Journal of Theoretical and Applied Information Technology
15th December 2012. Vol. 46 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCEMENT OF SOFTWARE COMPONENT COMPATIBILITY IN ENTERPRISE SOFTWARE DEVELOPMENT USING HIGH PEER AND LOW PEER

K.R. SEKAR[1], K.S. RAVICHANDRAN[1], ABHISHEK SINGH[1]
School Of Computing, Sastra University, Tirumalaisamudram,
Thanjavur, Tamilnadu 613401, India

ABSTRACT

In any business enterprise application, services playing a vital role and are provided by the software components. Every application requires a myriad of components based on umpteen types of services. Among the components, finding an enhanced compatible component is a herculean task. The work can be accomplished in terms with quality of service (QoS) with respect to different types of Applications and Operating systems. The QoS relatively paramount for a certain combination can be identified through the outcome. Here High and Low Peers have been segregated by identifying the threshold. A novel methodology is incorporated using mapping with limits for High Peer and Low Peer qualities. The above scenario will facilitate the application domain to make use of perfect fitting component available in bountiful.

Keywords: *Software Component (SC), High Peer, Low Peer, QoS, Application, Operating System (OS).*

1. INTRODUCTION

Components are categorized into two parts in the present software industry, namely commercial and customized components. Normally commercial components are in the end of the third party vendor, rather the supplier. Customers like software companies buy their required component only through the suppliers. Appropriate selection of the component is basically through third party vendor description towards the component, and such a component is referred to as commercial component. Customized components are made for the literal need of software design and their services. They are more accomplished and have good accuracy because of their nature. Customized components need more experts, are time consuming and are not cost effective. Core functionalities are available in the same location so that the customers can have high reliability with a fat component. On the other hand, marketable components do not provide the precise services, rather too many or little bit less. Compared to specially formulated components, saleable components are more economical and need less effort. Considering the above facts, the company can answer the billion-dollar question whether to make or buy and overcome the herculean task of identifying the most economical and reliable component from the enormous 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 [1]. The QoS of the component selection is based on the following 16 component parameters which are, Performance (Pe), Security (Se), Scalability (Sc), Accuracy (Ac), Reliability(Re), Portability (Po), Documentation (Do),Usability(Us),Consistency(Co),Customization (Cu), Maintenance (Ma), Interface complex ability (Ic),Robustness(Ro),Flexibility(Fl), Interoperability (In), and Semantic (Sm). _____ (1)
Applications and the operating systems also have good quality of services as stated above. Their QoS are classified into High Peer and Low Peer. High Peer has got some QoS which are more relevant with high functionality of certain Application and OS combination. High frequency QoS are available in a High Peer segment and the remaining will be placed inside Low Peer segment. Ranking has been made according to the chronology in the segments for every QoS. Each application interacts with multiple OS, through which the ultimate vision is to identify the ingredient of high degree QoS for certain combination.The arrangement of the paper is as follows. Introduction elaborates in section-1. Section - 2 describes elaborately the work related to this paper, the proposed methodology is explained in section-3 and finally, the conclusion is given in section-4.

Journal of Theoretical and Applied Information Technology
15th December 2012. Vol. 46 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. RELATED WORKS

Functional aspects, metrics and domain-specificity are the common factors to identify the right component for the architecture. Once the component satisfies the above said factors, they are more reliable and accommodated components [1].
Q-application needs Q-component for making their service brilliant. 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 [2].
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].
In the proposed work, scenarios are developed for each Quality attribute, also the risk and the tradeoff is mentioned. Performance, request satisfaction, reliability and security are some of the QoS evaluated to get the right piece of component for the architecture, like web services [4].
Functional based services, nonfunctional based services and user based services are approaches used to identify the best selection (QSS) of the component for service oriented architecture (SOA) [5].

3. PROPOSED METHODOLOGY

The need of finding an efficient component is that it helps us in optimizing business applications. There are different types of applications out of which few important ones are selected. These are given in equation (2).
1.Web based applications (W), 2.Network applications(N), 3.Expert systems applications(E), 4.Desktop applications(D), 5.Embbeded applications(Em), 6.Tool based applications(T)
------ (2)
All the applications performed today are dependent on the OS we use. Here some of the famous OS are considered for the purpose of finding the suitable efficient component in an application and are listed in equation (3).
1.Windows XP(X), 2. Windows Vista(V), 3. Apple Mac OS X(M), 4. Linux(L), 5. Novel Netware(N), 6. UNIX(U), 7. Sun Solaris(S) ------ (3)

The methodology uses the QoS with high effect on the combination to determine the efficient, compatible components for any particular Application and OS. The components QoS is given in equation (1). Now, these QoS are arranged into chronological order. These QoS are then assigned with rank specific to the particular application or OS, and the adhered table is prepared for the application. Then these ranked QoS for the applications are mapped into the High Peer and Low Peer depending on the place in the priority of the QoS for the particular Application. Thus Table-1 is obtained for High peer QoS, Table-2 for the Low Peer, for applications.

Table 1: High peer Qos for various Applications.

S. No.	Applications	High Peer				
1	Web based applications	Pe	Co	Us	Re	Sc
		0.30	0.15	0.10	0.07	0.06
2	Network applications	Re	Se	Pe	Ac	Ic
		0.30	0.15	0.10	0.07	0.06
3	Expert systems applications	Ac	Re	Ic	Pe	Se
		0.30	0.15	0.10	0.07	0.06
4	Desktop applications	In	Us	Re	Ac	Se
		0.30	0.15	0.10	0.07	0.06
5	Embedded applications	Se	Pe	Sc	Re	Co
		0.30	0.15	0.10	0.07	0.06
6	Tool based applications	Re	Pe	Us	Co	Cu
		0.30	0.15	0.10	0.07	0.06

Here the value of the QoS for High Peer is found using the formula
Value = Rank of the QoS/(Sum of Ranks in High Peer + Sum of Ranks in Low Peer) eqn.(I)
whereas the value for the Low peer QoS is always a constant i.e. a minimum limited value of 0.03.

Table-2: Low Peer QoS for various Applications

S. No.	Applications	High Peer										
1	Web based applications	Ma	Do	Se	Ac	Po	Fl	Cu	Ic	Ro	In	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03
2	Network applications	Co	Us	Sc	Ma	Do	In	Po	Fl	Cu	Ro	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03
3	Expert systems applications	Us	Co	Sc	Ma	Do	Po	Fl	Cu	Ro	In	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03
4	Desktop applications	Co	Do	Pe	Sc	Ma	Fl	Cu	Po	Ro	In	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03
5	Embedded applications	Do	Us	Ma	Ac	Po	Fl	Cu	Ic	Ro	In	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03
6	Tool based applications	Ma	Fl	Sc	Do	Se	Ac	Po	Ic	Ro	In	Se
		.03	.03	.03	.03	.03	.03	.03	.03	.03	.03	.03

Journal of Theoretical and Applied Information Technology
15th December 2012. Vol. 46 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

After forming the tables for the Applications the next thing which needs to be taken into consideration is Operating System (Os). These Os are the interface in between the Application and processor, its job is to manage resource for any System. As the work environment for every System may vary depending the processor or utility support or any other aspect, more than one types of Operating System are available so as many as possible are needed to be contemplated. Already some of the Operating Systems are specified in the table 3.These Operating System are the once against which the QoS will be mapped similar to that as in the Application case i.e. the QoS will be distribute into the pools of High Peer and Low Peer depending on the position of that QoS in the priority stack of the QoS for certain Operating System, also the ambiguous ones will be handled in the chronological manner. After all the Qos are arranged and ranked, values are awarded to each one of them in order to come up with the similar tables. Similarly Tables for the Os and the QoS are also prepared. Thus we came up with Table-3 and Table-4 for High Peer and Low Peer QoS respectively.

Table-3: High Peer QoS for various OS

S No	OS	High Peer					
1.	Windows XP	Us	Pe	Re	Sc	Co	Se
		.30	.14	.10	.07	.06	.05
2.	Windows Vista	Pe	Sc	Re	Co	Us	Se
		.30	.14	.10	.07	.06	.05
3.	Apple MacOS X	Se	Co	Re	Pe	Us	Sc
		.30	.14	.10	07	.06	.05
4.	Linux	Se	Pe	Re	Co	Sc	Us
		.30	.14	.10	.07	.06	.05
5.	Novel Netware	Re	Se	Pe	Sc	Co	Us
		.30	.14	.10	.07	.06	.05
6.	Unix	Re	Pe	Sc	Se	Co	Us
		.30	.14	.10	.07	.06	.05
7.	Sun Solaris	Sc	Re	Pe	Co	Se	Us
		.30	.14	.10	.07	.06	.05

Table-4: Low Peer QoS for various OS

S No	OS	High Peer									
1.	Windows XP	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
2.	Windows Vista	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
3.	Apple MacOS X	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
4.	Linux	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
5.	Novel Netware	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
6.	Unix	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03
7.	Sun Solaris	Ae	Pe	Do	Cu	Ma	Ie	Ro	Fl	In	Se
		.03	.03	.03	03	03	03	03	03	03	03

After the Tables are prepared, these QoS are mapped for the combinations of Applications and OS. And final tables were made for any particular combination. This step uses a mapping function for finding the limits for High Peer and Low Peer differentiation. These functions are as follows:

$f_{high}=\{x: w_1(x)+w_2(x)>0.06, \forall w_1 \varepsilon App, \forall w_2 \varepsilon OS\}$ (4)

w_1= Any entity available in the set of Applications.
w_2= Any entity available in the set of OS.
x= Any of the QoS taken into consideration.

From the equation (4) the value 0.06 is fixed as a threshold value for separating High Peer and Low Peer. This threshold is taken according to the developer team and the perception of the author.

f_{high} was used to determine the high peer were as f_{low} was used for the low peer.

$f_{low}=\{x:w_1(x)+w_2(x)<=0.06, \forall w_1 \varepsilon App, \forall w_2 \varepsilon OS\}$

These functions produce the set of QoS belonging to the High Peer and Low Peer for the particular Application and OS combination; in this case Web based application and Windows XP. The set formed from the f_{high} is

$=\{0.60,0.29,0.20,0.14,0.12,0.08\}$

$=\{w_1(\text{Usability})+w_2(\text{Usability}), w_1(\text{Performance})+w_2(\text{Performance}), w_1(\text{Reliability})+w_2(\text{Reliability}), w_1(\text{Scalability})+w_2(\text{Scalability}), w_1(\text{Consistency})+w_2(\text{Consistency}), w_1(\text{Security})+w_2(\text{Security})\}$

={Usability, Performance, Reliability, Scalability, Consistency, and Security}

Thus the set of the High Peer QoS for the combination is formed and represented in Table-5. Similarly the set for the f_{low} is also derived in Table-6.

$f_{low}=\{0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.06, 0.06\}$

$=\{w_1(\text{Accuracy})+w_2(\text{Accuracy}), w_1(\text{Portability})+w_2(\text{Portability}), w_1(\text{Documentation})+w_2(\text{Documentation}), w_1(\text{Customization})+w2(\text{Customization}), w_1(\text{Maintenance})+w_2(\text{Maintenance}), w_1(\text{Interface complex ability})+w_2(\text{Interface complex ability}), w_1(\text{Robustness})+w_2(\text{Robustness}), w_1(\text{Flexibility})+w_2(\text{Flexibility}), w_1(\text{Interoperability})+w_2(\text{Interoperability}), w_1(\text{Semantic})+w_2(\text{Semantic})\}$

={Accuracy, Portability, Documentation, Customization, Maintenance, Interface coplex ability, Robustness, Flexibility , Interoperability, Semantic}

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table-5 Combination of Web based Application and Windows XP

High Peer						
QoS	Us	Po	Ro	Sc	Co	So
	.60	.29	.20	.14	.12	.03

Table-6: Combination of Web based Application and Windows XP

Low Power										
QoS	Ac	Po	Do	Cu	Ma	Ic	Ro	Fl	In	Sc
	.05	.06	.06	.06	.06	.06	.06	.06	.06	.06

From the above functions and table the graph is derived explaining the effectiveness of the QoS on the combination of the Application and OS, hence providing with the efficient QoS. The component handling High Peer QoS will definitely be the efficient, compatible component.

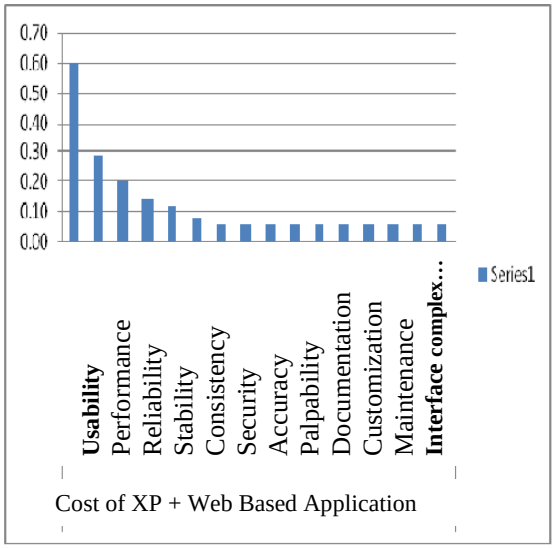

Chart 1: Ranking QoS for Web Based Application under Windows XP

In the above chart 1, X-axis is QoS and Y-axis for performance level. Further the same methodology is used for defining the combinations, which may use the particular QoS as the defining factor for the efficient compatible component. The function used is

f={1: w1(x)+w2(x)>0.10 ,$\forall w_1 \varepsilon$App, $\forall w_2 \varepsilon$ OS and $\forall x \varepsilon$ QoS}

The vCost for XP + Web Based Application

w1 = any entity from the Application set.
w2 = any entity from the OS set.
x = any of the QoS proposed.

The value of 0.10 is found from the summation of values for High Peer and Low peer, thus setting a separating value for the set of QoS supporting high efficiency. This is also according to the developer team and perception of the author. Hence it came up with the set of matrices for each QoS, here 1and 0 of them are taken into consideration. Following are the formed matrices: The symbol of the columns is clearly mentioned in equations (2) and (3).

Pe	X	V	M	L	N	U	S
W	1	1	1	1	1	1	1
N	0	1	0	0	0	0	0
E	0	1	0	0	0	0	0
D	0	1	0	0	0	0	0
Em	1	1	0	1	0	1	0
T	1	1	0	1	0	1	0

Re	X	V	M	L	N	U	S
W	0	0	0	0	1	1	0
N	1	1	1	1	1	1	1
E	0	0	0	0	1	1	0
D	0	0	0	0	1	1	0
Em	0	0	0	0	1	1	0
T	1	1	1	1	1	1	1

Co	X	V	M	L	N	U	S
W	0	1	0	0	0	0	0
N	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
Em	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0

Sc	X	V	M	L	N	U	S
W	0	0	0	0	0	0	1
N	0	0	0	0	0	0	1
E	0	0	0	0	0	0	1
D	0	0	0	0	0	0	1
Em	0	0	0	0	0	0	1
T	0	0	0	0	0	0	1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Se	X	V	M	L	N	U	S
W	0	0	1	1	0	0	0
N		0	1	1	1	0	0
E	0	0	1	1	0	0	0
D	0	0	1	1	0	0	0
Em	1	1	1	1	1	1	1
T	0	0	1	1	0	0	0

Cu	X	V	M	L	N	U	S
W	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
Em	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0

Po	X	V	M	L	N	U	S
W	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
Em	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0

W	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
Em	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0

Ac	X	V	M	L	N	U	S
W	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0
E	1	1	1	1	1	1	1
D	0	0	0	0	0	0	0
Em	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0

Us	X	V	M	L	N	U	S
W	1	0	0	0	0	0	0
N	1	0	0	0	0	0	0
E	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0
Em	1	0	0	0	0	0	0
T	1	0	0	0	0	0	0

Through these matrices the QoS supporting the efficient compatible component can be determined among the various component used in any software. Finding such component may lead to a smart optimization of the application used on any OS described. This may also help in a reverse manner, by finding the least efficient component i.e. the components using all the Low Peer QoS. By finding these components, identifying the limitation of any functionality will be easier, thus providing an idea of which component should be taken in consideration while resolving the efficiency of the Application on any given Operating System. Here, the same concept is implemented using the set of components than can be used for various Applications and OS. Let us consider the available components as C1, C2, C3, C4….., C10, where each component is having a set of Qos which provide high results e.g.

C1={ Pe, Co, Fl, Do}, C2={Se, Ac, Us, Po}, C3={Ic, Co, Ma, Se}, C4={Ro, Pe, Us, Ma}, C5={In, Cu, Co, Se}, C6={Sc, Re, Fl, Do}, C7={Co, Ic, Ma, In}, C8={Pe, Se, Sc, Ac}, C9={Re, Po, Do, Us}, C10={Co, Se, Ro, Ma}.

Table 5: Result of the example considered

	Windows Xp	Windows Vista	Apple Mac OS X	Linux	Novel Netware	Unix
Web based applications	C4,C1,C2, C8	C1,C3,C4, C5,C7,C8, C10	C8,C1 C2,C4,C5	C8,C1 C2,C4,C5	C1,C4,C6 C8,C9	C1,C4,C6 C8,C9
Network applications	C2,C4,C6, C9	C1,C2,C4, C8,C9	C2,C5,C6, C8,C9	C2,C5,C6, C8,C9	C2,C5,C6, C8,C9	C6,C9
Expert System applications	C2,C4,C8, C9	C1,C2,C4, C8	C2,C5,C8	C2,C5,C8	C2,C5,C6 C8,C9	C2,C6,C8 ,C9
Desktop applications	C2,C4,C9	C1,C4,C8	C2,C5,C8	C2,C5,C8	C6,C9	C6,C9
Embedded applications	C1,C2,C4, C5,C8,C9	C1,C2,C4, C5,C8	C2,C5,C8	C1,C2,C4, C5,C8	C2,C5,C6, C8,C9	C1,C2,C4, C5,C6,C8, C9
Tool based applications	C1,C2,C4, C6,C8,C9	C1,C2,C4, C8,C9	C6,C2,C5, C8,C9	C6,C2,C5, C8,C9,C1 C4	C6,C9	C1,C6,C8, C9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1 Results & 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, resulting into an efficient system. Similarly, the QoS lying in Low peer for 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.

4. CONCLUSION

From the Table 5 a set of components were chosen for any combination of OS and Application. This is used for selecting the preferable components among a given set of valid components. These components are the ones which are having any of the QoS in the High Peer for that particular combination of OS and Application. Hence they are capable of providing an efficient system.

REFRENCES:

[1] Lamia Yessad and Zizette Boufaida,"A QoS Ontology-based Component Selection", International Journal on Soft Computing (IJSC), Vol.2, No.3, August 2011,pp 16-30.

[2] Daniel A. Menasce, Honglei Ruan and Hassan Gomaa," A Framework for QoS-Aware Software Components",*WOSP'04*, 2004,pp 186 – 196

[3] Girish J. Brahnmath, Rajeev R. Raje, Andrew M. Olson, Mikhail Auguston, Barrett R. Bryant and Carol C. Burt, "A Quality of Service Catalog for Software Components", The Proceedings of the Southeastern Software Engineering Conference, Huntsville, Alabama, 2002,pp , 441-452

[4] M.Thirumaran , P.Dhavachelvan , S.Abarna and G.Aranganayagi,,"Architecture for Evaluating Web Service QoS Parameters using Agents", International Journal of Computer Applications (0975 – 8887) ,Vol. 10– No.4, November 2010, pp 15-21

[5] M. Sathya, M. Swarnamugi , P. Dhavachelvan and G. Sureshkumar ,"Evaluation of QoS Based Web-Service Selection Techniques for Service Composition", International Journal of Software Engineering (IJSE), Vol. (1): Issue (5), 2011,pp 73-90

[6] Q. Sun, S. Wang, H. Zou& F. Yang ," QSSA: A QoS-aware Service Selection Approach", International Journal of Web and Grid Services, Vol. 7, No 2, May 2011, , pp 147-169

[7] L. Yessad, & Z. Boufaida ,"QoS-based Component Selection using Semantic Web Technologies", In Proceedings of ICWIT'10, 231-240, June 2010, pp 16-19

[8] E. Giallonardo& E. Zimeo , "More Semantics in QoS Matching " , Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07), 2007, pp 163-171

[9] Matthias Klusch, Patrick Kapahnke, "Semantic Web Service Selection with SAWSDL-MX" , German Research Center for Artificial Intelligence, Vol.: 416, 2008, pp 3-16

[10] Roy Grønmo, Michael C. Jaeger."Model-Driven Methodology for Building QoS-Optimized Web Service Compositions", In Proceedings of the fifth IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'05), Springer Press, May 2005, pp 68–82

[11] VuongXuan Tran, HidekazuTsuji ,Ryosuke Masuda." A new QoS ontology and its QoSbased ranking algorithm for Web services", Journal on Simulation Modelling Practice and Theory,ScienceDirect, Vol. 17, Issue 8, September 2009, pp 1378-1398

[12] Mohammad Alrifai, Thomas Risse," Combining global optimization with local selection for efficientQoS-aware service composition", In Proceedings of the 18th international conference on World Wide Web, ACM, ISBN: 978-1-60558-487-4, 2009, pp 881- 890

[13] J.F. Tang, L.F. Mu, C.K. Kwong , X.G. Luo ,"An optimization model for software component selection under multiple applications development", European Journal of Operational Research, Vol. 212, Issue 2, July 2011, pp 301–311

[14] Marko Palviainen, Antti Evesti, Eila Ovaska, "The reliability estimation, prediction and measuring of component-based software", The Journal of Systems and Software, 2011, pp 1054–1070

[15] C.K. Kwong, L.F. Mu, J.F. Tang, X.G. Luo," Optimization of software components selection for component-based software system development", Computers & Industrial Engineering, 2010, pp 618–624