performance, user friendly, interoperability and independence**.** Altogether, 20 QoS are used here as high peers and remaining, if any, are used as low peers. **Operating Systems** like 1. Windows XP (XP), 2. Windows Vista (WV), 3. Apple Macintosh OS X (AM) 4. Linux (LX), 5. Novel Netware (NN), 6. UNIX (UX), 7. Sun Solaris (SS). ***Operating Systems*** has got its own significant QoS.

The selection process should be via ***Q-Component*** and ***Q-application***. **Q** represents Quality-based commodity. Q-application needs Q-component for making their services better. Q-component selection for a Q-application desires some quality so that the Q-component will be rightly fitted in the Q-application. Each and every Q-component has got a description rather, with service directory, through which the components are identified for their services [1]. Quantifying the QoS ingredients always gives more accuracy in the selecting components. Quantifying every ingredient of the component gives more vision for dependability and Mean Turn Around Time (MTAT) rather, response time is calculated for their component selection in the architecture using UniFrame approach [3].

Applications and the Operating Systems also have good QoS as stated above. Their QoS are classified into high peer and low peer according to the user's need. High Peer has got some QoS which are more relevant with high functionality of certain applications' and operating systems' combination. High frequency QoS are offered in a high peer segment and the remaining will be placed inside low peer segment. Ranking has been made to the QoS according to the chronological order with respect to the applications and operating systems. Each application interacts with multiple operating sys-

tems, through which it is aimed at identifying the ingredient of high degree QoS for certain combinations. Reliability and security are important factors to be noticed.
Function and non functional activities are the ideologies that bring good quality of measure towards the selection of a component. Functionality-based services, non-functionality-based services and user-based services are approaches used to identify the best selection (QSS) of the component for service oriented architecture (SOA) [11] [15]. For finding optimum selection of the components, methodologies like optimization and soft computing techniques are employed in many research papers. The current paper focuses on Web Service components which are based on Service Selection, with QoS being retained. For this purpose particle swarm optimization and Fuzzy logic assisted in selecting appropriate components by decomposing global constraints into local constraints, ultimately leading to a more cost effective technique [12].
Rank correlation coefficient methodology is applied to find the suitable reusable component that matches with the above said applications and operating systems. The arrangement of the paper is as follows: Introduction elaborates in section 1. Section 2 describes elaborately the work related to the paper, the proposed methodology is explained in section 3, Results and discussions in the section 4, the conclusion is given in section 5 and inférence of the paper in section 6.

2. Related Work

In web-based applications, web services render their services to a great extent. Dynamic discovery and automatic configuration with the web service play an essential role in the recent times.
Optimizing the QoS and global constrains to the above said. This paper focuses on integer programming and optimal global QoS decomposition. Another thing is global constraints are taken as a local constraints for solving the problem. Distributed local collection of web services is also proposed. These are the methodologies that outperform the existing computational time to the core [9]. Component-based development system is inevitable in the modern world. For any rapid application development, software components play an essential role. QoS optimization and global constraints are the considerations already available in the market. New research ideas are marching towards the domain and application engineering is improving reliability and reusability. The reusability of components for multiple applications can be found at a short period of time through matrix formulation. Every time applications are grouped, components give more mileage to the software developers [13].
In the application of healthcare and road traffic systems, reliable components are essential to provide the services to the APIs. Model-based prediction and reliability measuring are the vital areas to be noticed while creating components for business transactions. System level protection can be adapted via incremental design and iterative development. The above said measures should be taken during the design and implementation phase [7].
For the past two decades, software components are an unavoidable commodity for the applications in the side of academics and industries. Selecting the appropriate compo-

nent for a business service is a difficult task. Coupling and cohesion are some methodologies available in previous research for the selection of components. Maximizing the functional activities of the application brings up great quality to small and medium-sized Companies. Genetic algorithms are the most preferred method for the researcher to select a component [4].

3. Proposed Methodology

The need of finding an efficient component is that it helps us to optimize business applications. There are different types of applications out of which few important ones are selected.

1. Web based applications,
2. Expert systems applications and
3. Embedded applications.

Ontology-based methodology nowadays plays an important role in evaluating QoS. Applications transformed into coupled services through web services are connected by semantic similarities. OWL ontology for QoS, evaluates in QoS matching environment QoS metrics were developed to measure quality. The matching algorithm developed prevents mismatching of components with web applications. [2]. Ontology-based and multi attribute decision also offers to select the correct component. Web service composition makes the distributed system work reliably. Component-to-component web service offers interoperability at a great deal. The QoS ontology-based ranking and the persons working under the application should be satisfied with the required services. Automatic configuration and dynamic discovery of the web service will bring up better selections to the application with different levels of demands. Analytic Hierarchical Process and Multi-Decision attribute will afford to give more flexibility and more vibrancy to the application. [14]. Logic-based and text retrieval techniques are employed for the selection of the appropriate software component selection. Semantic Web Service selection techniques were developed based on logic based matching and text retrieval strategies. These were semantically matched with Web Services Description Language specifications. The matchmaker tool developed increased the performance using query response time [8].

All applications performed today are dependent on the operating system being used. Here some of the famous Operating Systems are considered for the purpose of finding the suitable efficient component in an application. High peer and low peer QoS are listed for the operating systems. High peers have the QoS like performance, consistency, usability, reliability, scalability, accuracy, interface complexity, security, interoperability and customization. Low peers are listed with maintenance, documentation, flexibility, robustness and portability. Here a comparison with the applications and operating systems is made through QoS using Rank Correlation Coefficient method. This method also selects appropriate component with compatibility. Dominance QoS for the applications are taken into account for this process.

Table 1: Web Applications with QoS

Web appln\Qos	CM	MG	DP	SE	PE	DE	CT	AV	RC	TR	CX
Meebo	1	0	1	1	1	1	1	1	1	0	0
Facebook	1	1	1	0	1	0	0	1	1	0	0
Own cloud	1	1	1	1	1	0	1	1	1	1	1
Openstreet map	1	1	1	0	1	0	0	1	1	0	0
Youtube	1	1	1	0	1	0	0	1	0	0	0

Legend 1:Compatibility (CM), Manageability (MG), Deployability (DP), Security (SE), Performance (PE), Dependency (DE), Cost (CT), Availability (AV), Reachability (RC), Training (TR) and Complexity (CX)

Table 1 has 1's and 0's. '1' represents QoS are matching with the application and '0' represents no matching with the given applications. Rank Correlation Coefficients are applied Row wise and Column wise to gauge the applications with respect to the QoS. According to the weight of the application, it is sorted in descending order. Those entries are followed in the below Tables as such.

Table 2: Rank Correlation Coefficient (Row Wise)

Web appln\Qos	CM	MG	DP	SE	PE	DE	CT	AV	RC	TR	CX
Own cloud	1	1	1	1	1	0	1	1	1	1	1
Facebook	1	1	1	0	1	0	0	1	1	0	0
Openstreet map	1	1	1	0	1	0	0	1	1	0	0
Meebo	1	0	1	1	1	1	1	1	1	0	0
Youtube	1	1	1	0	1	0	0	1	0	0	0

Reference –Legend 1

Table 3: Rank Correlation Coefficient (Column Wise)

Web appln\Qos	CM	DE	PE	AV	RC	MG	SE	CX	TR	CT	DP
Own cloud	1	1	1	1	1	1	1	1	1	1	0
Facebook	1	1	1	1	1	1	0	0	0	0	0
Openstreet map	1	1	1	1	1	1	0	0	0	0	0
Meebo	1	1	1	1	1	0	1	1	0	0	1
Youtube	1	1	1	1	0	1	0	0	0	0	0

Reference –Legend 1

Web Applications such as Own cloud, Facebook, Openstreet map, Meeba and Youtube survives on QoS support like compatibility, deployment, performance and availability upto 100%. Own cloud, Facebook and Openstreet map needs QoS support like reachability and manageability upto 100%. Other QoS are not contributed more than 50%, thus needing to be ignored. Dense ones should be taken for granted to identify

the needs of the QoS to the respective applications. The remaining applications like Expert systems and embedded systems follow the same scenario.

Table 4: Expert Applications with QoS

Expert System / QoS	AV	CO	TR	CP	ER	EF	UF	IO	IN
Financial Decision Making	1	1	1	1	0	1	1	0	0
Knowledge Publishing	1	1	1	1	0	1	1	0	0
Process Monitoring and Control	1	0	1	1	0	0	0	0	0
Online medical system for diagnosing a problem	1	1	1	1	1	1	1	0	0
Government services such as working out tax and benefits	1	1	0	1	0	0	0	1	1

Legend 2: Availability (AV), Consistency (CO), Trainability (TR), Capturability (CP), Energetic (ER), Effieciency (EF), User Friendly (UF), Interoperability (IO) and Independence (IN).

Table 5: Rank Correlation Coefficient (Row Wise)

Expert System / QoS	AV	CO	TR	CP	ER	EF	UF	IO	IN
Online medical system for diagnosing a problem	1	1	1	1	1	1	1	0	0
Financial Decision Making	1	1	1	1	0	1	1	0	0
Knowledge Publishing	1	1	1	1	0	1	1	0	0
Government services such as working out tax and benefits	1	1	0	1	0	0	0	1	1
Process Monitoring and Control	1	0	1	1	0	0	0	0	0

Reference Legend 2

Table 6: Rank Correlation Coefficient (Column Wise)

Expert System / QoS	AV	CP	CO	TR	EF	UF	ER	IN	IO
Online medical system for diagnosing a problem	1	1	1	1	1	1	1	0	0
Financial Decision Making	1	1	1	1	1	1	0	0	0
Knowledge Publishing	1	1	1	1	1	1	0	0	0
Government services such as working out tax and benefits	1	1	1	0	0	0	0	1	1
Process Monitoring and Control	1	1	0	1	0	0	0	0	0

Reference Legend 2

In Expert System applications, QoS like availability, capturability and Consistency working for all the application up to 93%. Trainability, Efficiency and User Friendly

QoS working for three expert applications like Online medical system for diagnosing a problem, Financial Decision Making and Knowledge Publishing up to 100%.

Table 7: Embedded Applications with QoS

Embedded Appln\Qos	MT	EX	EF	UF	CO	CT	DE	RB	PE
Smart phones	1	1	1	1	0	1	0	1	1
Hazard detecting system	1	1	1	0	1	1	1	1	1
ATM	1	0	1	1	1	0	0	1	1
Elevators	1	0	1	1	0	0	0	1	1
Finger Recognization	1	1	1	1	1	0	0	1	0
Airport security	1	1	1	1	1	1	0	1	0

Legend 3: Maintainability (MT), Expertise (EX), Efficiency (EF), User friendly (UF), Consistency (CO), Cost (CT), Dependency (DE), Robustness (RB) and Performance (PE).

Table 8: Rank Correlation Coefficient (Row Wise)

Embedded Appln\Qos	MT	EX	EF	UF	CO	CT	DE	RB	PE
Airport security	1	1	1	1	1	1	0	1	0
Finger Recognization	1	1	1	1	1	0	0	1	0
Smart phones	1	1	1	1	0	1	0	1	1
Hazard detecting system	1	1	1	0	1	1	1	1	1
ATM	1	0	1	1	1	0	0	1	1
Elevators	1	0	1	1	0	0	0	1	1

Reference Legend 3

Table 9: Rank Correlation Coefficient (Column Wise)

Embedded Appln\Qos	MT	EF	RB	EX	UF	CO	CT	PE	DE
Airport security	1	1	1	1	1	1	1	0	0
Finger Recognization	1	1	1	1	1	1	0	0	0
Smart phones	1	1	1	1	1	0	1	1	0
Hazard detecting system	1	1	1	1	0	1	1	1	1
ATM	1	1	1	0	1	1	0	1	0
Elevators	1	1	1	0	1	0	0	1	0

Reference Legend 3

In Embedded System applications, QoS like maintainability, efficiency and robustness work for all applications upto 100%. Expertise, user-friendliness and consistency in QoS works for four applications like airport security, finger recognition, smart phones and hazard detecting system work upto 83%. QoS like cost and performance for smartphones, hazard-detecting system, ATMs and elevators work up to 75%.

Table 10: Operating Systems and QoS Reviews

OS/QoS	Reviews	SE	MA	PE	RE	FL	CO	DE	EF	TR
Win Xp	**25**	20	20	19	18	17	16	14	12	9
Win Vs	**27**	23	22	20	20	13	15	12	11	9
MaC	**32**	28	16	15	14	13	23	20	22	9
LnX	**30**	26	24	23	22	21	12	14	13	17
NN	**20**	14	13	12	11	9	17	8	10	10
UnX	**23**	18	17	17	16	14	10	10	9	14
Solaris	**18**	11	11	10	10	7	11	8	7	6

Legend 4: Security (SE), Maintenance (MA), Performance (PE), Reliability (RE), Flexibility (FL), Consistency (CO), Deployment (DE), Efficiency (EF) and Training (TR).

Legend 5: Window XP (Win Xp), Windows Vista (Win Vs), Apple Macintosh (MaC), Linux (LnX), Novel Netware (NN) and Unix (UnX).

Here in the operating system, previous Rank Correlation Coefficient methodology is followed with an additional parameter-review. Other entries available there represent the demand for the QoS with respect to the operating systems. The percentage is calculated for different QoS and the same is normalized from 0 to 1. If the value is more than or equal to 0. 50 it is treated as ‘1’ else ‘0’. The calculations are made accordingly for the beneath tables.

Table 11 : Operating Systems and QoS Percentage values are normalized between 0 to 1

OS/QoS	Reviews	SE	MA	PE	RE	FL	CO	DE	EF	TR
Win Xp	**25**	0. 81	0. 79	0. 78	0. 72	0. 69	0. 62	0. 57	0. 48	0. 37
Win Vs	**27**	0. 85	0. 81	0. 77	0. 73	0. 49	0. 53	0. 43	0. 4	0. 35
MaC	**32**	0. 89	0. 49	0. 46	0. 43	0. 41	0. 39	0. 61	0. 68	0. 29
Lnx	**30**	0. 86	0. 81	0. 76	0. 73	0. 69	0. 49	0. 45	0. 42	0. 58
NN	**20**	0. 68	0. 63	0. 59	0. 56	0. 46	0. 55	0. 42	0. 51	0. 48
UnX	**23**	0. 79	0. 76	0. 72	0. 68	0. 62	0. 47	0. 42	0. 39	0. 62
Solaris	**18**	0. 61	0. 59	0. 57	0. 55	0. 41	0. 62	0. 44	0. 41	0. 36

References Legend 4 and Legend 5

Table 12: Rank Corrélation coefficient

OS/QoS	SE	MA	PE	RE	FL	CO	DE	EF	TR
Win XP	1	1	1	1	1	1	1	0	0
Win VS	1	1	1	1	0	1	0	0	0
MaC	1	0	0	0	0	0	1	1	0
LnX	1	1	1	1	1	0	0	0	1
NN	1	1	1	1	0	1	0	1	0
UnX	1	1	1	1	1	0	0	0	1
Solaris	1	1	1	1	0	1	0	0	0

References Legend 4 and Legend 5 for the Table 12

Table 13: Rank Correlation coefficient (Row Wise)

OS/QoS	SE	MA	PE	RE	FL	CO	DE	EF	TR
Win XP	1	1	1	1	1	1	1	0	0
LnX	1	1	1	1	1	0	0	0	1
UnX	1	1	1	1	1	0	0	0	1
NN	1	1	1	1	0	1	0	1	0
Solaris	1	1	1	1	0	1	0	0	0
Win VS	1	1	1	1	0	1	0	0	0
MaC	1	0	0	0	0	0	1	1	0

References Legend 4 and Legend 5

Operating Systems like Windows XP, Linux, Unix, Novel Netware, Solaris and Windows Vista have QoS like security, maintainability, performance and reliability up to 100% support. Window XP, Linux and Unix having the QoS like consistency, Deployment and Flexibility up to 56% Support and Novel Netware, Solaris, Window Vista and Apple Macintosh having QoS like Consistency and Deployment up to 50% Support. Therefore, 50% and above QoS are useful for different operating systems.

Table 14: Rank Correlation coefficient (Column Wise)

OS/QoS	SE	MA	PE	RE	CO	DE	FL	EF	TR
Win XP	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0
LnX	**1**	**1**	**1**	**1**	**0**	**0**	**1**	0	1
UnX	**1**	**1**	**1**	**1**	**0**	**0**	**1**	0	1
NN	**1**	**1**	**1**	**1**	**1**	**0**	0	1	0
Solaris	**1**	**1**	**1**	**1**	**1**	**0**	0	0	0
Win VS	**1**	**1**	**1**	**1**	**1**	**0**	0	0	0
MaC	1	0	0	0	**0**	**1**	0	1	0

References Legend 4 and Legend 5

Model-based selection of the software component is also used. Web services are increasing in the market for all types of web based applications. Web services are providing business transaction services to the web applications. Selecting optimal web service is a Herculean task and can be made easier using QoS. Control pattern flow approach optimizes QoS, so that to select a correct web service component. UML is used here for modeling QoS and the same is treated as the measure for selecting a right web service. Local selection and global selection web services are compared to obtain an appropriate web service for multiple web applications. These type of web services is used in the field of Gas Dispersion cases. [10]

Table 15: Components, Applications, QoS and Support

Components	Applications	QoS	Support
C1	**WEB Applications**-Own cloud, Facebook, Openstreet map, Meeba and Youtube	Compatibility, Deployment, Performance and Availability	100 %
C2	**WEB Applications**-Own cloud, Facebook and Openstreet map	Reachability and Manageability.	100 %
C3	**Expert Systems**-Online medical system for diagnosing. Financial Decision Making. Knowledge Publishing Government services such as working out tax and benefits. Process Monitoring and Control	Availability, Capturability and Consistency	93%
C4	**Expert Systems**- Online medical system for diagnosing. Financial Decision Making and Knowledge Publishing	Trainability, Efficiency and User Friendly	100%
C5	**Embedded Systems**- Airport security, Finger Reorganization Smart phones, Hazard detecting system, ATM,, Elevators	Maintainability, Efficiency and Robustness	100%
C6	**Embedded Systems**- Airport security, Finger Recognization, Smart phones and Hazard detecting system working	Expertise, User Friendly and consistency	83%

Table 15: Components, Applications, QoS and Support (Cont…)

Components	Applications	QoS	Support
C7	**Embedded Systems-** Smart phones, Hazard detecting system, ATM and Elevators	Cost and Performance	75%
C8	**Operating System-** Windows XP, Linux, Unix, Novel Netware, Solaris and Windows Vista	Security, Maintainability, Performance and Reliability	100%
C9	**Operating System-** Window XP, Linux and Unix	Consistency, Deployment and Flexibility	56%
C10	**Operating System-** Novel Netware, Solaris, Window Vista and Apple Macintosh	Consistency and Deployment	50%

In Table 15, Software components like C1, C2, C3….. C10 are employed and they have their respective QoS and that it will provide necessary services to the required application. The same component is used here as a reusable component for different applications. Three different categories of applications like Web, Expert and Embedded systems have got two or three components for their own applications. On the other hand Operating systems also got their reusable component with required QoS. Support level of the components are measured and mentioned in the above table.

Permutation and combinations between the different categories of applications and operating systems are made, to find the required components to make a combination between an operating systems and a software application. Mapping should be made for the compatible selection of the essential reusable software components are fortified in the Table 16.

In Table 16, Software applications usually running over an operating system is a normal phenomena and identifying the required compatible reusable software components plays an important role and the same will be for provided 'N' number of services to different categories of applications. Permutation and combinations are applied between the operating systems and applications using the Table 15. Here Table 16 has different components that literally help and provides vital services to the operating system with respect to their application. High peer software components have a dominant QoS and the same is taken into account for measurement in terms with different operating systems and applications.

Those reusable software components are used and they provide a long lasting effect to the customers need.

Table 16: Components selection with respect to different applications and operating systems

	Applications/Operating Systems	Win XP	Linux	Unix	Novel Netware	Solaris	Win Vista	Apple Macintosh
Web Applications	Own cloud	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C10
	Facebook	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C10
	Openstreet map	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C9,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C8,C10,C2	C1,C10
	Meebo	C1,C8,C9	C1,C8,C9	C1,C8,C9	C1,C8,C10	C1,C8,C10	C1,C8,C10	C1,C10
	Youtube	C1,C8,C9	C1,C8,C9	C1,C8,C9	C1,C8,C10	C1,C8,C10	C1,C8,C10	C1,C10
Expert Applications	Online medical system for diagnosing a problem	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C10,C4
	Financial Decision Making	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C10,C4
	Knowledge Publishing	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C9,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C8,C10,C4	C3,C10,C4
	Government services such as working out tax and benefits	C3,C8,C9	C3,C8,C9	C3,C8,C9	C3,C8,C10	C3,C8,C10	C3,C8,C10	C3,C10
	Process Monitoring and Control	C3,C8,C9	C3,C8,C9	C3,C8,C9	C3,C8,C10	C3,C8,C10	C3,C8,C10	C3,C10
Embedded Applications	Airport security	C5,C8,C6,C9	C5,C8,C6,C9	C5,C8,C6,C9	C5,C8,C6	C5,C8,C6	C5,C8,C6	C5,C10,C6
	Finger Recognization	C5,C8,C6,C9	C5,C8,C6,C9	C5,C8,C6,C9	C5,C8,C6	C5,C8,C6	C5,C8,C6	C5,C10,C6
	Smart phones	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C10	C5,C8,C6,C7,C10	C5,C8,C6,C7,C10	C5,C10,C6,C7
	Hazard detecting system	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C10	C5,C8,C6,C7,C10	C5,C8,C6,C7,C10	C5,C10,C6,C7
	ATM	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C7,C10	C5,C8,C7,C10	C5,C8,C7,C10	C5,C10,C7
	Elevators	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C6,C7,C9	C5,C8,C7,C10	C5,C8,C7,C10	C5,C8,C7,C10	C5,C10,C7

Legend 6 : Windows XP (Win XP) and Windows Vista (Win Vista)

4. Results and Discussions

Through the proposed ideology it is intended to say that for any specific combination of Application and Operating System, the QoS lying in the High Peer will provide efficient compatible component. Thus, while choosing the components for any application those components can be preferred and resulting into an efficient system. Similarly, the QoS lying in Low peer the combinations can also be identified thus providing the information of the component with least efficiency, so that more importance can be paid to that component in providing high functionality of that particular application. Rank Correlation coefficient provides greater accuracy in the selection of compatible component so that everything works in the combination of any operating system with applications. Table 16 clearly shows the compatible reusable software components that can be implemented for different operating systems with varieties of applications.

5. Conclusion

The Table 16 shows a set of components can be chosen for any combination of operating systems and applications. Selecting the preferable compatible software components among a given set of valid components is done. The components are the ones which have any of the QoS in the high peer for that specific combination of operating system and application. Hence, they are capable of providing a more efficient system. In the future, many other methodologies may be applied to find the suitable software component to work under any operating system and applications. But at this moment, our finding provides good efficiency in the selection process of reusable software component.

6. Inference

This research article explores three domains-web, expert and embedded alone with its corresponding applications, operating systems and components. Rank correlation coefficient is applied upon this and dense matrix mapping is employed to use the right software component for the respective application adhering to the specific operating system.

Reference

1. Daniel A. Menasce, HongleiRuan and Hassan Gomaa (2004), ” A Framework for QoS-Aware Software Components”, WOSP'04,, pp 186 – 196
2. Giallonardo. E & E. Zimeo, “More Semantics in QoS Matching (2007) “, Proceedings of the IEEEInternational Conference on Service-Oriented Computing and Applications SOCA'07),, pp 163-171
3. GirishJ. Brahnmath, Rajeev R. Raje, Andrew M. Olson, Mikhail Auguston, Barrett R. Bryant and Carol C. Burt, (2002). "A Quality of Service Catalog for Software Components”, The Proceedings of the Southeastern Software Engineering Conference, Huntsville, labama,, pp, 441-452.
4. Kwong C. K, L. F. Mu, J. F. Tang, X. G. Luo. (2010). ”Optimization of software components selection forcomponent-based software system development”, Computers& Industrial Engineering, pp 618–624
5. Lamia Yessad and ZizetteBoufaida, (2011). "A QoS Ontology-based Component Selection”, International Journal on Soft Computing (IJSC), Vol. 2, No. 3, pp 16-30.
6. M. Thirumaran, P. Dhavachelvan, S. Abarna and G. Aranganayagi.. (2010). ”Architecture for Evaluating Web Service QoS Parameters using Agents”, International Journal of Computer Applications, (0975 – 8887), Vol. 10– No. 4, pp15-21
7. Marko Palviainen, AnttiEvesti, EilaOvaska. (2011). “Thereliability estimation, prediction and measuring ofcomponent-based software”, The Journal ofSystems and Software, pp 1054–1070
8. Matthias Klusch, Patrick Kapahnke, (2008). “SemanticWeb Service Selection with SAWSDL-MX”, German Research Center for Artificial Intelligence, Vol. : 416, pp 3-16
9. Mohammad Alrifai, Thomas Risse. (2009). "Combiningglobal optimization with local selection forefficientQoS-aware service composition”, InProceedings of the 18th international conference onWorld Wide Web, ACM, ISBN: 978-1-60558-487-4, pp 881-890
10. Roy Grønmo, Michael C. Jaeger. (2005). ”Model-DrivenMethodology for Building QoS-OptimizedWeb Service Compositions”, In Proceedings of thefifth IFIP International Conference onDistributed Applications and Interoperable Systems(DAIS’05), Springer Press, pp 68–82

11. Sathya. M, M. Swarnamugi, P. Dhavachelvan and G. Sureshkumar. (2011). ”Evaluation of QoS Based Web-Service Selection Techniques for Service Composition”, International Journal of Software Engineering (IJSE), Vol. (1): Issue (5), pp 73-90
12. Sun. Q, S. Wang, H. Zou& F. Yang (2011). ” QSSA: A QoS-aware Service Selection pproach”, International Journal of Web and Grid Services, Vol. 7, No 2, pp 147-169
13. Tang. J. F, L. F. Mu, C. K. Kwong, X. G. Luo. (2011). ”Anoptimization model for software component selection under multiple applications development”, European Journal of Operational Research, Vol. 212, Issue 2,, pp 301–311
14. VuongXuan Tran, HidekazuTsuji, RyosukeMasuda. (2009). ” A new QoS ontology and its QoSbasedranking algorithm for Web services”, Journal onSimulation Modelling Practice and Theory, Science Direct, Vol. 17, Issue 8, pp 1378-1398
15. Yessad. L, & Z. Boufaida. (2010). ”QoS-based Component Selection using Semantic Web Technologies”, In Proceedings of ICWIT’10, 231-240, pp 16-19