A survey for service selection approaches in dynamic environments

[1]*Lindelweyizizwe Manqele,* [2]*Mqhele Dlodlo,* [3]*Louis Coetzee and* [4]*George Sibiya*

[1,2] University of Cape Town, Private Bag X3, Rondebosch, 7701, South Africa

[1, 3, 4] CSIR, Meraka Institute, P O Box 395, Pretoria, 0001, South Africa

[1, 3, 4] {lmanqele, lcoetzee1, gsibiya}@csir.co.za

[2]mqhele.dlodlo@uct.ac.za

Abstract

The usage of the service selection approaches across different dynamic service provisioning environments has increased the challenges associated with an effective method that can be used to select a relevant service. The use of service selection approach should depend on certain factors. In order to address this challenge, the literature analysis is conducted on various service selection approaches. A proposed approach needs to be tested by manipulating the relevant services description of available services. This paper, proposes various aspects that needs to be considered when choosing a method of service selection. The aspects were used in a scenario to select the effective method and later the method was evaluated based on response time, recall and precision metrics. The experiments showed that the approach works better based on the results yielded among the technical algorithms applied on the approach. The content-based algorithm returned more relevant services to the user and took shorter time as compared to the collaborative filtering.

***Keywords*— service selection, Internet of Things, cloud computing, big data and dynamic environments**

I. INTRODUCTION

The dynamic environment is the environments that keeps changing. The influence of change includes time, location, status, and other factors. The dynamic environments involves an increasing number of smart interconnected devices and sensors (e.g. cameras, biometric, smart meter, and medical sensors) for a smart world [2], these interconnected devices and sensors are referred as "Things". The dynamic environment considered in this study is the emerging technology of the Internet of Things (IoT). IoT appears to be the anticipated technology occurrence of machine-to-machine or machine-to-human communication over the Internet that will influence the future [3]. The environment like IoT is not dependent on developing new technology but on connecting and integrating existing technologies. Peer-to-peer communications among devices will push services down to the device layer that implements "Things" and create new opportunities for functionality like discovery and selection [5]. In order to select a relevant service, the user needs an effective approach that suits the environment. The approach should be able to gather all the input required to match the service with the requirement(s). This study, therefore, classifies the different approaches used in the dynamic environments and later suggests the important factors that should be considered to choose the approach that will work better for any defined dynamic environment.

II. BACKGROUND

In order to appreciate the significance of this paper and the study more generally, it is necessary to put the discussion of service selection in context. To do this, it is necessary to draw from service-oriented computing (SOC) literature because this lays the foundation for the topic of service selection. Service Oriented Computing is discussed next. Service-oriented computing is defined as a computing paradigm that utilises services as fundamental elements for developing business solutions and applications [7].The set of concepts, principles and methods that represent computing in service-oriented architecture (SOA), in which software applications are constructed based on independent component services with standard interfaces, is referred to SOC [6]. SOC supports the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous and highly dynamic environments. The emergence of the SOC paradigm promises to enable businesses and organisations to collaborate in an unprecedented manner by means of technologies such as web services [14]. Kuropka, Laures& Troger [8] captured the provisioning of web services by means of a service-delivery life cycle which is accomplished through three sub-cycles, namely the planning, binding and enactment sub-cycles. The service-delivery life cycle presents a good view of what is necessary in order to provide electronic services of any kind [8].

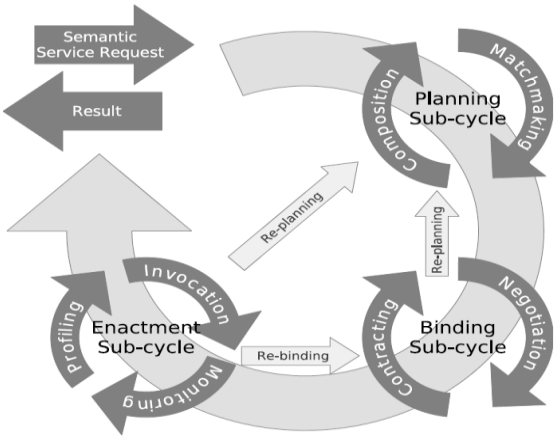

Figure 1: Service delivery lifecycle [8]

When a user makes a request, the first step that a service-providing platform takes is to *find* a service that can satisfy the incoming request. If a requested service is not readily available, the service space (e.g. a smart campus) searches for those services that can satisfy some aspects of the service request until the request can be satisfied in its entirety. This process is called planning and consists of two main activities: service selection or matchmaking, and service composition. When a service is found that can satisfy the entire request or some aspect of it, such a service is selected for invocation.

This is service selection. In the case where multiple services are needed to satisfy the request, when each of those services

is found and selected dynamically or manually, this is the process of service composition and such services are selected for composition. Service composition becomes important when a complex request from the client cannot be satisfied by a single service. The focus of this chapter, and the study as a whole, is service selection that, according Figure 1, is done during the planning sub-cycle of the service-delivery lifecycle.

When a service is found, the service provision platform needs to bind to this service – this entails contracting on some of the non-functional requirements of the request and, negotiating if such requirements cannot be readily satisfied. After binding with a service, the bound service can now be enacted. Enactment of a service also involves invoking the service, profiling its execution and monitoring it. When errors occur, the service-provision platform may resort to re-binding or, if the error is severe, a complete re-assembling of a service is done by returning to the planning sub-cycle. In the preceding description of the steps necessary to provide a service, no particular mention was made of the nature of services being provided. However, as the focus of this chapter is service selection, it is necessary to describe the nature of services. The services that are the subjects of selection are IoT services, which are discussed next.

III. LITERATURE

In the IoT domain, any software component that enables controlling of an IoT device or provides information on an entity or enables controlling of an IoT device is called a "resource" [9][34]. A well-defined and standardised interface, offering all necessary functionalities for interacting with entities and related processes is called a "service" [34]. An IoT service is modelled as a virtual concept that is exposed by an IoT resource. IoT services delivery devices mostly have limited computation capabilities, their exposed resources operate in dynamic environments, and they are less reliable than the general web service [11]. The model in figure 2 is created based on the SOAP/WSDL and RESTful service technologies [9].

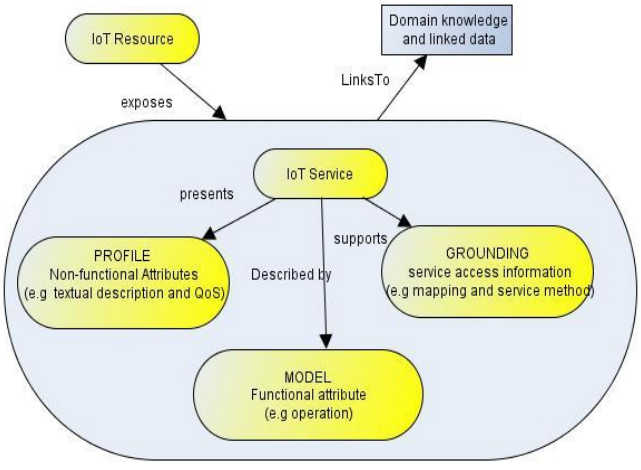

Figure 2: Overview of the IoT description model [9]

The SOAP/WSDL-based services have associations with business process modelling and have been adopted in the business world, while RESTful style services are data-centric and have been prevalent in web 2.0 applications due to their flexibility and simplicity [31]. Figure 2 gives an overview of the IoT description model. In this figure, the profile of a service defines the non-functional aspects of the service and it contains properties for linking to semantic concepts in the existing knowledge bases or taxonomies which are essential for service search and discovery. Barnaghi *et al*. [9] proposed the OWL-S ontology as a semantic model for SOAP/WSDL services, and OWL-S is designed based on the so-called "Profile-Process-Grounding" pattern. In the literature, ontology is presented as a model-based approach, therefore the approach adopted by Wang [10] solves the complexity that originates from process modelling. In contrast, RESTful services is a simple service ontology that excludes the profile and grounding modelling which is important for service discovery and access.

The model was developed by identifying and analysing commonalities between different service technologies. It represents a trade-off between the SOAP/WSDL-based and RESTful services. The concept of grounding provides a mapping between the concepts defined in the semantic description ontology and those defined in the service documents such as WSDL (for SOAP-based services). The mapping concept is optional in IoT services, which usually do not present service documents. The next section presents existing service selection approaches in dynamic environments.

EXISTING SERVICE SELECTION APPROACHES

Existing approaches in helping a user to compare and select infrastructure for services in cloud computing involve manually reading the provider documentation to find out which services are most suitable for hosting an application. Therefore, selecting and composing the right services with which to meet the application's requirements is still a challenge. Web services are typically highly configurable and a service requester often has dynamic preferences for service configuration. Services composed need to be planned in an optimised way. There may be no single web service that directly offers the desired functionality. A combination of web services may need less investment or capabilities than a single service. In most cases, providers compose web services in order to offer the composite service as a new web service. The service has to be selected in an environment consisting of multiple functionally equivalent operators, but with possibly different implementations and time-varying resources [35]. Yu and Marganeic [1] presented a survey and a classification of the service selection approaches. A survey and a classification includes: firstly, Model for non-functional properties where a service requestor needs to objectively distinguish services based on his non-functional criteria to get the most appropriate choice amongst a number of available services with equal or similar functionality. A non-functional properties model is required, it can be used in service descriptions as well as in service requests [3]. Secondly, is the hierarchical properties, this approach is meaningful to place properties into a hierarchical structure. This approach allows, for example, to gather properties and order them by domains and by broader aspects such as performance or safety [1]. The speed and the quality properties are the performance aspects while security and privacy are a safety aspects. Lastly, user preference approach. In this approach, service requestors usually have varying preferences for the non-functional criteria depending

on the situation they find themselves in, and of course different requesters will have different preferences [4]. Jembere *et al*. [27] Yu and Marganeice [1] outlined that personalisation could be any information that can be used to adapt the interaction of a user with a system or service to the needs and preferences of the user or user group. Personalisation of a user's information must be well defined and made available for the context-aware system. Current service selection approaches are as follows:

A. Service selection based on the multi-agent approach

This approach, as proposed by Maximilien and Singh [5], uses the ontology for Quality of Service (QoS) and a new model of trust. The approach gathers and shares a service based on ratings; even its user preferences are based on ratings. This approach assumes that the system should give an empirical basis for the service for selection. These ratings are made to be quality-specific through monitoring and user input. Agents used in this approach show that they are able to dynamically adjust their trust assignment and select a service that best satisfies a user. The challenge associated with this approach is that a user is forced to make an ad hoc decision about the service requested. The selection is based on how a given service behaved previously (as deduced from ratings). However, this approach depends on users sharing their experiences. The service that performed poorly previously should be given a chance to improve in the next request. Another challenge identified in this approach according to the focus of this research is the design of a system that would be able to handle user preferences in a machine-readable format such that cloud work flow engines will be able to process them, regardless of whether the preferences are explicitly given by the user or leant from user-session data.

B. Service selection based on ontology

This approach is based on semantic matching for each service. Semantic matching focuses on meanings behind every service comprising the repository. Services may differ in their syntax but if they serve the same purpose or functionality, they will likely be recommended. Some ontologies use models and those models are supported by QoS properties like name, category, data type, relationship, priority, dynamic attributes and other properties. Maximilien and Singh [5] proposed QoS ontology that let service agents match advertised quality levels for its consumers with specific QoS preferences. Much such ontology has been proposed in the literature. DAML-QoS complements DAML-S by providing a better QoS metrics model [6]. This approach presents a matchmaking algorithm for QoS property constraints and describes different matching degrees. OWL-Q ontology is proposed by Kyriakos & Dimitris [13], an approach that addresses a challenge of web-service registries returning many functionally equivalent web services advertised for each user request. QoS is used for distinguishing between functionally equivalent web services. Discovery algorithms for QoS-based web services fail to yield accurate results because they rely on either syntactic or semantically poor QoS metric descriptions. Hence, these discovery algorithms cannot infer the equivalence of QoS metrics based on descriptions provided by different parties. This approach does not incorporate evaluation of metric matching in order to show their performance and accuracy. Damiano, Giallonardo & Zimeo [7] proposed a query language that is used to define complex constrains and it is called onQoS, which captures QoS requirements. The language that was used before was based on SPARQL, whereas onQoS is able to select services based on QoS requirements and specifications. This approach focuses only on improving quality of service language, and presents how the user defines themselves on semantic webs. According to Tran, Tsuji and Masuda [12], there is a need to distinguish and rank web services that have similar functionalities. QoS has been used as an important factor in distinguishing the quality of web services. Tran [12] proposed a web-service QoS-based ontology (WS-QoSOnto), this approach supports in describing QoS information in detail and facilitating different service participants expressing their QoS offers and demands at various levels of expectation. This approach needs to develop a ranking algorithm for web services, basing it on the QoS description specified by the QoS ontology. The challenge identified according to the focus of this research, is the need to find a mechanism to query for inferred preferences from preference repositories and a personalised selection of services both functional and non-functional requirements.

C. Service selection based on Quality of Service

SOA enables a multitude of service providers to provide loosely coupled and interoperable services at various QoS and cost levels in a number of service domains. The QoS-based mechanism is a non-functional-based service selection approach that evolves trust computing and market-oriented computing development. Ahsan [14] defines QoS as the collective effort of service performance that determines the degree of satisfaction from a user about the recommended service. QoS has non-functional constraint requirements that must be met during the process of selecting services. Those are constraints such as reliability, response time, throughput and integrity. The QoS value from the perspective of a consumer can be positive, negative, closer, or exact and sometimes functional properties make use of domain ontology [15]. In order to provide the consumer with the requested service, non-functional properties use QoS ontology. QoS has non-functional constraint requirements that must be met during the process of selecting the services. The QoS constrains are reflected in various parameters that the provider can monitor during service invocation and are used to evaluate the quality level. Considering the relevance of this paper, some of those constraints are reliability, response time, cost, throughput, integrity and platform/API. The QoS constrains are briefly discussed as follows:

Integrity is a degree of trust that is expected from a service provider for reliability and availability purposes. However, the user can check whether the recommended service matches the job submitted when the system claims to be trusted. *Reliability* is the ability of a system or component to perform its required functions under stated conditions for a specified period. *Response time* is the total amount of time taken to recommend the service. It consists of execution time and network-transmission time. The job-execution time depends on the workload and system performance, and can be estimated using existing performance-estimation techniques. The network-transmission time depends on network latency and the size of input data. Response time can be predicted exactly and simply by using processing speed, representing the computing power

of the service provider. *Throughput* is the movement of inputs and outputs through a selection process. It can be described as the rate at which a system generates services per unit of time. *Cost* is the total amount charged per successful execution. Depending on the formula used, which is likely to be determined by the nature of environments where selection is processed, cost may include data volume transferred (which is currently charged for data space such as kilobytes, megabytes, gigabytes, terabytes, etc.), execution time, and other properties. Cost reasonableness attracts more people that would like to participate in the market. *Availability* is important in presenting services during runtime. The availability is not a problem in environments like cloud since they provide on-demand services. Services are provided when they are needed, and providers understand the sense of emergency. *In Platform/application Program Interface (API),* a user may want to specify the API requirements. It is straightforward to deploy a Java-based application to Google's AppEngine [16]. Yau & Yin [17] proposed a selection method based on the QoS ranking. This approach uses the results of the last phase to select relevant services according to the functional aspect. The approach has two phases. First, the classification of the data-mining algorithms copes with the web-service environment into the QoS level based on the QoS constraints. Lastly, it composes the best services by means of the services' semantic connections.

The challenge with the QoS-based approach is that the service-selection system is not able to differentiate similar services based on their features and QoS parameters. The QoS description can be either semantic or syntactic. Semantic QoS is more concerned with the meaning and description of the service. The semantic approach makes the process of selection difficult for composite service (service composed of other services) while the syntactic approach is more concerned with the language. The language used for search engines that uses key words to match the request with requested information is based on the QoS syntactic approach. The system should be able to select a service that best satisfies the user as a candidate for IoT service composition [28]. Other challenge associated with the QoS-based approach is that QoS is most used during trial-and-error tests. The QoS-based approach does not address selection adequately for open environments like model-based or trust environments. QoS is based on requirements from the server and needs from the client that does not exhibit autonomic characteristics. QoS does not have enough support to help users define their QoS requirements.Finding services that are relevant to a service request is the core function of service discovery. The way the results are presented to the client is also a matter of great importance. Presenting search results in a ranked order simplifies service selection work for a client [29]. There have been work done on web-service ranking [30]. Wang *et al*. [31] used the process of hybrid matchmaking that works on the set of returned services. This process aims to find the most relevant way to query and rank services in order. Other challenge is that QoS has no consistent way for the consumer to select services because consumers perceive quality through the prism of their own experience, and evaluate those service maps to the specific quality parameters offered by a provider [18]. Therefore, it is not effective enough to do selection based on QoS only. There is a need to support it with the functional approach in order to balance both the system's nature and selection criteria that distinguishes web services using a set of well-defined QoS criteria [32].

D. Service selection based on functional requirements

The functional-based approach provides information on how the system selects services. The functionalities lead to selection criteria that formulate the algorithm to be used during the service-selection process. A service-level agreement (SLA) is used, but most of the functional approaches are based on artificial intelligence [19]. There is an increase in agreement on the implementation and management of the functional aspects of services but the interest is shifting towards non-functional attributes that describe the QoS [33].

E. Service selection based on user-centred QoS

The approach based on user-centred QoS is proposed by Mobedpour & Ding [20], whereby experienced users are not the focal point. Instead, the proposal is more expressive and flexible for non-expert users to define their own QoS requirements. The QoS-based approach in this context is designed to help users to find their best matching services using their quality requirements. The system design guides the user through the selection process with sufficient information. The user-centred QoS-based approach is based on artificial intelligence and it gives a user an interface to browse to check the available service(s) in order to gain ideas and make a choice. This approach targets non-expert users and supports ranked, relaxed, preference and fuzzy results. Two challenges have been identified in this approach. First, the challenge associated with this approach is that it assumes that a user is capable of formulating queries for the service-selection process. There may be no service that matches the requested service from those that are available. Lastly, is to get a user to request in whatever an acceptable form and break it down into required user task and preferences, which can then be easily mapped to the services existing on the cloud.

The selection criteria in the literature are based on description, semantics, quality, rating, effectiveness, scalability of the service and other aspects. According to the study conducted by Yu and Marganeic [1], it is not easy to evaluate each approach based on the universal defined approach. Hence, this paper proposes aspects that should be considered when selecting an approach that will yield effective results.

IV. PROPOSED EVALUATION METHOD

The system should support the following considerations:
User preference-based - users may have their opinions and reasons when choosing services. For that, there is a need for the well-defined interface for the user profile to capture the user preferences. There is a need for a system that will define both functional and non-functional properties. User preference is one component that integrates human interaction with the system. In this work, context is defined as location, time and duration. It helps to provide well-defined and standardised interfaces offering all functionalities for interacting with entities and related processes (Barnaghi et al., 2013). IoT services are less functional than web services; however, one may want to know about the nature and the state of a "Thing" in order to continue with querying the service. *Domain-*

specification- specifies the scope of data and gives awareness in order to visualise a dataset. Smart environments are based on ubiquitous computing, where environments interact with their inhabitants on a device layer [21]. The data have to be stored intelligently and used for smart monitoring and actuation. This suggestion leaves one with the question as to whether the use of an artificial intelligence algorithm will satisfy users or will it give the users interfaces with which they can express themselves when submitting the job requested. In order to answer this question, there is a need to invoke the user profiling component that captures user preference on a well-defined interface. According to Silva *et al.* [21], the other challenge faced by smart-environment technologies is to improve decision making, sense making, user experience, and cater for the convenient situations, saving energy, and other concerns that may arise as needs increase and technology improves. Based smart-environment technologies, it is hoped that it will improve the quality of life and reduce the ecological impact of humankind, since it eliminates human involvement. This discussion shows that a domain has its own expectations, requirements and challenges.

Storage - enables businesses to publish and discover service listings and define how the services or software applications interact over the Internet. Universal Description, Discovery, and Integration (UDDI) is the commonly used storage mechanism in SOA and web services. A smart campus has the distributed resources and services. However, according to Manikrao & Prabhakar [23], there is a need to get a storage mechanism that will be able to integrate services logically to make the concept. Cloud storage serves as a centralised database where data is stored in virtualised pools of storage which are generally hosted by third parties [22]. Physically, the resource may span multiple servers and multiple locations. The security of the services depends upon the hosting units, and on the applications that leverage the cloud storage. *Scalability* - refers to the ability of a system, network, or process to handle a growing amount of services in a capable manner, or its ability to be expanded to accommodate that growth [24]. Distributed systems have a limited memory and have sharing restrictions. Cloud service is capable of providing online object storage for files and functionalities, then deliver them globally or locally depending on the domain specification. IoT is a global network infrastructure linking the physical and the virtual objects through the exploitation of data capture and its communication capabilities [24]. Such an environment demands large data-storage facilities and the sharing of resources and services.

Algorithm- the importance of the algorithm is to support recommendation, since the selection of services will be done based on user preference for a specific need. Singh & Hunhs [25] indicated that users save time by using the recommender system that helps them to choose from a variety of options. The purpose of recommender systems is to pre-select information a user might be interested in Singh & Hunhs [25]. The heuristic algorithm should support recommendation and dynamic environments. *Evaluation*- the evaluation provides proof of the concepts that whether an architectural or algorithm proposed as the solution works better than other solutions reported in the literature, it is worthwhile to check the relationship that exists between the architectural components and how they perform together. Evaluation should be carried out using metrics such as recall, precision and response time. Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Precision and recall are usually presented as a percentage or decimal. Response time is the total time interval when a service is invoked until the service is recommended. *QoS* - is a non-functional property of a system. According to Yu & Lin [26], non-functional properties include service tracking. Service tracking is a broker that has a service repository to record all feasible web services.

V. APPLICATION SCENARIO

A. Scenario

The user is looking for a service that can do function Z (check if the blinds are closed or opened). The user may request for this service within or outside the premises of the smart campus. The user expects the application to be able to recommend the relevant service(s) to the user.

Based on the scenario, in order to select a mechanism that can be used to select the IoT services, the desired mechanism should support the following environmental considerations: *Integration*- in order to select services, there is a need to find a storage mechanism that will enable the integration of services from different departments of the smart campus. *Scalability*- the mechanism adopted should be able to support a large number of services integrated into a database. *User preference*- the service-selection system should be flexible to allow users to define the service that corresponds to their profile. *Algorithm* - the mechanism should support user preference and personalisation. *Domain* - the mechanism to be proposed should be able to support the smart-campus environment.

B. Findings:

The application was implemented on Android Studio version 0.8.2.The application was installed and tested on android device Galaxy S4 GT-19500 that utilizes. SQLite to store user's authentication information. The performance of the application was measured by the response time.

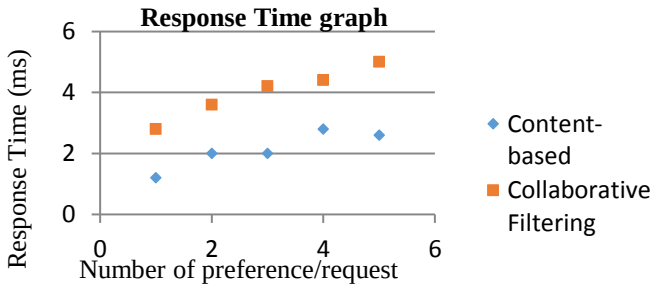

Figure 3: Response time comparison

The total time taken by the application from request to the recommendation, figure 3 shows that content-based was more effective than the collaborative filtering. Preference recall measures how good the algorithm developed in making sure that there is no missing relevant recommendations as indicated in figure 4. Preference precision measures how good the algorithm is in reducing irrelevant recommendations. A good preference model is expected to optimise these two parameters.

The results show that content-based is more effective than collaborative filtering technique.

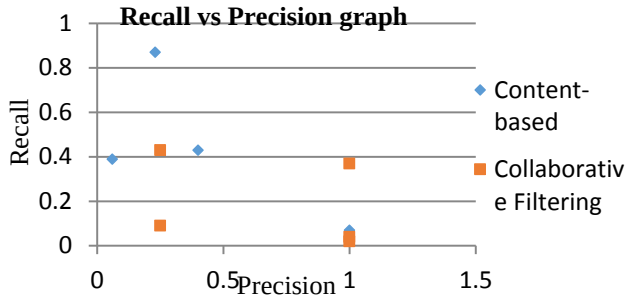

Figure 4: Recall vs precision evaluation

VI. CONCLUSION

The existing service selection approaches identified were multi-agent based, ontology-based, QoS-based, user-centred QoS-based and functional based approaches. Each approach serves its own purpose in various infrastructures and environments. Most of the approaches studied were based on QoS. The findings indicate that it is important to consider other aspects such as domain, storage, scalability, algorithm, QoS, and evaluation metrics to improve the effectiveness and performance. The findings were tested on IoT services provided on a smart campus. The results indicated that content-based performed better than collaborative filtering algorithm having considered all the aspects discovered on this paper.

REFERENCES

[1] Yu, H.Q. and Reiff-Marganiec, S. (2008). Non-functional Property based service selection. A survey and classification of approaches. In: Non Functional Properties and Service Level Agreement in Service Oriented Computing Workshop co-located with The 6th IEEE European Conference on Web services, 12 - 14 Nov 2008, Ireland, Dublin.

[2] INTERNET SOCIETY, 2015. The Internet of Things: An Overview.

[3] Chung, Lawrence, et al. Non-functional requirements in software engineering. Vol. 5. Springer Science & Business Media, 2012.

[4] Chouiref Z., Belkhir A., Benouaret K., Hadjali A., A fuzzy framework for efficient user-centric Web service selection, Applied Soft Computing, Volume 41, April 2016, Pages 51-65, ISSN 1568-4946.

[5] Maximilien E.M. and Singh M.P., 2004. A Framework and Ontology for Dynamic Web Services Selection. Publisher: IEEE Computer Society.

[6] Chen Z., Chia L., and Lee B. "Service discovery and measurement based on DAML-QoS ontology." Special interest tracks and posters of the 14th international conference on World Wide Web. ACM, 2005.

[7] Deora, V. (2007). Quality of Service Support for Service Discovery and Selection in Service Oriented Computing Environment (Doctoral dissertation, Cardiff University).

[8] Kuropka, D., Laures, G. and Tröger, P., 2008. Core concepts and use case scenario. *In Semantic Service Provisioning* (pp. 5-18). Springer Berlin Heidelberg.

[9] Barnaghi, P., Wang, W., Henson, C. and Taylor, K., 2012. Semantics for the Internet of Things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS), 8(1), pp.1-21.

[10] Wang, W., De, S., Cassar, G. and Moessner, K., 2013. Knowledge representation in the internet of things: semantic modelling and its applications. automatika, 54(4), pp.388-400.

[11] Botta, A., De Donato, W., Persico, V. and Pescapé, A., 2014, August. On the integration of cloud computing and internet of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on* (pp. 23-30). IEEE.

[12] Tran, V.X., Tsuji, H. and Masuda, R., 2009. A new QoS ontology and its QoS-based ranking algorithm for Web services. Simulation Modelling Practice and Theory, 17(8), pp.1378-1398.

[13] Kritikos, K. and Plexousakis, D., 2007, November. OWL-Q for semantic QoS-based web service description and discovery. In Proceedings of the 2007 International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 243 (pp. 114-128). CEUR-WS. org.

[14] F, S.M., 2006. A framework for QoS computation in web service and technology selection. Computer Standards & Interfaces, 28(6), pp.714

[15] Sathya, M., Swarnamugi, M., Dhavachelvan, P. and Sureshkumar, G., 2010. Evaluation of qos based web-service selection techniques for service composition. International Journal of Software Engineering, 1(5), pp.73-90.

[16] Zhao, L., Ren, Y., Li, M. and Sakurai, K., 2012. Flexible service selection with user-specific QoS support in service-oriented architecture. Journal of Network and Computer Applications, 35(3), pp.962-973.

[17] Yau, S.S. and Yin, Y., 2011, July. Qos-based service ranking and selection for service-based systems. In *Services Computing (SCC), 2011 IEEE International Conference on* (pp. 56-63).

[18] Maabed, U.M., El-Fatatry, A.M. and El-Zoghabi, A.A., 2013. Towards Enhanced Service Oriented Architecture to Improve E-Services Applications. International Journal of Computer Science & Communication Networks, 3(1), p.1.

[19] Moghaddam, M., 2015. Combinatorial Auction-based Mechanisms for Composite Web Service Selection.

[20] Mobedpour, D. and Ding, C., 2013. User-centered design of a QoS-based web service selection system. Service Oriented Computing and Applications, 7(2), pp.117-127.

[21] Guerra, C.A.N. and da Silva, F.S.C., 2008, April. A middleware for smart environments. In AISB 2008 Convention Communication, Interaction and Social Intelligence (Vol. 1, p. 22).

[22] Sukhamrit, K., Kuljit, K. and Dilbag, S., 2012. Evaluating Performance of Web Services in Cloud Computing Environment withHigh Availability. Global Journal of Computer Science and Technology: B Cloud & Distributed, 12(11), p.2.

[23] Manikrao, U.S. and Prabhakar, T.V., 2005, August. Dynamic selection of web services with recommendation system. In Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on (pp. 5-pp). IEEE.

[24] Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M. and Ochiai, O., 2015. Big data challenges in building the global earth observation system of systems. Environmental Modelling & Software, 68, pp.1-26.

[25] Singh, M.P. and Huhns, M.N., 2006. Service-oriented computing: semantics, processes, agents. John Wiley & Sons.

[26] Yu, X., Lin, J., Zack, D.J. and Qian, J., 2006. Computational analysis of tissue-specific combinatorial gene regulation: predicting interaction between transcription factors in human tissues. Nucleic acids research, 34(17), pp.4925-4936.

[27] Jembere, E., Xulu, S.S. and Adigun, M.O., A Grid Infrastructure for Knowledge-based applications in Open and Dynamic Computing Environments.

[28] Zhao, X., Huang, P., Liu, T. and Li, X., 2012. A hybrid clonal selection algorithm for quality of service-aware web service selection problem. Int J Innov Comput Inf Control, 8(12), pp.8527-8544.

[29] Maximilien, E.M. and Singh, M.P., 2004,. Toward autonomic web services trust and selection. In Proceedings of the 2nd international conference on Service oriented computing (pp. 212-221). ACM.

[30] Mirmotalebi, R., Ding, C. and Chi, C.H., 2012, November. Modeling user's non-functional preferences for personalized service ranking. In International Conference on Service-Oriented Computing (pp. 359-373). Springer Berlin Heidelberg.

[31] Wang, W., De, S., Cassar, G. and Moessner, K., 2013. Knowledge representation in the internet of things: semantic modelling and its applications. automatika, 54(4), pp.388-400.

[32] Cao, L., Li, M. and Cao, J., 2005,. Cost-driven web service selection using genetic algorithm. In International Workshop on Internet and Network Economics (pp. 906-915). Springer Berlin Heidelberg.

[33] Zemni, M.A., Benbernou, S. and Carro, M., 2010. A soft constraint-based approach to qos-aware service selection. In International Conference on Service-Oriented Computing (pp. 596-602). Springer Berlin Heidelberg.

[34] De, S., Barnaghi, P., Bauer, M. and Meissner, S., 2011. Service modelling for the Internet of Things. In Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on (pp. 949-955). IEEE.

[35] Lamparter, S., Ankolekar, A., Studer, R. and Grimm, S., 2007, May. Preference-based selection of highly configurable web services. In Proceedings of the 16th international conference on World Wide Web (pp. 1013-1022). ACM.