I J C T A, 9(37) 2016, pp. 889-896
© International Science Press

Self Directed Web Service Composition: A Platform

S. Ganesh Kumar* **and K. Vivekanandan****

Abstract: The expanding intricacy of business procedures requires enhanced techniques for web administrations. In request to satisfy this, it is troublesome for administrators to keep up with the developing interest and the huge measure of customizations required by the clients. A conceivable arrangement that will help in this circumstance is to create techniques and innovations that bolster independent structures of web administrations. Such arrangements ought to adjust powerfully to changes in the necessities or the earth. This paper portrays a stage which actualizes such arrangement, in view of Quality of Service (QoS). The stage model, which is introduced here, can screen web benefit QoS and figure out if the administration organization satisfies the in general quality required by the end users.

Keywords: Quality of Service, Service level agreement, Web administrations, Compositions, SOA.

1. INTRODUCTION

Service Oriented Architectures (SOA) play a vital part in empowering joining of business with IT. Administrations are a key idea in SOA and they speak to reusable substances that ought to minimize the improvement exertion and give implies for data trade for both administration shoppers also, suppliers. Then again, the multifaceted nature of business issues is expanding and to explain them, clients could utilize various administrations into a structure to execute a business handle. In any case, as business gets increasingly adaptable today, customers require extra usefulness what's more, customizations towards the administrations they utilize. In other words, a static structure [15] is not competent to satisfy all client necessities in a long haul point of view. This makes benefit suppliers look for approaches to manage the expanding number of administration request while in the meantime giving customized Service Level Agreement (SLA) administration [6]. An answer for this issue is to give compositions that are autonomic and are proficient to adjust to changes in client necessities or nature. Such arrangements can adjust as indicated by some quantifiable guidelines. Give us a chance to consider that, for a creation, one ought to pick an administration out of a set of administrations that have comparable usefulness. To comprehend that, it is conceivable to pick an administration that offers the best [8][10] Quality of Service (QoS). In addition, it would be better if the sythesis is not static but rather changes powerfully agreeing to changes in the QoS of administrations (in view of changes in workload, number of solicitations, and so forth) or in client necessities. For instance, if more clients send solicitations to an administration, its reaction time may raise to an undesirable level, and after that another administration ought to be found and coordinated into the organization. The objective is to make this with insignificant human intercession and actualize the change progressively and straightforwardly for the client. This paper displays a stage for building self-governing web benefit organizations in light of QoS. The stage gives intends to social event information for assessment [20] of administration QoS qualities (like execution, accessibility, unwavering quality, cost, and so forth. Such means include:

- An augmented administration registry, which is utilized as a storehouse for accumulation of administration QoS information and empowers simple [27] web-administrations inquiry and determination
- A model to figure benefit QoS, as indicated by the information in the registry

* Assistant Professor, Department of Computer Science & Engineering, SRM University, Chennai. *Email: sganeshkumar78@gmail.com*
** Professor, Department of Computer Science & Engineering, Pondicherry Engineering College, Pondicherry. *Email: kvivek27@yahoo.com*

- A calculation to discover and select the administrations [25] that will best meet the concurred SLA of the structure
- Automatically and straightforwardly incorporate chose web administrations into a working structure

A key part of the proposed stage is that [13][18] administration organizations are resolved and upgraded progressively at runtime. This liberates chairmen and engineers from executing any QoS related changes.

Whatever is left of this paper is sorted out as takes after: Section 2 makes a review of the related work; Section 3 presents the model that we use for assessing the nature of web benefit structures and deciding the best arrangement; Section 4 presents the plan and usage of our stage for self-ruling web benefit arrangement. Section 5 presents a basic contextual analysis to represent utilization of the stage and approve it, lastly, Section 6 finishes up the paper and states headings for further research.

2. RELATED WORKS

Quality attributes are very important in terms of design and reasoning about of software systems. They are regarded as key concerns in software architecture design and selection of relevant web services [9]. Many researchers have also managed to solve the problem with formal definition and management of software quality in general. For example, there exist a lot of theoretical models[14] for evaluation of reliability, performance complexity, etc. However, such models tend to be relevant only at theoretical level as they are either quite general and have some unrealistic assumptions [26] that make them inapplicable in practice or too complex to be applied in abroad range of domain areas. The work that relates to ours may be split into two main directions: first one is related to models for calculation of [4][22] software QoS and second one – to methods for dynamic and autonomous web service composition. Many research efforts combine the two directions and use the overall QoS of a service composition to determine whether it should change or not. There are a lot of models available that consider the quality characteristics and based on them provide the best service composition. Liu et al.] proposed of QoS model that is open and extensible [19]. They provide an implementation of a QoS registry that stores the web services QoS data and allows consumers to search against it. A key point here is that such data is obtained through user feedback, i.e., consumers that use the services rate them and provide their feedback to the QoS registry. Ran. proposed a model for web service [17] discovery based on QoS. They argue that current web service registries limit service discovery to functional requirements only and non-functional properties should be paid more and more attention. They extend the current web services registration and discovery model by introducing a new role – Web Service QoS Certifier. The concept of a certifier is also covered in. The certifier is responsible to certify/verify the claimed non-functional properties of the web service providers. The DYSCO platform provides a complex solution for dealing with dynamic web service composition. The platform allows [24] automatic generation of executable business processes and SLA for each web service. It also provides mechanisms for monitoring the used web services and updating the business process when SLA deviations are discovered. AgFlow is a middle ware platform that allows quality-driven dynamic web services composition. The platform provides a [23] multidimensional QoS model that is responsible for capturing the non-functional propertie [9][21]s of the web services. This work introduces two approaches for selecting web services – local optimization and global planning. An adaptive execution engine is responsible for the runtime adaptation of the web services composition. It replans the execution any time when any of the services is unavailable or the quality properties exceed predefined thresholds. The web service composition algorithm proposed by Lu et al. is based on seven QoS properties – running cost, runtime, success ratio, usability, trustworthiness, degree of security and degree of semantic correlation. A limitation of the [12] algorithm is that only semantic (immeasurable) QoS properties are considered. In addition, it is not clear how the QoS properties are [16] assessed. In contrast, Yu et al. rely on measurable QoS properties and especially on latency, execution cost, availability and accuracy. The advantage of the proposed solution is that it is applicable to data intensive web service compositions. It combines the tabu search and the genetic programming techniques. The last one is applied also in the web service composition approach presented in. An approach for self-healing web service composition is introduced by Aziz et al. It repairs the web service composition when some of its components violate the QoS constraints. The headers of the SOAP messages [7] are extended in order to provide information about QoS properties. The approach includes three main phases: monitoring [1][3], diagnosis and repairing. When QoS degradation is detected during diagnosis phase, a repairing procedure is started. As a result the failed web service is replaced with another one obtained from the UDDI registry. A possible drawback of the approach is that it relies on SOAP as communication protocol and is not clear how it could be applied when the composition includes REST web services. The web service composition system presented by Brahmi and Gammoudi is based on cooperative agents. The agents are organized as a social network and cooperate to find the optimal composition with respect of QoS. The approach proposed by Xia and Yang is focused on QoS optimization and redundancy[2] removal. An advantage of the approach is that it removes most of the redundant web services minimizing total execution cost of the composition. Unfortunately, the QoS optimization is based only on two QoS properties – response time and throughput. Birgit and Marchand-Maillet solved the web service composition problem partially by providing an algorithm for QoS-aware selection. The algorithm uses a rank aggregation instead of direct measures of QoS values. Its core includes so called abstract voter that sorts the web services according to a particular QoS property, named QoS factor. The web service composition problem is formalized as problem of [11] traversing a Petri Net. The estimation of composition's quality is performed through utility function that aggregates the functional, QoS and transactional properties of the web services. Currently, there is no universal approach for autonomous management of dynamic web service composition based on QoS. In this work, we propose a platform that deals runtime with QoS monitoring, adaptation [5] and discovery of web services, in order to determine the best possible composition. Another advantage of the approach presented here is that it is compatible with the Business Process Execution Language (BPEL) standard and is capable of implementing an executable composition.

3. SELF DIRECTED WEB SERVICE COMPOSITION

In this section, we introduce the model we use for self-governing web benefit creation. It incorporates breaking down the quality information for each qualified web benefit and deciding the best composition that matches a predefined set of value necessities. The introduced model depends on [16] and [17], yet adds the accompanying extra elements to accomplish the objective:

- Introduces the idea of a web benefit class as a conceptual element that may allude to numerous web administrations, giving the same usefulness and interface.
- Uses weight (i.e., needs) of value credits to decide the heaviness of real client necessities.
- Analyzes all administration structures that might be coordinated so as to locate the one that best matches the client characterized business prepare.

In addition, the model presents the idea of a web benefit class. Every classification is characterized by regular usefulness and an interface and might be connected with various web administrations. Hence, the exhibited demonstrate requires that clients characterize their business forms by determining the web benefit classes as opposed to the solid web benefit usage. As such, every business process is considered as a sythesis of different web benefit classifications. Solid web administrations are relegated after the model is connected and the best structure is known. At present, to our best information, there is no bound together standard for overseeing web benefit classifications. For the reason for this exploration, we have utilized a focal administration registry for finding web benefit classes and related web benefit usage for each of them. The initial phase in our model is deciding the arrangement of web benefit classes that assemble our business procedure. In this case, we are not intrigued by the succession of the specific web benefit summons yet need to know the arrangement of various web benefit classifications like.

The business process is the combination of all the WSC_i, where WSC_n – a webservice for a single category, W_s – the implementation of a particular webservice.

The quality attributes can be calculated using the R and the associated weights W. where R is a set of all r_i and W is a set of all w_i.

Where we can find the efficiency using the set of R and the associated weights. This model permits us to locate the best web benefit structure in light of a predefined set of value necessities and their related weights. It could without much of a stretch be connected to any number of web administrations and stretched out to bolster different sorts of value characteristics. This model is connected in the execution of our self-governing administration creation stage. The specialized execution of the stage is introduced in section 4.

4. A PLATFORM FOR SELF DIRECTED WEB SERVICE COMPOSITION

In this section, we introduce the outline and usage of our stage. It depends on the model portrayed in the past segment and takes after the design exhibited on Figure 1. The model for deciding the best administration sythesis is executed in the BPEL Extension segment. The present stage permits runtime upgrades to the conveyed benefit organizations with no human supervision. Our stage additionally gives an amplified web benefit registry that permits shoppers to hunt down the administrations they require furthermore ask data for their QoS qualities. At last, every one of these information are prepared by a BPEL augmentation apparatus which is a composition of our past work and it permits dynamic official of the chose web benefits in the characterized creation [4]. The stage comprises of the taking after parts:

1. BPEL Extension – expansion conveyed on business handle server permitting to play out the runtime creation of web administrations and acclimate to the quality necessities of every client.
2. Expanded Service Registry – a standard administration registry with a DB expansion for continuing quality characteristics information for the web administrations. Access to this information is uncovered as a major aspect of the registry interface permitting administration buyers to utilize it for their composition examination.
3. Web Service Interceptor – device that can capture any web benefit call and gather the required quality characteristics information. This information is then put away in the amplified benefit registry and made accessible of other administration buyers.

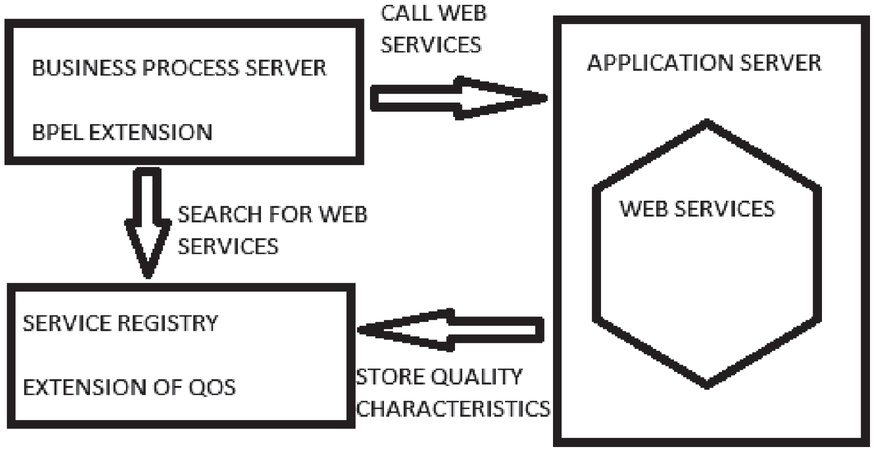

Figure 1: Architecture Platform For Expandable Web Services

In the following subsections, we give a point by point portrayal for each of the stage segments.

A. BPEL Extension: Our BPEL augmentation permits overhauling a BPEL procedure at runtime. It is produced by BPEL expansion determination and can be sent and connected to any BPEL agreeable server. In this work, the WSO2 BPS server is used to test the expansion.

B. Expanded Service Registry: The expanded administration registry gives the standard UDDI (Universal Description, Discovery and Integration) interface. Officially existing Apache jUDDI v.3.0.4 registry is utilized for this reason. All web benefits that the stage can work with are enlisted there.

   As expressed for the sake of the segment it gives developed usefulness. We have conveyed a database that stores the quality attributes for every web benefit conjuring. This information is put away in crude organization with the goal that it can be utilized with different models. To make this information open we have created Apache jUDDI-like administrations, with the goal that customers could get the quality information they requirement for building their administration structure. Those web administrations are uncovered as SOAP administrations. So as to make the expansion as inexactly coupled to the UDDI registry we have actualized it as a different apparatus that end clients could incorporate with. This approach permits benefit purchasers to utilize the UDDI registry standardly and just the individuals who are keen on the QoS information could trigger the important inquiries against the augmentation. What's more, our expansion knows about the administration classes. A key point here is that we attempt to evade the idea of utilizing a web benefit QoS certifier. We would not let benefit suppliers distribute any QoS information for their web administrations. Or maybe, we would expect each supplier that is keen on giving such information to introduce the supposed web benefit interceptors that we give. They will store the applicable information in the expanded registry. Along these lines the interceptors act like a Confirmed web benefit QoS information supplier.

C. Web Service Interceptor: The web benefit interceptor is a module, in charge of gathering quality related information for every web benefit summon. This module is sent on every server that has the executed web administrations. Since there are different innovations for executing and uncovering web administrations we have constrained ourselves to utilizing the Apache Axis 2 system. It gives components to extensibility and we could without much of a stretch incorporate our custom rationale. In addition the Apache system plan incorporates instruments for creating custom handlers for the bolstered web administrations. We exploit this usefulness and we have created custom handlers that block the web benefit summons. There are two sorts of handlers – message stream handlers and mistake stream handlers. The message stream handlers prepare the standard web benefit conjuring while the blunder stream ones are enacted when the web benefit comes up short. For every web benefit conjuring we get the accompanying information – SOAP message measure, preparing time and ID information like operation connection ID, IP addresses, and so on. This information is then put away in our database and uncovered for figuring of value information. One of the significant plan objectives for web administrations interceptor module is particularity. Along these lines, it is actualized in a simple to design and alter way. The interceptor module itself is bundled as a module document record. This document is sent on the servlet compartment by making an organizer named "modules" in the "webapps/axis2/WEB-INF" index

5. EXPERIMENTS

So as to demonstrate the advantage of the our Service compositon Platform, this segment presents tests that show how it performs in determination of the best (by QoS) web benefit. The present usage is still a model what's more, extra approval will be made when it develops. The test concentrates on two quality attributes – execution and accessibility. We have set the weight for each for the quality attributes to 0.7 for execution furthermore, 0.3 for accessibility. With the end goal of this investigation, we have characterized three web benefit classifications, each speaking to a scientific operation – Multiply, Power what's more, Add. For every class we have built up an arrangement of three web administrations with the same usefulness and interface however reproducing diverse quality attributes – standard, moderate furthermore, arbitrarily accessible. To make the investigation we made a business procedure that uses the three web benefit classifications. Each of them is called in a steady progression. For this situation, we are not inspired by the last aftereffect of the estimation yet we give careful consideration to the quality attributes of the executed business handle. Figure 2 speaks to the business procedure we use in our tests.

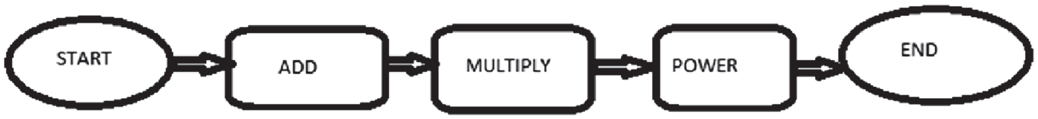

Figure 2: Business Procedure Steps

Our model can be presented as

$$BP = \{WS_{ADD}, WS_{MULTIPLY}, WS_{POWER}\}$$

After many invocations we have collected the quality characteristics of each web service, we have presented the obtained values in the Table 1.

Table 1
Aggregated QOS of the Experimental service

Attribute	*Service*	*Avg. performance*	*Avg. availability*
	slow	0.531	0.9
Add	available	0.24	0.69
	standard	0.234	0.9
	slow	0.49	0.9
Multiply	available	0.241	0.64
	standard	0.238	0.9
	slow	0.535	0.9
Power	available	0.229	0.68
	standard	0.225	0.9

For this situation, there are 27 conceivable creations that could be constructed. In any case, every structure will have distinctive esteem for the whole quality and the stage ought to select the one that has the most astounding score. Figure 2 exhibits a design of the computed values for the nature of arrangements for the conceivable blends. To run this recreation we have set the weights for execution and accessibility to 0.5 and 0.5. The arrangement on the highest point of the illustrations has most astounding score contrasted with the rest. This is the arrangement {Add Standard, Multiply Standard, Power Standard}.

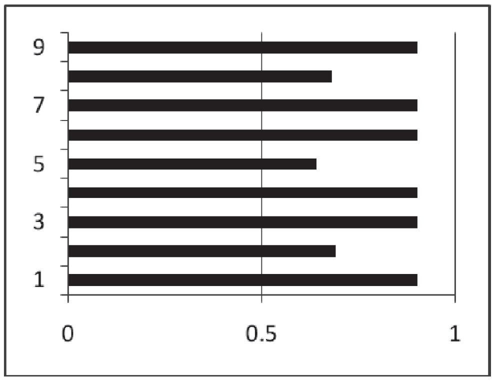

Figure 3: Scores for the composition

At the point when the test began our stage broke down the characterized business prepare and the related quality objectives. In view of the characterized web benefit classes the pertinent benefit executions were found and the last arrangement was set. The web benefits that were chosen as an aftereffect of our investigation QoS information for chosen benefits after test.

The aggregate time for the execution of the business procedure is 0.71 seconds and the accessibility stays 100%. This is the most ideal creation that fits the predefined quality necessities and the related weights for each of them.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a stage for self-governing web benefit structures. This stage is capable to screen the web administrations execution and assemble information for assessment of administration quality attributes. This information is accessible through web benefit registry augmentation. The paper likewise proposes a model for investigation of the quality information and deciding the best administration organization. A key part of this model is the presentation of web administration classification as an theoretical method for characterizing an arrangement of administrations giving the same usefulness and interface. Also, our stage depends on open source programming and is intended for simple extendibility and modifiability. In the long haul such an approach could spare a ton of authoritative work and increment the level of consumer loyalty. The stage gives intends to self-governingly adjusting the running business forms in light of predefined client objectives as far as SLA. Nonetheless, we can express the accompanying bearings for future research, in arrange for the stage to give a completely useful end-toend arrangement:

1. Developing the extent of the web benefit interceptor – as of now, we bolster Apache Axis2 based web benefits yet we plan to create interceptors for other web benefit systems that can be broadened.
2. Broadening the quantity of value properties – at present, we have centered our examination on execution, accessibility and throughput. We consider augmenting the quantity of upheld quality traits inside the interceptors and the expanded web benefit registry.
3. Enhancing the model for selecting best web benefit composition – a powerless point for our model is the choice of the best web benefit structure. It is normal that all conceivable organizations are broke down and after that the best one is chosen. As a state of change, we consider upgrading the choice calculation to work in a more proficient way.
4. Perform point by point approval of the stage – The displayed stage is still a model. As the stage gets more develop, extra approval what's more, examinations ought to be performed.

References

1. F. N. Souza, T. C. Silva, D. J. M. Cavalcanti, N. S. Rosa, and R. M. F. Lima, "*A meta-model for qos monitoring in a dynamic service-component platform*," in Proceedings of the 2015 IEEE International Conference on Services Computing (SCC'15), 2015, pp. 459–466.
2. S. Nepal, C. Paris, and A. Bouguettaya, "*Trusting the social web: issues and challenges*," World Wide Web, Vol. 18, No. 1, pp. 1–7, 2013.
3. F. Naumann, U. Leser, and J.C. Freytag, "*Quality-Driven Integration of Heterogenous Information Systems*," Proc. Int'l Conf. Very Large Databases (VLDB), pp. 447-458, 1999.
4. IBM Optimization Solutions and Library, http://www-3.ibm. com/software/data/bi/osl/index.html, 2003.
5. J. O'Sullivan, D. Edmond, and A.t. Hofstede, "*What's in a Service?*" Distributed and Parallel Databases, Vol. 12, Nos. 2-3, pp. 117-133, Sept. 2002.
6. G. Oulsnam, "*Unravelling Structured Programs*," The Computer J., Vol. 25, No. 3, pp. 379-387, 1982.

7. M. H. Hasan, J. Jaafar, and M. F. Hassan, "*Monitoring web services' quality of service: A literature review*," Artificial Intelligence Review, Vol. 42, No. 4, pp. 835–850, 2014.

8. S. Patil and E. Newcomer, "*ebXML and Web Services*," IEEE Internet Computing, Vol. 7, No. 3, pp. 74-82, May/June 2003.

9. C. Peltz, "*Web Services Orestrestration and Choreography*," Computer, Vol. 36, No. 10, pp. 46-52, Oct. 2003.

10. Y. Y. Luo, J. Long, and J. J. Wen, "*Qos collection for web services based on ws-monitor model*," in Proceedings of the 5th Asia-Pacific Symposium on Internetware, 2013, pp. 1–4.

11. M. Pinedof, Scheduling: Theory, Algorithms, and Systems, second ed. Prentice Hall, 2001.

12. S. Ponnekanti and A. Fox, "*SWORD: A Developer Toolkit for Building Composite Web Services*," Proc. Alternate Tracks of the11th World Wide Web Conf., May 2002.

13. J. M. Zhu, P. J. He, Z. B. Zheng, and M. R. Lyu, "*A privacy preserving qos prediction framework for web service recommendation*," in Proceedings of the 2015 IEEE International Conference on Web Services (ICWS'15), 2015, pp. 241–248.

14. B. Raman, S. Agarwal, Y. Chen, M. Caesar, W. Cui, P. Johansson,K. Lai, T. Lavian, S. Machiraju, Z. Morley-Mao, G. Porter, T. Roscoe, M. Seshadri, J.S. Shih, K. Sklower, L. Subramanian, T. Suzuki, S. Zhuang, A.D. Joseph, R.H. Katz, and I. Stoica, "*The SAHARA Model for Service Composition Across Multiple Providers*," Proc. First Int'l Conf. Pervasive Computing, pp. 1-14, May 2002.

15. B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing Web serviceson the Semantic Web. The VLDB Journal, 12(4), November 2003.

16. S. Narayanan and S. McIlraith. Simulation, verification and automated composition of Web service. In Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA, May 2002. ACM. Presentation available athttp://www2002.org/presentations/narayanan.pdf.

17. S. R. Ponnekanti and A. Fox. SWORD: A developer toolkit for Web service composition. In Proceedings of the 11th World Wide Web Conference, Honolulu, HI,USA, 2002.

18. J. Rao, P. K¨ungas, and M. Matskin. Application of Linear Logic to Web service composition. In Proceedings of the 1st International Conference on Web Services, Las Vegas, USA, June 2003.

19. J. Rao, P. Küngas, and M. Matskin. Logic-based Web services composition: from service description to process model. In Proceedings of the 2004 International Conference on Web Services, San Diego, USA, July 2004. IEEE.

20. M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic web services reliability for qos-aware recommendation," IEEE Transactions on Services Computing, vol. 8, no. 3, pp. 425– 438, 2015.

21. H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

22. E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of Web services using semantic descriptions. In Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003, 2002.

23. R. Waldinger. Web agents cooperating deductively. In Proceedings of FAABS 2000, Greenbelt, MD, USA, April 5–7, 2000, volume 1871 of Lecture Notes in Computer Science, pages 250–262. Springer-Verlag, 2001.

24. X. M. Liu, A. Kale, J. Wasani, C. Ding, and Q. Yu, "*Extracting,ranking, and evaluating quality features of web services through user review sentiment analysis*," in Proceedings of the 2015 IEEE International Conference on Web Services (ICWS'15), 2015, pp. 153–160.

25. S. G. Wang, Z. B. Zheng, Z. P. Wu, M. R. Lyu, and F. C. Yang, "*Reputation measurement and malicious feedback rating prevention in web service recommendation systems*," IEEE Transactions on Services Computing, Vol. 8, No. 5, pp. 755–767, 2015.

26. H. Ludwig, "*Web services qos: External slas and internal policies or: How do we deliver what we promise*?" in Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops, 2003, pp. 115–120.

27. Z. B. Zheng, Y. L. Zhang, and M. R. Lyu, "*Investigating qos of real-world web services*," IEEE Transactions on Services Computing, Vol. 7, No. 1, pp. 32–39, 2014.