Journal of Information Systems and Telecommunication
Vol.10, No.2, April-June 2022, 80-88
http://jist.acecr.org
ISSN 2322-1437 / ESSN:2345-2773

A Novel Effort Estimation Approach for Migration of SOA Applications to Microservices

Vinay Raj[1*], Sadam Ravichandra[2]

[1].BVRIT Hyderabad, College of Engineering for Women, Hyderabad, India.
[2].National Institute of Technology Warangal, Telangana, India.

Received: 20 Aug 2020 / Revised: 27 Jul 2021/ Accepted: 05 Dec 2021
DOI:

Abstract

Microservices architecture's popularity is rapidly growing as it eases the design of enterprise applications by allowing independent development and deployment of services. Due to this paradigm shift in software development, many existing Service Oriented Architecture (SOA) applications are being migrated to microservices. Estimating the effort required for migration is a key challenge as it helps the architects in better planning and execution of the migration process. Since the designing style and deployment environments are different for each service, existing effort estimation models in the literature are not ideal for microservice architecture. To estimate the effort required for migrating SOA application to microservices, we propose a new effort estimation model called ***Service Points***. We define a formal model called service graph which represents the components of the service based architectures and their interactions among the services. Service graph provides the information required for the estimation process. We recast the use case points method and model it to become suitable for microservices architecture. We have updated the technical and environmental factors used for the effort estimation. The proposed approach is demonstrated by estimating the migration effort for a standard SOA based web application. The proposed model is compatible with the design principles of microservices and provides a systematic and formal way of estimating the effort. It helps software architects in better planning and execution of the migration process.

Keywords: Service Oriented Architecture; Microservices; Migration; Service Graph; Effort Estimation.

1- Introduction

Distributed systems have evolved rapidly beginning with the monolithic style of designing applications. Monolithic application has a large codebase, deployed as a single unit and the components of the application are highly coupled. Monolithic architecture has a limitation in the size and complexity of the application.

With the increase in the complexity of enterprise applications, business requirements, and the need for designing distributed applications has led to the evolution of SOA [1]. Service oriented architecture (SOA) has been widely used in designing large enterprise applications in the last two decades. It has mainly emerged to overcome the scalability and deployment challenges of monolithic applications. SOA is a style of designing applications where all the components in the system are designed as services. A service is a reusable software code that performs various business tasks that can be simple or complex based on the business requirements. SOA is mainly used in the integration of multiple software components using the Enterprise Service Bus (ESB) as the communication channel [2]. ESB is the backbone of SOA which helps in providing the features of the middleware system. ESB acts as a mediator between the service requester and provider and provides a platform for high performance and scalability. SOA gained more popularity with the evolution of web services which is the popular implementation of SOA concepts. Web services are also services that can be designed, accessed, and discovered over the internet using communication protocols such as XML based Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL). Web services use HTTP and Representational State Transfer (REST) protocols for the transfer of messages through the internet. The web services architecture comprises three main components namely service provider, service consumer, and service registry. A single web service can be used by multiple clients at the same time and can be easily deployed. Though SOA has gained huge demand in designing applications, it exhibits few design and deployment challenges [3]. Many changes and updates occur in large enterprise applications [25] and when there is a need to update a particular service, due to the

✉ **Corresponding Author**
rvinay1@student.nitw.ac.in

dependency it has on other services and the existence of tight coupling with ESB, it requires to redeploy multiple components for a change in a single service. Deploying multiple services at a time leads SOA to the monolithic style of deployment and it impacts the business [4]. Additionally, with the increase in ever-changing business requirements, few services in SOA are tending towards monolithic in size making the application complex and difficult to maintain. Scaling such monolithic applications is a bottleneck as SOA follows centralized governance [5]. Services that are overloaded can be scaled horizontally by making multiple copies of the same service but the hardware cost increases. Further, web services use complex and heavyweight protocols such as SOAP for the exchange of messages between the services [27].

To over these challenges in existing architectures, microservices emerged as a new style of designing applications using cloud-based containers for deployment [26]. It is a style of designing applications where each service is a small, loosely coupled, scalable and reusable service that can be designed and deployed independently [6]. Each service should perform only one task and should have its own database and independent deployment architecture. Microservices uses communication protocols like HTTP/REST and JSON for data exchange between the services. Unlike SOA, microservices can be deployed independently as there is no centralized governance and no dependency on middleware technologies. It is very easy to scale on-demand microservices with the use of cloud-based containers. Microservices architecture suits well with the DevOps style as every task is to be broken into small units and complete SDLC is to be done independently [7]. DevOps and agile methodologies require the fast design of applications and deployment to production.

With the various benefits of microservices, software architects have started migrating their existing legacy applications to microservices architecture [8]. Many companies including Netflix, Amazon, and Twitter have started building their new applications with this style of architecture [9]. As microservices has emerged recently, there is a huge demand in both industry and academia to explore the tools, technologies, and programming languages used in this architecture. However, some of the software architects are in chaos whether to migrate to this new style or not as they are unaware of the pros and cons of using microservices. The major challenge is estimating the effort required to migrate the existing applications to microservices [8][10].

Effort estimation helps software architects in the proper execution and management of the project. Effective estimation helps in proper scheduling of the software engineering activities. Software effort is given by the formula effort = people * time [11]. It has to be done during the early stage of the application design as it gives insights on the effort and cost required to complete the application. Moreover, estimating the accurate effort required for the migration process is a challenging task. Underestimation and overestimation of the effort required may lead to serious project management issues. Software effort estimation techniques are divided into four types namely, empirical, regression, theory-based, and machine learning techniques based estimation [12]. Empirical way of estimating is very popular as it gives a clear picture of the effort required numerically and few of the models include function point, use case point, and analogy based techniques. The function point and COCOMO model fail to estimate the effort and cost required to design the application [13]. Parametric and non-parametric forecasting models are used in regression approaches. Multiple Linear Regression (MLR), Stepwise Regression (SR), Poisson Regression, Standard Regression, Ordinary Least Squares (OLS), and Stepwise Analysis of Variance are some of the most used regression approaches. Theory-based approaches are based on theoretical concepts that characterize certain parts of software development processes. Machine Learning based approaches for estimating software effort includes Artificial Neural Networks, Classification and Regression Tree, Case-based Reasoning, Genetic Algorithm, Genetic programming, and Rule Induction. Moreover, these techniques are not suitable for measuring the effort for service-based systems as they are designed for procedural object-oriented systems.

Use Case Points (UCP) is a commonly used technique because of its simplicity, fastness, and accuracy to a certain extent [14]. UCP approach is based on the use case diagrams for calculating the effort. Many variations and enhancements have been published in the literature to improve the accuracy of the approach [15][16][17]. Though the use case point approach is based on the use case diagrams of object-oriented concepts, attempts have been made for estimating the effort for service-oriented architectures [18]. All the traditional approaches available for effort estimation cannot be used directly for service-based systems. Approaches need to be modified and extended to cope with these service-based systems like service oriented architecture and microservices architecture [19].

To the best of our knowledge, there has been no work or very little work done in estimating the effort required for migration of service oriented architectures to microservices architecture. In this paper, we attempt to propose an approach for effort estimation by recasting the existing use case point model by enhancing it to suit appropriately for microservices. Generally, effort estimation requires knowing about the system before the design phase which is difficult. In our work, we use the service graph representation of the microservices application which is generated by the migration approach

[20] and it gives detailed information about the number of services and dependency it has on other services.

The remaining paper is organized as follows. The types of services involved in the migration process are discussed in section 2. The approach for effort estimation is presented in section 3. Evaluation of the proposed approach using a case study application is presented in section 4. Section 5 concludes the paper.

2- Types of Services Involved During the Migration Process

To migrate SOA based applications to a microservices architecture, the monolithic services need to be broken into small and independent services. However, there may exist few services in SOA based application which perform a single business task and can be directly considered as microservices. For systematic estimation of the effort, business services are classified into available, migrated, new, or composed services [21]. However, there exist many other types of services that are involved in achieving the business requirements such as utility services, process services, proxy services, integration services and, suspended services, etc. Here, we discuss the significance of each service in the migration of SOA to microservices architecture.

Available service: Services that can be used directly in the new architecture are treated as available services. Service which does a single business task and is independent of other services can be directly considered as microservice. It requires no development effort and hence it is considered as available service.

Migrated service: Service which is extracted from legacy applications and generated by applying different migration strategies is considered as migrated service. Here, the services in SOA which are partitioned to form microservices will be considered as migrated service. These services require an effort for redesigning the new application. The difference between available service and migrated service is available service can be directly used as microservice whereas migrated service requires effort for transforming itself to a new microservice.

New service: Service which is built from scratch and required for achieving the business needs is considered a new service. It requires effort and it is very easy to calculate the effort for new service. However, as both SOA and microservices architectures are service-based systems, no new services will be required while migration from SOA to microservices. Therefore, we will not consider this kind of service in effort estimation.

Composed service: Service which is formed by combining one or more services is considered as composed service. By the definition of microservices, each service should perform only a single task and independent from other services. Therefore, there will be no composed services in the new architecture.

It is inferred from the above that in the effort estimation of the migration process, only the migrated services need to be considered. So the proposed model considers only the migrated services in the effort estimation.

3- Effort Estimation Approach

3-1- Service Graph

Graph theory has been widely used in solving many complex problems in software engineering as the flow of messages and dependency between the software components can be graphically represented. As services are the software components in SOA based applications, we develop a new graph called service graph (SG) to extract the candidate microservices. We start by introducing the concept of service graph which plays a fundamental role in our proposed approach.

Service graph (SG) is a regular graph generated for the visual analysis of communication and dependency between the services of an SOA application. The service graph is the simplest representation of the number of services and the interactions among those services. The generalized form of any given SOA based application as a service graph is shown in Fig 1.

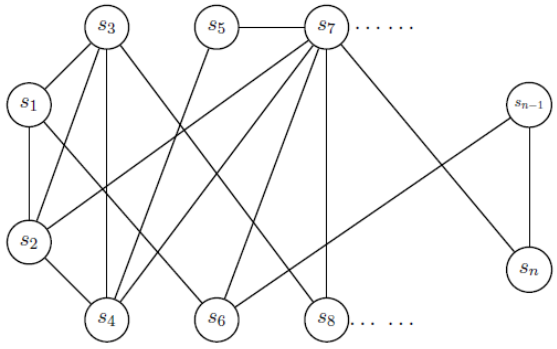

Figure 1: Formal Representation of SOA application

3-1-1-Service Definition

Let a graph G(*V, E*) be a service graph with **n** nodes, where the nodes of the graph represent a set of services in the application, and edges between the nodes represent the interactions or dependency each service has with other services in the application.

Let V={s_1,s_2,s_3,...} be the nodes of the service graph where s_1,s_2,s_3,… are services and E= {(s_1,s_2), (s_1,s_3), (s_2,s_4), ...} be the edges between the nodes which represent the dependency between the services. A service can be

represented as a set of coordinating and interacting processes as defined in Eq. (1).

$$S_i = < P_1^i, P_2^i, P_3^i, \ldots.., P_n^i, \Lambda > \quad (1)$$

where S_i is the logical service instance, P_k^i indicates k^{th} process implementing logical service functionality f_i through the programmatic interface I_i and Λ represents network communication function between individual processes [22].

3-2-Proposed Approach

Our approach is stimulated from the use case points model of effort estimation. The use case point method depends on the use case diagram and our model depends on the service graph as we are estimating the effort for service-based architectures. The service graph is a blueprint for the application to be designed and it gives complete information regarding the number of services and complexity of the services based on the dependencies on other services. Similar to the use case point method, we propose a service point (SP) model to estimate the effort required for migration to microservices. We classify the services and then calculate the weights and points using the classification of the services. Technical and environmental factors are two important factors that play a major role in effort estimation. The factors accessed for the existing use case point method does not suit well for microservices architecture. Therefore, we have updated the technical and environmental factors considering the principles of service-oriented systems. The steps for effort estimation using the service point technique is illustrated in Fig. 2.

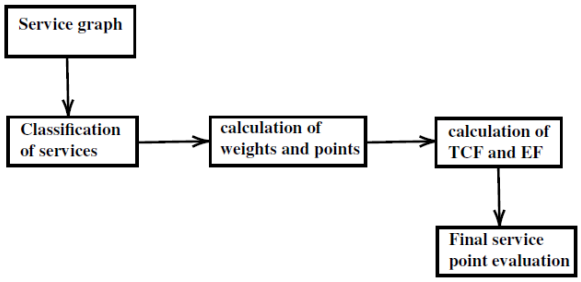

Figure 2: Service point calculation steps

3-2-1-Classification of Services

The first step of the service point approach is to classify the services based on the interactions it has with other services. Unlike the use case point, we don't have actors here. So, we consider the dependencies each service has on other services and classify them as simple, average, and complex. The service graph helps in the identification of services and their dependencies. A service is classified as simple if it interacts with less than four services, average if it interacts with less than eight services, and service is treated as complex if it interacts with more than or equal to eight services. The number of interacting services for a particular service helps in determining the service complexity. Based on the complexity, different weights are assigned to each service which is used in the calculation of service weights. The classification of services and the weights assigned are given in Table 1.

Table 1: Classification of services with weights

Service Complexity	***Number of interacting services***	***Weight***
Simple	Less than or equal to 3	5
Average	4 to 7	10
Complex	More than 7	15

3-2-2-Calculation of weights and points

The next step is to calculate the unadjusted service points based on the weights assigned in Table 1. It is calculated by summation of number of services of each type multiplied by weight assigned to corresponding service type. Unadjusted Service Points (USP) is calculated as shown in Eq. (2).

$$USP = \sum_{i=1}^{3} S_i \times W_i \quad (2)$$

Where S_i is the number of services of type i and W_i is the corresponding weight of the service of type i where i={simple, average, complex}.

3-2-3-Technical and Environmental factors

We calculated the unadjusted service point value from the Eq. (2) and the final value of service point depends on technical and environmental factors. The 21 factors [23] relate to the factors which contribute to the complexity and the efficiency of the system. However, most of the factors included in existing works presented in the literature are not suitable for both service oriented architecture and microservices. Therefore, we have removed few factors and added new factors relevant to microservices architecture.

Each factor has a value assigned between 0 and 5 depending on the importance and impact the factor has on the system. In the existing use case points approaches, weights have been assigned based on the experience in their projects [23]. However, we have conducted an online survey to collect the inputs from different practitioners working on SOA and microservices architectures, software

architects involved in the migration process and developers working with microservices architecture. We have posted the online questionnaire in multiple social networking platforms including the groups in LinkedIn, Twitter and Facebook etc. The questionnaire included the following questions.

- What is the current role/designation of the participant?
- How much work experience the participant has in SOA and microservices projects?
- Does the participant has real time experience in migration projects?
- How much rating does the participant would like to rate for each of the 21 factors?

The rating of each factor between 0 and 5 for each factor are collected through this survey. Based on the data collected, we have taken the average of ratings and assigned them to all the factors. The weights assigned and ratings of technical and environmental factors are indicated in Table 2 and Table 3.

3-2-3-1-Calculation of Technical Complexity Factor(TCF)

To calculate the TCF, total weight of the 13 factors is calculated which is obtained by multiplying the value assigned to each factor between 0 to 5 and weights assigned to each factor. Calculation of TFactor is given by Eq. (3),

$$TFactor = \sum_{i=1}^{13} TF_i \times W_i \quad (3)$$

where TF_i is the rating of the technical factor ***i*** and W_i is the weight assigned to corresponding factor. As per the use case points method, the impact of Technical Complexity Factor (TCF) on the proposed service points should vary from a range of 0.6 to 1.3. The formula to calculate TCF is given as below:

TCF = C1 + C2 * TFactor.

Hence, we consider the lowest range value for C1 i.e 0.6 and C2 is calculated as C2 = (1.3-0.6)/50 = 0.014 where 50 is the maximum value of TCF. Therefore, the TCF value for the proposed service point is calculated by the below Eq. (4).

$$TCF = 0.6 + (0.01 \times TFactor) \quad (4)$$

Table 2: Technical Factors

F_i	***Factors contributing to complexity***	W_i	***Rating***
F_1	Distributed systems	2	5
F_2	Application performance objectives	1	4
F_3	End-user efficiency	1	2
F_4	Complex internal processing	1	2
F_5	Reusability	1	3
F_6	Easy Installation	0.5	1
F_7	Interoperability	0.5	2
F_8	Portability	0.5	1
F_9	Changeability	1	1
F_{10}	Coupling	1.5	5
F_{11}	Modularity	2	4
F_{12}	Statelessness	1	3
F_{13}	Independent deployment	1	4

3-2-3-2-Calculcation of Environmental Factor (EF)

Similarly, the impact of environmental factors in the final service point is evaluated by finding the EF score. To calculate the EF value, the weight of each factor is multiplied with rating assigned to each factor. It is given by the Eq. (5).

$$EFactor = \sum_{i=1}^{8} EF_i \times W_i \quad (5)$$

The impact of Environmental Factor (EF) is more on the proposed service points method and it varies from a range of 0.0425 to 1.4. The formula to calculate EF is as below:

EF = C1 + C2 * EFactor.

Since, its impact is high, we consider the highest range value for C1 and C2 is calculate as C2 = (1.4-0.0425)/37.5 = 0.03 where 37.5 is the maximum value of EF. The EF for proposed approach is calculated by the below Eq. (6).

$$EF = 1.4 + (-0.03 \times EFactor) \quad (6)$$

Table 3: Environmental factors

F_i	***Factors contributing to efficiency***	W_i	***Rating***
F_1	Familiar with containers	1.5	3
F_2	Service configurations	1	2
F_3	Analyst capability	0.5	4
F_4	Application experience	0.5	2
F_5	Cloud computing experience	1	2
F_6	Motivation	1	5
F_7	Polyglot	1.5	2
F_8	Stable requirements	1	4

3-2-4-Final service point evaluation

The final Service Points (SP) is calculated by multiplying the unadjusted service point with both technical and environmental factor values. It is given by below Eq. (7).

$$SP = USP \times TCF \times EF \quad (7)$$

According to Karner [23], the effort required to implement each service point takes 20 hours. Therefore, to estimate the final man-hours, the calculated service point should be multiplied with 20 to get the effort required for migration. Moreover, it is observed that effort required for migrating and designing a microservices application is more compared to designing existing legacy applications [8].

4- Case Study Application

To evaluate and demonstrate the proposed approach, we choose a standard web application that is built based on SOA. In [24], the author has chosen a Vehicle Management System (VMS) application to demonstrate the migration of the legacy application to SOA style. Taking the SOA application as input and applying the microservices extraction approach proposed by Raj, V. *et al.* [20], we have generated the service graph for corresponding microservices based application. The service graph of microservices application is represented in Fig. 3. The service graph is the prototype of a microservices application that has to be built through the migration process. From the service graph represented in Fig. 3, it is clear that there are 12 services in the migrated system. The details of the SOA services, extracted microservices, and the type of services are mentioned in Table 4. As mentioned in Section 3.1, only the migrated service for estimating the effort as few services in SOA based applications can be directly considered as microservices. The calculation of service points according to the proposed approach is presented in the next section.

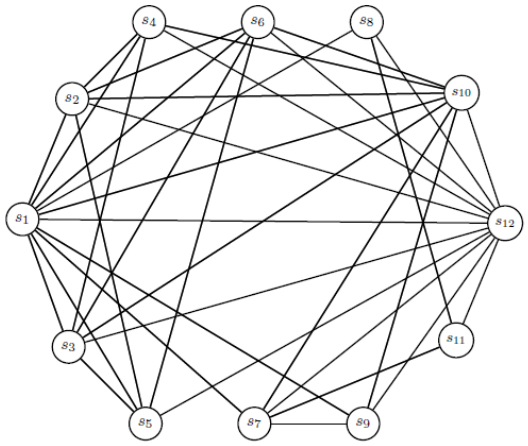

Fig. 3 Service graph representation of microservices based web application

Table 4: Details of extracted microservices from SOA application

SOA Services	***Microservices***	***Notation in service graph***	***Type***
Config Service	Config Service	S_1	Available
Part Service	Part Service	S_2	Available
Product Service	Product Service	S_3	Available
Compare Service	Compare Service	S_4	Available
Incentives & Pricing Service	Incentives Service	S_5	Migrated
	Pricing Service	S_6	Migrated
Dealer & Inventory Service	Dealer Service	S_7	Migrated
	Dealer Locator Service	S_8	Migrated
	Inventory Service	S_9	Migrated
Lead Service	Get-A-Quote Service	S_{10}	Migrated
	Lead Processor Service	S_{11}	Migrated
User Interface Client Service	User Interface Client Service	S_{12}	Available

4-1-Classification of Services

The details of the services along with classification are presented in Table 5. Based on the classification and the weights and ratings of technical and environmental factors, we calculate the service point value used for migration of SOA based application to microservices architecture.

Table 5: List of services along with classification for microservices based application

Service #	***Interacting Services***	***Classification***	***Services considered in estimation***
S_1	2,3,4,5,6,7,9,10, 12	Complex	
S_2	1,4,5,6,10,12	Average	
S_3	1,4,5,6,10,12	Average	
S_4	1,2,3,10,12	Average	
S_5	1,2,3,6,12	Average	✓
S_6	1,2,3,5,10,12	Average	✓
S_7	1,9,10,11,12	Average	✓
S_8	11,12	Simple	✓
S_9	1,7,10,12	Average	✓
S_{10}	1,2,3,4,6,7,9,12	Complex	✓
S_{11}	7,8,12	Simple	✓
S_{12}	1,2,3,4,5,6,7,8,9, 10,11	Complex	

As discussed in Section 2, the services which are classified as migrated services are considered in effort estimation. The other types of services either does not require effort or

not suitable in this migration process. Only the services with tick mark will be considered for effort estimation as they are migrated services.

4-2-Calculation of USP

Unadjusted service point value is calculated by multiplying the number of services based on each classification and the weights assigned to each type. From the information from Table 5, there are 2 simple, 4 average and 1 complex services. Therefore, the value of USP is

$$USP = (2 \times 5) + (4 \times 10) + (1 \times 15) = 10 + 40 + 15 = 65.$$

4-3-Considering the Ratings of the Factors Collected Through Online survey

4-3-1-Technical Complexity Factor

First, we need to calculate the TFactor using the information from Table 2. TFactor value is calculated as given below

$$TFactor = \sum_{i=1}^{13} TF_i \times W_i = 46.5$$

Now, we calculate the TCF value.

$$TCF = 0.6 + (0.01 \times TFactor) = 0.6 + (0.01 \times 46.5) = 1.065$$

4-3-2-Environmental Factor

Similarly, we calculate the EFactor using the information from Table 3 and then use this value of EFactor to calculate the EF value.

$$EFactor = \sum_{i=1}^{8} EF_i \times W_i = 23.5$$

Environmental Factor (EF) is calculated by the below equation

$$EF = 1.4 + (-0.03 \times EFactor) = 1.4 + (-0.03 \times 23.5) = 0.695$$

4-3-3-Final service point calculation

The service point is given as the product of USP, TCF, and EF. It is calculated as below.

$$SP = USP \times TCF \times EF = 65 \times 1.065 \times 0.695 = 48.11$$

The total effort required for migrating the SOA based VMS application to microservices is calculated by multiplying the number of services points with 20 hours.

Total estimated effort = 48.11 * 20 ≈ 962 hours.

4-4-Considering the Default Value Suggested by Karner

Karner suggests that if we cannot fill the values for the factors for any reason, we can use the default value as 3 for all the factors [23]. Considering this default value for all factors, we calculated the TCF, EF and service points values.

4-4-1-Technical Complexity Factor

$$TFactor = \sum_{i=1}^{13} TF_i \times W_i = 42.$$

$$TCF = 0.6 + (0.01 \times TFactor) = 0.6 + (0.01 \times 42) = 1.02.$$

4-4-2-Environmental Factor

$$EFactor = \sum_{i=1}^{8} EF_i \times W_i = 24.$$

$$EF = 1.4 + (-0.03 \times EFactor) = 1.4 + (-0.03 \times 23.5) = 0.68.$$

4-4-3-Final service Point Calculation

$$SP = USP \times TCF \times EF = 65 \times 1.02 \times 0.68 = 45.1.$$

Total estimated effort =45.1 * 20 ≈ 902 hrs.

4-5-Observation

By considering the TCF, EF and SP values of both the calculations, the values are very close to each other. Hence, the ratings of factors collected by online survey can be used as reference for estimating the effort of other projects as well.

Table 5: Comparison of values

Ratings	***TCF***	***EF***	***SP***
Collected through online survey	1.065	0.695	48.1
Considering Karner's default value	1.02	0.68	45.1

5-Conclusion

Effort estimation is an important software engineering activity which helps project managers and architects to effectively schedule the project. With the evolution of microservices, companies are migrating existing legacy applications to microservices architecture. In this paper, we propose a new technique which is recasted from the well known use case points technique to estimate the effort required for migration of SOA based applications to microservices architecture. We define a formal model called service graph which is the representation of any service based application. We have revised the technical and environmental factors as the existing factors are not compatible with the microservices architecture. We have conducted online survey to collect the ratings of each of these factors and used in the our effort estimation process. We have demonstrated the new technique through a case

study application and calculated the effort required for migration. We have compared the results with effort calculated by considering the default values for factors suggested by Karner.

However, this is the first attempt to estimate the effort required for migration of SOA based applications to microservices and hence, we could not compare it with existing techniques. This approach is applied only on a single case study application and in future we plan to evaluate the proposed technique on applications of different domains and large enterprise applications.

References

[1] Vinay Raj and R. Sadam, “Patterns for Migration of SOA Based Applications to Microservices Architecture,” Journal of Web Engineering, Jul. 2021, doi: 10.13052/jwe1540-9589.2051.

[2] J. Yin, H. Chen, S. Deng, Z. Wu, and C. Pu, “A Dependable ESB Framework for Service Integration,” IEEE Internet Computing, vol. 13, no. 2, pp. 26–34, Mar. 2009, doi: 10.1109/mic.2009.26.

[3] V. Raj and R. Sadam, “Evaluation of SOA-Based Web Services and Microservices Architecture Using Complexity Metrics,” SN Computer Science, vol. 2, no. 5, Jul. 2021, doi: 10.1007/s42979-021-00767-6.

[4] T. Cerny, M. J. Donahoo, and M. Trnka, “Contextual understanding of microservice architecture,” ACM SIGAPP Applied Computing Review, vol. 17, no. 4, pp. 29–45, Jan. 2018, doi: 10.1145/3183628.3183631.

[5] Z. Xiao, I. Wijegunaratne and X. Qiang, "Reflections on SOA and Microservices," 2016 4th International Conference on Enterprise Systems (ES), 2016, pp. 60-67, doi: 10.1109/ES.2016.14.

[6] J. Thones, “Microservices,” IEEE Software, vol. 32, no. 1, pp. 116–116, Jan. 2015, doi: 10.1109/ms.2015.11.

[7] D. Taibi, V. Lenarduzzi, C. Pahl, and A. Janes, “Microservices in agile software development: a workshop-based study into issues, advantages, and disadvantages,” in Proceedings of the XP2017 Scientific Workshops, 2017, pp. 1–5. doi: 10.1145/3120459.3120483

[8] D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation,” IEEE Cloud Computing, vol. 4, no. 5, pp. 22–32, Sep. 2017, doi: 10.1109/mcc.2017.4250931.

[9] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, “The pains and gains of microservices: A Systematic grey literature review,” Journal of Systems and Software, vol. 146, pp. 215–232, Dec. 2018, doi: 10.1016/j.jss.2018.09.082.

[10] X. Larrucea, I. Santamaria, R. Colomo-Palacios, and C. Ebert, “Microservices,” IEEE Software, vol. 35, no. 3, pp. 96–100, May 2018, doi: 10.1109/ms.2018.2141030.

[11] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger, “A probabilistic model for predicting software development effort,” IEEE Transactions on Software Engineering, vol. 31, no. 7, pp. 615–624, Jul. 2005, doi: 10.1109/tse.2005.75.

[12] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, “An empirical study of the effect of complexity, platform, and program type on software development effort of business applications,” Empirical Software Engineering, vol. 11, no. 4, pp. 541–553, Oct. 2006, doi: 10.1007/s10664-006-9023-3.

[13] S. M. Satapathy, S. K. Rath, and B. P. Acharya, “Early stage software effort estimation using random forest technique based on use case points,” IET Software, vol. 10, no. 1, pp. 10–17, Feb. 2016, doi: 10.1049/iet-sen.2014.0122.

[14] S. M. Satapathy, B. P. Acharya, and S. K. Rath, “Early Stage Software Effort Estimation using Random Forest Technique based on Optimized Class Point Approach”, INFOCOMP Journal of Computer Science, vol. 13, no. 2, pp. 22–33, Dec. 2014.

[15] M. R. Braz and S. R. Vergilio, "Software Effort Estimation Based on Use Cases," 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 2006, pp. 221-228, doi: 10.1109/COMPSAC.2006.77.

[16] S. Diev, “Use cases modeling and software estimation,” ACM SIGSOFT Software Engineering Notes, vol. 31, no. 6, p. 1, Nov. 2006, doi: 10.1145/1218776.1218780.

[17] P. Mohagheghi, B. Anda and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005., 2005, pp. 303-311, doi: 10.1109/ICSE.2005.1553573.

[18] G. Canfora, A. R. Fasolino, G. Frattolillo, and P. Tramontana, “A wrapping approach for migrating legacy system interactive functionalities to Service Oriented Architectures,” Journal of Systems and Software, vol. 81, no. 4, pp. 463–480, Apr. 2008, doi: 10.1016/j.jss.2007.06.006.

[19] Z. A. Siddiqui and K. Tyagi, “A critical review on effort estimation techniques for service-oriented-architecture-based applications,” International Journal of Computers and Applications, vol. 38, no. 4, pp. 207–216, Oct. 2016, doi: 10.1080/1206212x.2016.1237132.

[20] V. Raj and R. Sadam, “A Framework for Migration of SOA based Applications to Microservices Architecture,” Journal of Computer Science and Technology, vol. 21, no. 2, p. e18, Oct. 2021, doi: 10.24215/16666038.21.e18.

[21] E. A. Farrag, R. Moawad, and I. F. Imam, “An Approach for Effort Estimation of Service Oriented Architecture (SOA) Projects,” Journal of Software, vol. 11, no. 1, pp. 44–63, 2016, doi: 10.17706/jsw.11.1.44-63.

[22] A. Yanchuk, A. Ivanyukovich, and M. Marchese, “Towards a Mathematical Foundation for Service-Oriented Applications Design,” Journal of Software, vol. 1, no. 1, Jul. 2006, doi: 10.4304/jsw.1.1.32-39.

[23] G. Karner, “Resource estimation for objectory projects,” Objective Systems SF AB, vol. 17, pp. 1–9, 1993.

[24] P. Bhallamudi, S. Tilley and A. Sinha, "Migrating a Web-based application to a service-based system - an experience report," 2009 11th IEEE International Symposium on Web Systems Evolution, 2009, pp. 71-74, doi: 10.1109/WSE.2009.5630392.

[25] F.S. Aliee, S. Oviesi. A way to improve Adaptive Maintenance in Enterprise Architecture. Journal of Information Systems and Telecommunication;8(1):1-4.2020;doi: 10.7508/jist.2020.01.001

[26] V. Raj and S. Ravichandra, "Microservices: A perfect SOA based solution for Enterprise Applications compared to Web Services," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2018, pp. 1531-1536, doi: 10.1109/RTEICT42901.2018.9012140.

[27] V. Raj and R. Sadam, "Performance and complexity comparison of service oriented architecture and microservices architecture," International Journal of Communication Networks and Distributed Systems, vol. 27, no. 1, p. 100, 2021, doi: 10.1504/ijcnds.2021.116463.