Matrix Clustering based Migration of System Application to Microservices Architecture

Shahbaz Ahmed Khan Ghayyur
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University
Islamabad, Pakistan

Saeed Ullah
Department of Computer Science,
Federal Urdu University of Arts Science and Technology
Islamabad, Pakistan

Abdul Razzaq
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University
Islamabad, Pakistan

Salman Ahmed
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University
Islamabad, Pakistan

***Abstract*—A microservice architecture (MSA) style is an emerging approach which is gaining strength with the passage of time. Micro services are recommended by a number of researchers to overcome the limitations and issues encountered by usage of aging method of monolithic architecture styles. Previously the monolithic applications cannot be decomposed into smaller and different services. Monolithic styles application was the one build application. The issue resolution has the focus on lightweight independent application services in the form of sizable services, self-contained units with primary focus on maintenance, performance, scalability, and online services eliminating dependency. All quality factors have been thoroughly discussed in literature, system application migration is becoming an emerging issue with different challenges. This study is addressing the tight coupling to reducing this issue. Moreover, this literature review indicates some complex problems about the migration or conversion of system application into microservice. In architecture, dependency is a big challenge and issue in recent technology. Microservices are recommended by a number of researchers to overcome the limitations issue about how to migrate the existing system application to microservice. The need for a systematic mapping is essential in order to recap the improvement and identify the gaps and requirements for future studies. This study shows open issues first, new findings of quality attributes of microservices and then this study helps to understand the difference between previous traditional systems and microservices based systems. This research study creates awareness about system migration to microservices.**

Keywords*—Monolithic architecture; microservices architecture; systematic mapping; system migration; application transformation; traditional application development; emerging challenges; API

I. Introduction

A. Microservices

Microservice possesses important characteristics like it claims the responsibility of a single task, it meets all the requirements of a single business, it can be individually deployed, it is loosely coupled and it is independently responsible as it is self-contained [9], [10]. An enterprise application which is designed for a particular organization consists of different microservices which are responsible to communicate with one another with the help of a light-weight protocol and the API contract [2]. MSA design is generally preferred to the conventional Monolithic Architecture due to the fact that it can be continuously deployed and its scalability has no parallel while the conventional Monolithic Architecture lacks all these important features. Because of this undeniable charm of MSA design, most of the enterprises tend to prefer this design [6].

In the beginning, the developers introduced the concept of service orientation with the help of SOA. Later, the evolutionary process took place and service orientation became capable of supporting the easy and swift operability of the applications designed as per requirement [12]. Now, with more research being carried out in this field, researchers have started building independent, multiple and self-contained services to meet the challenges of the market [23]. Because of these features, there is no denying the fact that Software Architecture plays very important role in software lifecycle to support the quality and vital attributes of the software [13]. This approach helps the developers to make sure that quality attributes are up to the complete satisfaction and there exists no defect in the design of the software systems [23]. If maintenance and development of information system needs to be improved, Component Based Development is useful and viable solution for these requirements [13]. Need of SOA approach coupled with its products was felt because of the reasons that a Component Based Distributed Architecture was required in the market [30]. Moreover, a solution was required to make the business agile and meet the challenges that arise when a particular business need is to be met. Moreover, a compatible and flexible solution was required which may become capable of keeping pace with the evolution that takes place with every single day that passes [8].

In service oriented software companies, the micro services have become architecture style that is inspired by service oriented computing. [3], [6]. Microservices architecture helps to develop the complex application along with the distribution of the application in chunks or units by composing it [1]. Nowadays, in any system, the scalability, service discovery and communication among services that are being supported by microservices architecture in development phase are two important sections [45]. Simultaneously, microservices architecture also handles a heavy concurrency during input load [2]. In fact, the purpose for using microservices is that it works on latest platform and is independently deployable [1]. Microservices API can be written in any language. Then, Microservices architecture would automatically make all the languages compatible to display the desired output [4].

There is no denying the fact that the ever-changing evolutionary process of styles of communication and integration has proved to be cyclic. At times, some of its concepts seem to fall apart and looks as if these concepts would become obsolete with the passage of time but these concepts resurface in different and refined forms as the time elapses. Out of these two styles; Service Oriented Architecture is relatively an older concept because of obvious reasons [8].

B. Migration

System application migration is becoming an emerging issue with different challenges. Transform that migration is the procedure of moving from the usage of one functional environment to another operating environment with alike functionalities [32]. The migration procedure contains, and making sure the new environment's features are exploited, old settings do not require changing and that present applications continue to work.

C. Migration to Microservices

The focus of migration is that it indicates some complex problems about the migration or conversion of system application into microservice. Migration of the system to microservice optimizes decentralization, replace-ability and autonomy of software architectures [32]. Although, researchers are not convinced on any specific definition of microservice, its modelling techniques, and its properties [7], it is aware about system migration to microservices.

The components which are used in these vital software applications are made up of basic blocks which can be combined together depending upon the requirement [17]. Microservice architecture is preferred owing to the reasons that it has the capability to address all the concerns starting from requirement of the enterprise to the operations to be performed by the software of a particular business for which it is designed. Moreover, it can also claim the responsibility for individual teams [25], [38]. In this type of approach to find out the solution, the architecture, open source development, organizational structure and responsibility is vertically decomposed [14], [43].

D. Clustering

In this technique, reverse engineering also produces desired results. The technique used for the purpose of reverse engineering is clustering which is the considered the simplest and fundamental technique used in engineering and science [17]. Main and most important objective of implementing this technique is to make the observations clearer to develop a better understanding. This better understanding makes it easy to develop complex knowledge structure from given features. Clustering technique or method is generally preferred to identify all the related components of System Software Application along with their responsibilities. As the input used in this technique highlights the interconnectivity of all these components, this clustering technique is quite useful to minimize the interconnection among different components to produce optimum results [30]. Clustering is a technique in which large systems are decomposed into chunks and smaller and manageable systems in a distinct way that the entities which bear similarity with one another belong to the same subsystem while the entities with difference among one another are classified into different subsystems [17]. Clustering technique is generally used in identifying the software components which generally adopts one metric so that the similarity of components may be measured. The main advantages of this technique are that low coupling and high cohesion of components are achieved. These advantages play very important role to solve the problems which require software evolution [21, [27].

There exist a lot of clustering techniques out of which Hierarchical Agglomerative Clustering (HAC) and K-means clustering stand out. Hierarchical Agglomerative Clustering (HAC) plays very important role to find out the number of clusters or segments which do not work well or cause inconvenience because of malfunctioning in practice [16]. Moreover, K-means is also used to locate the numbers of clusters or segments which do not work well but the only problem that occurs is the fact that it cannot be applied in HAC algorithms.

E. Need of Systematic Mapping

Many different software companies have recently migrated to microservices or are considering migrating to microservices. These services are known as a style of an architecture that develops an application as a set of small services independently [7]. Now, microservices are becoming very popular with cloud platform which is an emerging style in the context of application development due to its independency, scalability, flexibility, performance, and manageability [3], [5]. There is a lot of research in this area that needs to be address. In the previous a few years, the software product companies and software consultancy firms have found the microservices approach useful because it allows the team and software organizations to increase the productivity [6].

Ever-changing needs of customers due to ever-changing situational contexts and business needs inspire the enterprises to introduce evolutionary concepts in software products to compete the market. Due to these developments, most of the Software Development Organizations and the businesses which include Software Production are facing bursting pressure to improve their Software Intensive Systems on daily basis. They can achieve this goal if they develop and release valuable and compatible software in a very short span of time to meet the challenges of the market [11].

II. Background Knowledge

Literature review sheds light upon the importance and architecture of microservice. Moreover, this literature review indicates some complex problems about the migration or conversion of system application into microservice. This discussion in literature answers the very first question of this research paper. It highlights all the issues which involve migration to microservice [32]. The main features which are creating and promoting the demand of such a technology are scalability, security, reliability, fast progress, and speed of the network [20]. So, the researchers are trying hard to introduce new software architecture styles and software development methods to meet all the demands of enterprises [6]. Migration of the system to microservice optimizes decentralization, replace-ability, traceability and autonomy of software architectures. Although, researchers are not convinced on any specific definition of microservice, but it is modelling techniques, and its properties [7].

Microservice plays very important role to capture software maintenance, architecture and evolution [15]. If software architecture recovery is taken into account, it becomes clear that the prevailing techniques in this field are quite limited because all these techniques are based upon reverse engineering [5].

Software designer or developer generally encounters two types of problems in practical. First issue is embedded in the fact that it is quite tough to determine specific cluster which is used for highly coupled components [15]. Second problem in line is to determine the cluster mapping which is applied on software modules [17]. Upon investigation, the technique of decomposition of software has made sure that the source code of software is in accordance with all the requirements gathered.

Main drawback of the traditional monolithic services is its lack of scalability when a certain task is to be executed within the service [9]. Long software release cycle because of the complexity of system is also a hurdle in traditional monolithic services. Because of these limitations monolithic approach has shifted towards the development of modern cloud application [35], [36].

ICT is making a name and becoming a reliable partner in demand driven and dynamic market environment because of its customer experienced, customer centered and ever-changing demand driven competitive market [38]. Because of this competitive race of different enterprises, most of the companies are transforming themselves into virtually organized bodies with pure digital styles. These virtually organized bodies are supported and enabled by the applications based on microservice [18]. Genuinely, microservice in any application is responsible to execute a single task, i.e. it works on only one business requirement at a time. Moreover, it is self-contained that it can complete its responsibility without depending upon any other software [6]. For example, it contains business and data layers, and presentation all together. Additionally, it is loosely coupled, light-weighted and autonomous [1].

If the applications are to be run in cloud with efficiency, it requires much more skill than what is necessary to deploy any type of software in virtual machines. It is always recommended to manage cloud applications continuously in order to utilize their resources according to the incoming load and to face the failures in order to replicate and restate all the components to provide resilience in case of unreliable infrastructure [42]. Once a program or software is designed keeping in view all the requirements, it becomes extremely tough for the designer to introduce radical changes which are later on demanded by business models or user frequently because it becomes more complicated for the developer to make changes when the code starts expanding because of the involvement of different people or specialist who make changes in the software [14], [26]. As more and more effort is required to coordinate for updating in tightly coupled model of monolithic approach, this whole process ultimately makes the release cycle of the application slow [37]. It also makes the model fragile and unreliable. Scalability is also a vital feature which is required in the operation and development of enterprise applications [9], [14], [22].

III. Research Method

A. *Planing of Mapping*

This mapping study, researcher is combining the knowledge for all issues which are related to system migration. There are different types of systems migration techniques but researcher is migrating the system migration that is based on components. This knowledge will help us to migrate the system application and mitigate these issues during migration and why researcher needs migration of system application. This study will aware about microservices understanding and its characteristics.

B. *Search Strategy*

The term of ‘microservice’ keyword and microservice architecture that found in the published articles journals, and conferences but rest were excluded. Our selected research papers 48 which are published between 2010 and 2017. Selected research papers’ electronic digital libraries are included which are Four (IEEEXplore, ACM DL, DirectScience, ResearchGate, GoogleScholar) (Table I). In this systematic mapping study the selected papers are maximum from the IEEEXplore.

TABLE I. Selected Electronic Databases

Electronic Database	**URL**
IEEE	http://ieeexplore.ieee.org/Xplore/
ACM	http://dl.acm.org/
ScienceDirect	http://www.sciencedirect.com/
GoogleScholar	https://scholar.google.com.pk/

C. Keywords

These keywords which are used for finding all the studies are:

((({Microservice} OR {Monolithic} OR {Traditional}) AND {Architect*}) AND ({System Migrat*} OR {Transform*} OR {Component} OR {API} OR {Cloud}) AND year >= 2010 AND year =< 2017

D. Selection of Primary Study

This section suggests that many studies were deeply checked before the selection of this study. Moreover, relevance to the research question was also given due consideration. At first, the papers were included after carefully reading title and abstract. In case of any ambiguity about the paper in title and abstract section, the researcher reviewed the complete paper by applying inclusion and exclusion criteria.

E. Search Engine

The term of 'microservice' keyword and microservice architecture that found in the published articles journals, and conferences but rest were excluded. Selected research papers' electronic digital libraries are included which are four (IEEEXplore, ACM DL, DirectScience, GoogleScholar). In this systematic mapping study the selected papers are maximum from the IEEEXplore.

F. Inclusion Criteria

- Studies had been published in journals, conferences, and workshops.
- Studies must be written in English.
- Studies must be accessible electronically.
- Collected studies must be published after 2010.
- Research papers will be included which are based on the expert opinion
- Research papers related to the topic, will be included as weak evidence which do not provide evidence

G. Exclusion Criteria

- Non peer reviewed studies (tutorials, slides, editorials, posters, keynotes) are also excluded.
- Peer reviewed but not published in journals, or conferences (e.g. Book, and blogs articles).
- Publications not in English
- Electronically non-accessible.

H. Conducting Mapping Study

Research papers which are published in different conferences or journals that would be a complete version, on the basis of studies, discussed in this article, will be included. Selected primary studies are 48 (Table II). But the further evaluation for these studies researcher has included the studies that are most appropriate to the topic.

TABLE II. ELECTRONIC DATABASE

Digital Library	Publications	Selected
IEEE explorer	208	32
ACM Digital Library	796	4
Science Direct	140	1
ResearchGate	3	1
Other		10
Total	1147	48

1) Challenges of Microservices (RQ1): These challenges are shown in the challenges keyword graph in Fig. 2. Selected papers have discussed about challenges in depth. Researchers have shown a list of open issues in table of current challenges that are open issues of microservices architecture. These open challenges are not discussed in detail in literature. Table IV shows the challenges of microservices.

2) Quality Attributes of Microservices (RQ2): These are quality attributes Scalability, Independency, Maintainability, Deployment, Performance, Reusability, Security, and Load Balancing have been discussed in this mapping study [9], [42]. But researcher have identified few more attributes of microservices which are Reliability, Portability, Availability, these are also important attributes. Other previous quality attributes have been discussed in this mapping study [6], but researcher find out different few more.

3) Motivation for Microservice Architecture (RQ3): Microservices is a new emerging style which is becoming very familiar adopting by industries. It helps the developer to develop the large and complex application to distribute the application in chunks or unit by composing this application [1]. It can be written in language by using APIs for microservices [3]-[5]. Mostly papers are discussing about its independent services that can be upgrade or new addition or services any time. Table III shows the motivations of Microservices.

TABLE III. RESEARCH QUESTIONS

No.	Research Question	Motivation
1	What challenges has been reported in literature about microservices architecture?	This question MSA will elaborate the current challenges. It will discuss in detail about research challenges of microservices.
2	What are the new quality attributes of microservice architecture?	This question aim to identify the new quality attributes of microservice architecture.
3	What are the main motivations for using MSA?	In this question will discuss the benefits of microservices and the aim is to get insight in what are the main reasons for organizations to architect in a microservices style.
4	What are the existing techniques to migrate the application to microservices?	The main to explore this question to highlight the techniques and methods which are helping to migrating the system application from traditional to microservices.

TABLE IV. COMPARISON BETWEEN TRADITIONAL & MICROSERVICES ARCHITECTURE

#	Traditional	SOA	Microservices
1	Single large application	Several applications sharing services	***Small autonomous services***
2	Single deployment unit	Multiple units depending on each other	***Independently deployable units***
3	Limited clustering possibilities	Distributed deployment	***Distributed deployment***
4	Homogeneous technologies	Heterogeneous technologies	***Heterogeneous technologies***
5	Shared data storage	Shared data storage	***Independent data storage***
6	Single point of failure	Single point of failure ESB	***Resilient to failures***
7	In-memory function calls	Remote calls through ESB	***Lightweight remote calls***
8	Single large team	Multiple teams with shared knowledge	***Independent teams owning full lifecycle***

4) Migrating to Microservices (RQ4): There is not proper technique that helps to migrate the complete application to microservices. One paper introduces its method to migrate the application to microservices by independent components but not dependent components [24], [36]. But this method does not help to migrate the complete system to microservices.

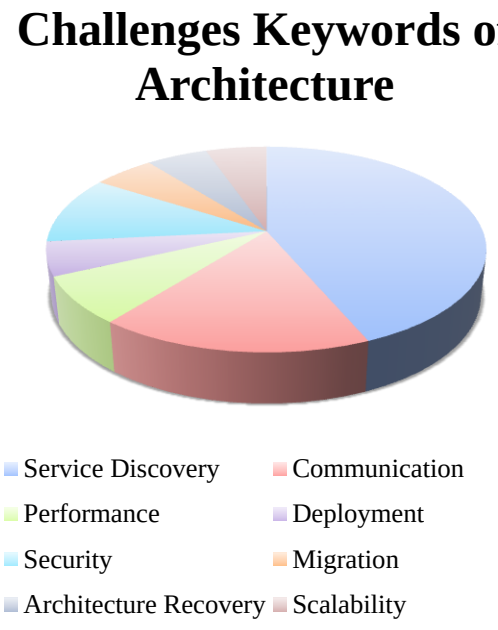

Fig. 1. Related to challenges keywords in architecture.

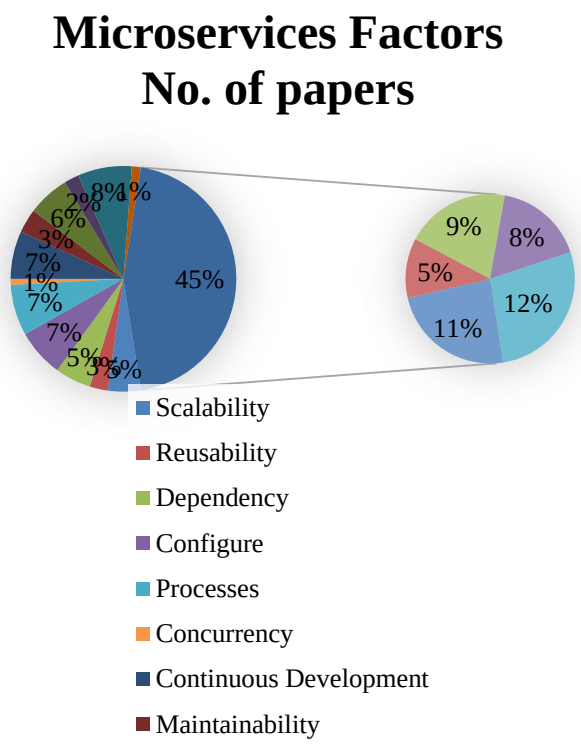

Fig. 2. No. of factors in papers of microservices.

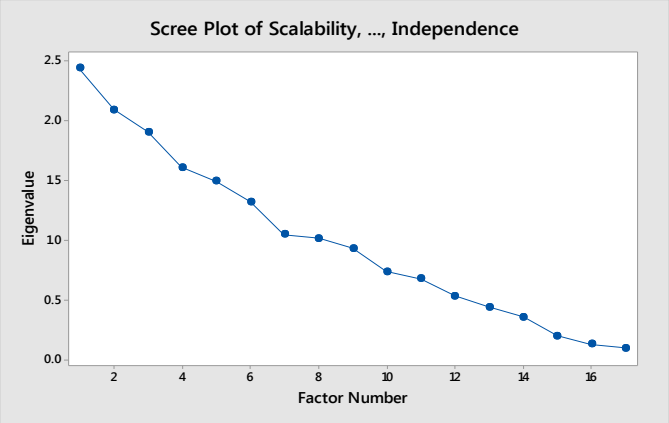

Fig. 3. Microservices factor list analysis variation graph.

5) Discussions of Figures

Fig. 1 shows all the challenges keywords of architecture, this figure shows the growth of keywords Table VI. All these keywords have been reported in literature.

Fig. 2 shows all factors of microservices architecture that can be easily measured. Graph shows the importance of factors by percentages.

Fig. 3 shows the variation of all factors. Researchers use the Minitab stats tool to find the variations of all factors.

Fig. 4 shows the analysis result of all factors that how many papers have discussed each factor.

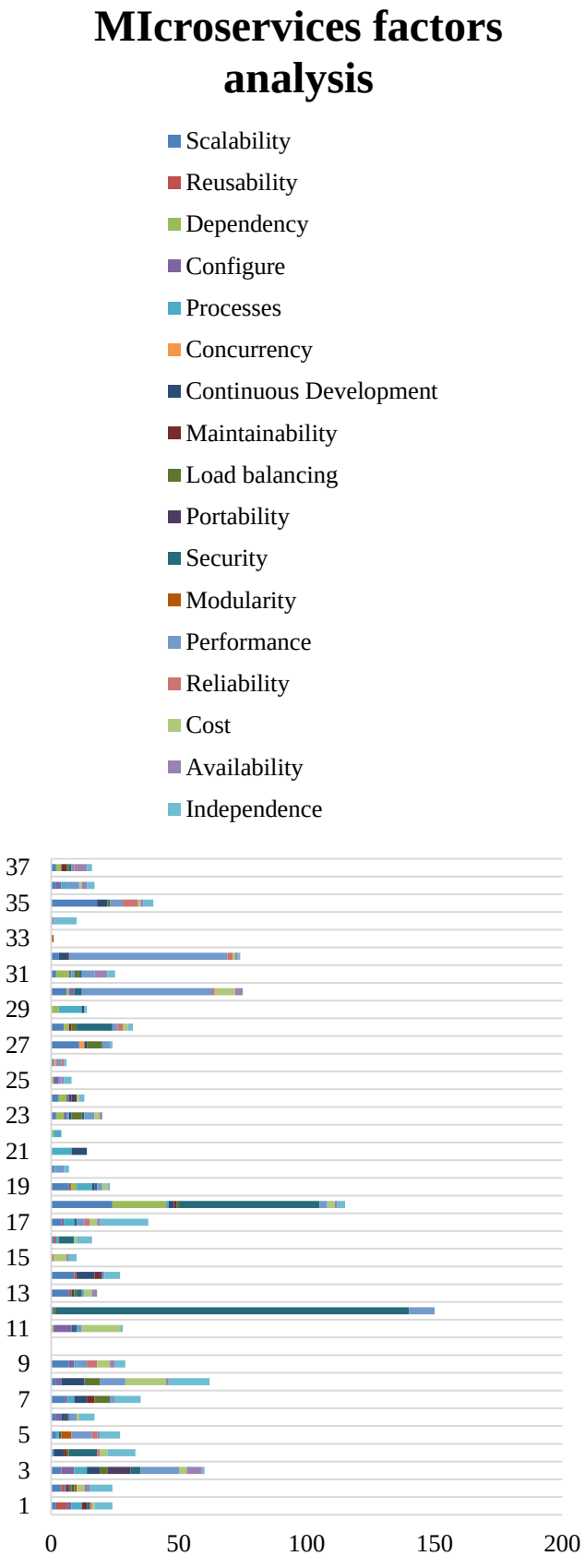

Fig. 4. Factors list of microservices and analysis result graph.

TABLE V. TOP FIVE EMERGING CHALLENGES

#	Challenge	Description	Ref.
1	**Challenges of reimagining and re-architecting a software product.**	It is the big challenge for software architect considering microservices is the need to reimagine and also re-think how the application will work.	[1], [5], [7], [12], [18], [31], [34]
2	**Testing can become challenging.**	Integration testing, it is necessary for the quality assurance engineer to clearly understand each of the different services in order to write the test cases effectively. Debugging meanwhile can mean the QA engineer having to analyze logs across different microservice environments.	[6], [31], [33], [34]
3	**System migration to microservices**	Old system application needs to be migrated to microservices.	[3], [9], [24], [31], [33], [34]
4	**Databases need to be completely decoupled from each other.**	It's easy to be decoupled but also a big challenge because previous database schemas worked with different table by its relations now in microservices it need to be changed this. When transitioning to cloud microservices, you need database models 100% decoupled from each other.	[6], [9], [31]
5	**Performance monitoring under continuous software change**	It's part of microservice architecture that need to be continually changes. Performance does matter very much when following this microservice architecture.	[2], [3], [4], [6], [10], [22], [31], [33], [34]

Fig. 5 found the frequency of all factors that how many papers discussed the each factor.

TABLE VI. FACTORS RELATED TO CHALLENGES

No.	Challenges	Factors
1	**Challenges of reimagining and re-architecting a software product.**	Dependency, Deployment, Configure, Process, Continues, Development, Maintainability.
2	**Testing can become challenging.**	Security, Dependency.
3	**System migration to microservices**	Performance, Independency, Cost, Scalability, Reusability, Continues, Development.
4	**Databases need to be completely decoupled from each other.**	Independency, Load Balancing, Process.
5	**Performance monitoring under continuous software change**	Performance, Continues Development, Availability, Load Balancing, Deployment.

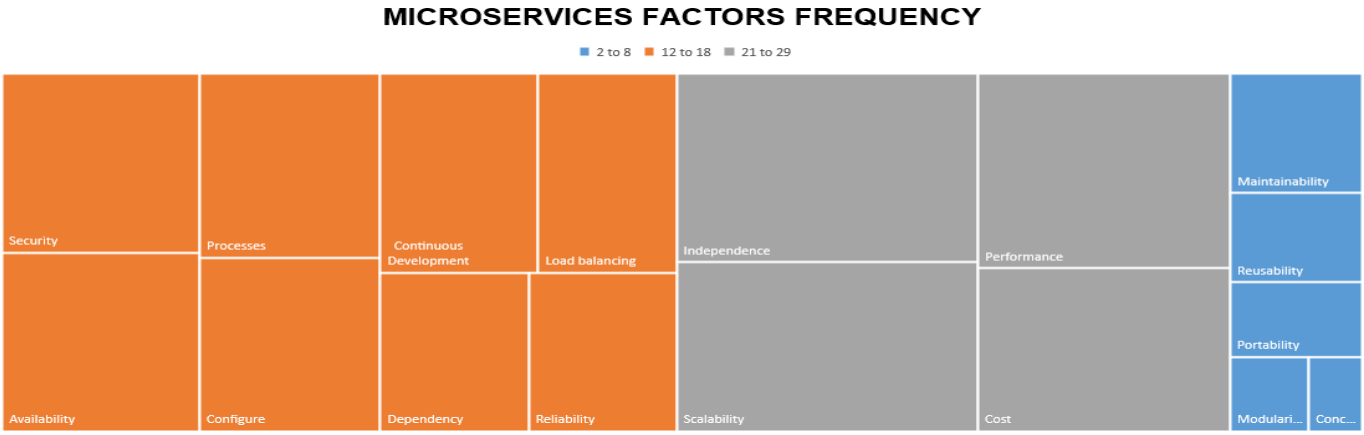

Fig. 5. Frequency of microserives' factors.

TABLE VII. IDENTIFIED FACTORS LIST OF MICROSERVICES

No.	Factors	Paper Reference
1	Scalability	2, 3, 4, 5, 8, 9, 10, 12, 14, 18, 19, 20, 26, 31, 37, 38, 39, 41, 42, 44, 15, 45, 47, 48
2	Reusability	18, 26, 37, 38, 48
3	Dependency	1, 3, 4, 5, 9, 11, 12, 19, 20, 46
4	Configure	1, 4, 8, 10, 12, 19, 20, 29, 37, 38, 39, 42, 44, 15, 45, 46
5	Processes	10, 11, 12, 8, 9, 18, 19, 26, 29, 37, 39, 40, 41, 45
6	Concurrency	2
7	Continuous Development	2, 7, 9, 11, 14, 19, 20, 26, 31, 39, 40, 42, 44, 15, 45, 48
8	Maintainability	3, 5, 9, 37, 38, 40, 44
9	Load balancing	2, 3, 5, 9, 12, 14, 15, 19, 38, 39, 40, 44, 47, 48
10	Portability	20, 31, 39, 42
11	Security	3, 4, 5, 12, 8, 9, 18, 19, 26, 37, 38, 39, 40, 41, 42, 46, 47, 48
12	Modularity	38, 41
13	Performance	1, 2, 3, 4, 5, 8, 9, 10, 12, 14, 15, 19, 26, 29, 31, 39, 41, 42, 44, 45, 46, 47, 48
14	Reliability	3, 4, 5, 7, 8, 14, 31, 37, 40, 41, 45
15	Cost	3, 4, 7, 8, 9, 10, 14, 15, 18, 19, 20, 26, 31, 37, 38, 39, 42, 45, 46, 48
16	Availability	1, 4, 5, 7, 8, 9, 10, 12, 14, 15, 19, 31, 38, 39, 41, 45, 48
17	Independence	1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 14, 15, 18, 20, 24, 26, 29 31, 37, 38, 39, 40, 41, 42, 44, 45, 46

6) Factors analysis method: In this section, researcher used a Minitab tool of stats to analyze the factors of microservices. This tool creates the four different graphs based on analysis result. Graph of scree plot tell us about variation among each factor and show it by dotted line that where the variation is occurring and how much it is. Researcher analyze the result of factors by values, these values mean that how many time each factor is used in literature that is counted as a value of factor for each paper. It means it will help to find the importance of factor and literature focus on factor. Researcher attached the results of all factors in Appendix A section, factor analysis result table (Tables VIII and IX). This research brings forth the number of factors in row and name of factors in column. And in Appendix B section, Fig. 6, 7 and 8 show the result in different view.

IV. Conclusion

This critical evaluation and mapping study has reviewed carefully the given studies on microservices architecture and the relevant architectural challenges reported in literature. Researchers have discussed in details about microservices. Write the planning of mapping study to produce the results that how it will be shown in this study and the major keywords that support to find the literature related to microservices. Research flow diagram is showing the flow or this study the selection of research papers. Research Questions is a major part of this study these are impact on result of this study. The first research question addresses the different challenges in microservices that is shown in Fig. 1 challenges keywords. These challenges keywords are discussed in depth in literature, but open issues are not deeply discussed. The second question discusses about quality attributes of microservices, most of them quality attributes are discussed in previous literature, but the researcher has identified few more quality attributes which are also important in microservices. The third question discusses motivations of microservices that can be seen in literature and comparison Table IV. The last fourth question is very important of this study is migration of system application to microservices, it shows the importance of migration to microservices in the comparison Table IV. Researcher found the list of emerging challenges in Table V. And highlight the factors of microservices in a list form of Table VII then use the Minitab static tool to analyze the factors of microservices and produce the result in the form of quantitative values and different graph. Scree plot is the major graph of this analysis and is discussed above in Fig. 5. Other graph and results are shown in Appendix sections which are Fig. 6, 7, and 8.

V. Future Work

Analyzed material is based on the state of the art research mined for migration, clustering, and services of Microservices. There is need for performing a detailed empirical analysis of system migration based on software industry input to establish a gap between theory and practice.

Further plans include for proposal of a migration technique for practitioners of micro-services, by which guidelines for software firms shall be proposed to increase their scalability with productivity. Future research in the area shall consider comparison of proposed methods to similar methods in literature, using a suitable framework.

References

[1] Luca Florio, Elisabetta Di Nitto. Gru: an Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures. 2016 IEEE International Conference on Autonomic Computing. IEEE, 2016.

[2] Nam H. Do, Tien Van Do, Xuan Thi Tran, Lorant Farkas, Csaba Rotter. A Scalable Routing Mechanism for Stateful Microservices. IEEE, 2017.

[3] Christian Esposito, Aniello Castiglione, Kim-Kwang Raymond Choo. Challenges in Delivering Software in the Cloud as Microservices. IEEE Cloud Computing published by the IEEE computer society. IEEE, 2016.

[4] Hamzeh Khazaei, Cornel Barna, Nasim Beigi-Mohammadi, Marin Litoiu. Efficiency Analysis of Provisioning Microservices. 2016 IEEE 8th International Conference on Cloud Computing Technology and Science. IEEE, 2016.

[5] G. Granchelli, M. Cardarelli, Towards Recovering the Software Architecture of Microservice based system, IEEE, 2017

[6] Nuha Alshuqayran, Nour Ali and Roger Evans. A Systematic Mapping Study in Microservice Architecture. 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications. IEEE, 2016.

[7] Sara Hassan, Andreas Oberweis, Rami Bahsoon. Microservices and Their Design Trade-offs: A Self-Adaptive Roadmap, 2016 IEEE International Conference on Services Computing, 2016.

[8] Zhongxiang Xiao, Andreas Oberweis, and Thomas SchXinjian Qiang. Reflections on SOA and Microservices, 2016 4th International Conference on Enterprise Systems. IEEE, 2016.

[9] Mohsen Ahmadvand and Amjad Ibrahim. Requirements Reconciliation for Scalable and Secure Microservice (De)composition. 2016 IEEE 24th International Requirements Engineering Conference Workshops. IEEE, 2016.

[10] Stefan Haselböck, Rainer Weinreich, Decision Guidance Models for Microservice Monitoring. 2017 IEEE International Conference on Software Architecture Workshop. IEEE, 2017.

[11] Rory V. O'Connor, Peter Elger, Paul M. Clarke, Exploring the impact of situational context – A case study of a software development process for a microservices architecture. IEEE, 2016.

[12] Csaba Rotter, Gergely. Telecom Strategies for Service Discovery in Microservice Environments. IEEE, 2017.

[13] Gholam Reza Shahmohammadi, Saeed Jalili. Identification of System Software Components Using Clustering Approach. 2010.

[14] Wilhelm Hasselbring, Guido Steinacker. Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. 2017 IEEE International Conference on Software Architecture Workshops. IEEE, 2017.

[15] Mario Villamizar, Oscar Garces, Harold Castro. Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud. IEEE. 2015

[16] Abdulaziz Alkhalid, Chung-Horng Lung, Duo Liu, Samuel Ajila. Software Architecture Decomposition Using Clustering Techniques. 2013 IEEE 37th Annual Computer Software and Applications Conference. IEEE, 2013

[17] Duo Liu, Chung-Horng Lung, Samuel A. Ajila. Adaptive Clustering Techniques for Software Components and Architecture. 2015 IEEE 39th Annual International Computers, Software & Applications Conference.

[18] Yale yu, Haydn Silveira, Max Sundaram. A Microservice Based Reference Architecture Model in the Context of Enterprise Architecture. IEEE, 2016.

[19] Giovanni Toffetti, Sandro Brunner, Martin Bl ochlinger. An architecture for self-managing microservices. ACM, 2015.

[20] David Jaramillo, Duy V Nguyen. Leveraging microservices architecture by using Docker technology. IEEE, 2016.

[21] Ibrar Hussain, Aasia Khanum, Abdul Qudus Abbasi, Muhammad Younus Javed. A Novel Approach for Software Architecture Recovery using Particle Swarm Optimization. 2014.

[22] Nam H. Do, Tien Van Do. A Scalable Routing Mechanism for Stateful Microservices. IEEE, 2017.

[23] Ben Horowitz. Website: "Adapting the Twelve‑Factor App for Microservices". July 28, 2016. Copied date: July 13, 2017.

[24] Alessandra Levcovitz, Ricardo Terra, Marco Tulio Valente. Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems. Google Scholar. Website 1, 2, Copied date: July 14, 2017

[25] Bc. Tomáš Livora. Thesis: "Fault Tolerance in Microservices". Masaryk University, 2016.

[26] Sascha Alpers, Christoph Becker, Andreas Oberweis.Microservice based tool support for business process modelling. IEEE, 2015.

[27] Jagdeep kaur, Pradeep tomar Validation of Software Component selection algorithms based on Clustering. Indian Journal of Science and Technology, 2016.

[28] Kamran Sartipi. Software Architecture Recovery based on Pattern Marching. IEEE ICSM.

[29] Matthias Vianden, Horst Lichter, Andreas Steffens. Experience on a Microservice-based Reference Architecture for Measurement Systems, 2014 21st Asia-Pacific Software Engineering Conference. IEEE, 2014.

[30] Suresh Marru, Marlon Pierce. Apache Airavata as a Laboratory: Architecture and Case Study for Component-Based Gateway Middleware. ACM, 2015.

[31] Robert Heinrich, André van Hoorn, Holger Knoche, Fei Li, Lucy Ellen Lwakatare, Claus Pahl, Stefan Schulte. Performance Engineering for Microservices:Research Challenges and Directions. ACM, 2017.

[32] Armin Balalaie, Abbas Heydarnoori, Pooyan Jamshidi. Microservices Migration Patterns.

[33] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara Fabrizio Montesi, Ruslan Mustafin, Larisa Safina. Microservices: yesterday, today, and tomorrow. 2017.

[34] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Migrating to Cloud-Native Architectures Using Microservices: An Experience Report. ResearchGate, 2017

[35] Jyhjong Lin, Lendy Chaoyu Lin, S. Huang. Migrating Web Application To Cloud with Microservice Architecture.

[36] Holger Knoche. Sustaining Runtime Performance while Incrementally Modernizing Transactional Monolithic Software towards Microservices. ACM, 2016

[37] Mazedur Rahman, Jerry Gao. A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development. IEEE. 2015

[38] Alexandr Krylovskiy∗ , Marco Jahn∗ , Edoardo Patti. Designing a Smart City Internet of Things Platform with Microservice Architecture. IEEE. 2015

[39] Hui Kang, Michael Le, Shu Tao. Container and Microservice Driven Design for Cloud Infrastructure DevOps. IEEE. 2016

[40] Gabor Kecskemeti, Attila Csaba Marosi and Attila Kertesz. Microservices validation:Mjolnirr platform case study. IEEE. 2015

[41] Joao Rufino, Muhammad Alam, Joaquim Ferreira, Abdur Rehman. Orchestration of Containerized Microservices for IIoT using Docker. IEEE. 2017

[42] Dong Guo, Wei Wang*, Guosun Zeng, Zerong Wei. Microservices Architecture based Cloudware Deployment Platform for Service Computing. IEEE. 2016

[43] Gabor Kecskemeti, Attila Csaba Marosi and Attila Kertesz. The ENTICE Approach to Decompose Monolithic Services into Microservices. IEEE. 2016

[44] Bj¨orn Butzin, Frank Golatowski, Dirk Timmermann. Microservices Approach for the Internet of Things. IEEE. 2016

[45] Gustavo Sousa, Walter Rudametkin, Laurence Duchien. Automated Setup of Multi-Cloud Environments for Microservices Applications. IEEE. 2016

[46] Srikanta Patanjali, Benjamin Truninger, Piyush Harshand Thomas Michael Bohnert. A Micro Service based approach for dynamic Rating, Charging & Billing for cloud.

[47] Yuqiong Sun, Susanta Nanda, Trent Jaeger. Security-as-a-Service for Microservices-Based Cloud Applications. IEEE. 2015

[48] Tomislav Vresk* and Igor Čavrak. Architecture of an Interoperable IoT Platform Based on Microservices. 2016

FACTOR ANALYSIS RESULT REPORT

APPENDIX A

Factor Analysis: Scalability, Reusability, Dependency, Independence

A. Principal Component Factor Analysis of the Correlation Matrix

TABLE VIII. UN-ROTATED FACTOR LOADINGS AND COMMUNALITIES

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6	Factor7
Scalability	0.109	-0.687	0.507	0.066	-0.263	-0.097	-0.170
Reusability	-0.151	-0.165	-0.495	-0.214	-0.340	0.041	-0.558
Dependency	-0.023	-0.468	0.527	-0.324	-0.273	0.299	0.069
Configure	0.697	0.350	-0.107	-0.075	-0.031	0.338	-0.277
Processes	0.042	0.034	-0.328	-0.558	-0.141	-0.258	0.300
Concurrency	-0.110	-0.218	0.196	0.095	0.729	-0.275	-0.359
Continuous Development	0.561	-0.469	-0.238	0.009	0.058	-0.170	0.392
Maintainability	0.017	-0.626	-0.394	-0.146	-0.241	-0.041	-0.224
Load balancing	0.477	-0.531	-0.041	0.059	0.527	-0.103	0.024
Portability	0.647	0.201	0.087	-0.406	-0.061	-0.373	0.012
Security	-0.165	-0.201	0.418	-0.206	-0.051	0.382	0.263
Modularity	-0.199	0.149	-0.071	0.358	-0.204	-0.248	0.227
Performance	0.345	0.217	0.393	0.241	-0.159	-0.155	-0.000
Reliability	-0.012	-0.077	0.187	0.614	-0.423	-0.340	-0.105

Cost	0.625	0.058	-0.059	0.353	0.027	0.586	-0.037
Availability	0.574	0.201	0.281	-0.130	-0.258	-0.289	-0.157
Independence	0.198	-0.328	-0.542	0.436	-0.156	0.089	0.212
Variance	2.4409	2.0895	1.9048	1.6085	1.4945	1.3187	1.0477
% Var	0.144	0.123	0.112	0.095	0.088	0.078	0.062
Variable	**Factor8**	**Factor9**	**Factor10**	**Factor11**	**Factor12**	**Factor13**	**Factor14**
Scalability	-0.104	0.243	0.123	-0.036	-0.089	-0.080	0.020
Reusability	-0.060	-0.147	0.343	0.035	0.085	-0.154	0.221
Dependency	0.140	0.318	-0.031	0.246	0.020	-0.067	-0.088
Configure	0.073	0.162	0.143	-0.135	-0.108	0.158	-0.207
Processes	-0.307	0.318	0.247	0.053	0.295	0.217	-0.066
Concurrency	0.006	0.128	0.251	-0.020	0.107	-0.072	-0.144
Continuous Development	-0.264	-0.116	0.060	0.049	-0.260	-0.090	0.151
Maintainability	0.184	-0.310	-0.183	0.084	-0.127	0.265	-0.215
Load balancing	0.194	-0.072	0.003	0.017	0.157	0.212	0.138
Portability	0.134	-0.052	0.116	-0.250	-0.199	-0.213	-0.104
Security	0.127	-0.442	0.292	-0.408	0.157	0.063	0.014
Modularity	0.620	0.121	0.431	0.172	-0.146	0.108	0.042
Performance	-0.259	-0.485	0.186	0.425	0.145	-0.029	-0.173
Reliability	-0.292	0.110	-0.007	-0.343	0.029	0.206	-0.006
Cost	-0.117	0.152	0.141	0.097	0.020	0.065	0.130
Availability	0.350	-0.009	-0.280	0.012	0.301	0.008	0.195
Independence	0.178	0.074	-0.050	-0.116	0.311	-0.312	-0.211
Variance	1.0140	0.9291	0.7360	0.6743	0.5294	0.4405	0.3555
% Var	0.060	0.055	0.043	0.040	0.031	0.026	0.021

Variable	**Factor15**	**Factor16**	**Factor17**	**Communality**
Scalability	0.011	0.165	-0.139	1.000
Reusability	0.059	-0.021	0.054	1.000
Dependency	0.050	-0.121	0.130	1.000
Configure	0.021	0.148	0.101	1.000
Processes	-0.035	0.002	-0.043	1.000
Concurrency	-0.187	-0.052	0.039	1.000
Continuous Development	-0.145	0.041	0.117	1.000
Maintainability	-0.110	-0.057	-0.053	1.000
Load balancing	0.255	-0.002	0.009	1.000
Portability	0.091	-0.142	-0.070	1.000
Security	-0.072	0.009	0.009	1.000
Modularity	-0.024	0.006	0.000	1.000
Performance	0.054	0.017	0.001	1.000
Reliability	0.037	-0.101	0.080	1.000
Cost	-0.102	-0.145	-0.114	1.000
Availability	-0.167	0.034	0.020	1.000
Independence	0.017	0.024	0.001	1.000
Variance	0.1976	0.1253	0.0938	17.0000
% Var	0.012	0.007	0.006	1.000

TABLE IX. FACTOR SCORE COEFFICIENTS

Variable	**Factor1**	**Factor2**	**Factor3**	**Factor4**	**Factor5**	**Factor6**	**Factor7**
Scalability	0.044	-0.329	0.266	0.041	-0.176	-0.074	-0.163
Reusability	-0.062	-0.079	-0.260	-0.133	-0.228	0.031	-0.532
Dependency	-0.009	-0.224	0.277	-0.201	-0.183	0.227	0.066
Configure	0.286	0.168	-0.056	-0.047	-0.021	0.257	-0.264
Processes	0.017	0.016	-0.172	-0.347	-0.094	-0.196	0.287
Concurrency	-0.045	-0.105	0.103	0.059	0.488	-0.208	-0.343
Continuous Development	0.230	-0.224	-0.125	0.006	0.039	-0.129	0.374
Maintainability	0.007	-0.299	-0.207	-0.091	-0.161	-0.031	-0.214
Load balancing	0.195	-0.254	-0.022	0.037	0.352	-0.078	0.023
Portability	0.265	0.096	0.046	-0.252	-0.041	-0.283	0.011
Security	-0.067	-0.096	0.220	-0.128	-0.034	0.289	0.251
Modularity	-0.081	0.071	-0.038	0.223	-0.136	-0.188	0.217
Performance	0.141	0.104	0.206	0.150	-0.106	-0.118	-0.000
Reliability	-0.005	-0.037	0.098	0.382	-0.283	-0.258	-0.100
Cost	0.256	0.028	-0.031	0.220	0.018	0.444	-0.035
Availability	0.235	0.096	0.147	-0.081	-0.173	-0.219	-0.150
Independence	0.081	-0.157	-0.285	0.271	-0.104	0.067	0.202
Variable	**Factor8**	**Factor9**	**Factor10**	**Factor11**	**Factor12**	**Factor13**	**Factor14**
Scalability	-0.102	0.262	0.167	-0.054	-0.168	-0.183	0.056
Reusability	-0.059	-0.158	0.466	0.052	0.161	-0.349	0.622
Dependency	0.138	0.342	-0.042	0.365	0.038	-0.152	-0.247
Configure	0.072	0.175	0.194	-0.200	-0.203	0.358	-0.582
Processes	-0.303	0.342	0.336	0.078	0.558	0.493	-0.185
Concurrency	0.006	0.138	0.341	-0.030	0.202	-0.163	-0.406
Continuous Development	-0.260	-0.125	0.081	0.073	-0.491	-0.204	0.425
Maintainability	0.182	-0.334	-0.249	0.125	-0.240	0.601	-0.605
Load balancing	0.191	-0.077	0.004	0.026	0.296	0.482	0.388
Portability	0.132	-0.056	0.157	-0.371	-0.376	-0.483	-0.294
Security	0.125	-0.476	0.397	-0.605	0.297	0.142	0.041
Modularity	0.611	0.130	0.586	0.255	-0.276	0.246	0.119
Performance	-0.255	-0.522	0.253	0.631	0.273	-0.067	-0.485
Reliability	-0.288	0.118	-0.010	-0.508	0.054	0.467	-0.016
Cost	-0.115	0.163	0.192	0.144	0.038	0.147	0.366
Availability	0.346	-0.010	-0.381	0.018	0.569	0.019	0.550
Independence	0.176	0.080	-0.067	-0.171	0.587	-0.707	-0.594

Variable	Factor15	Factor16	Factor17
Scalability	0.056	1.316	-1.480
Reusability	0.300	-0.164	0.572
Dependency	0.254	-0.968	1.383
Configure	0.104	1.179	1.074
Processes	-0.175	0.020	-0.456
Concurrency	-0.945	-0.412	0.418
Continuous Development	-0.732	0.326	1.245
Maintainability	-0.555	-0.454	-0.562
Load balancing	1.289	-0.013	0.099
Portability	0.459	-1.131	-0.749
Security	-0.364	0.075	0.093
Modularity	-0.124	0.050	0.001
Performance	0.275	0.132	0.015
Reliability	0.189	-0.810	0.850
Cost	-0.519	-1.158	-1.215
Availability	-0.847	0.275	0.216
Independence	0.087	0.191	0.007

APPENDIX B

B. Analysis Graph section

Fig. 6 shows us relationship among factors in form of groups. It tells the different relationships in order to positive and negative. This graph allows us to rapidly locate similar observations.

Fig. 7 tells us the co-relationships in two ways among factors horizontally and vertically. It tells us the relationship just between two components.

Fig. 8 shows us relationships among factors in the form of pairs.

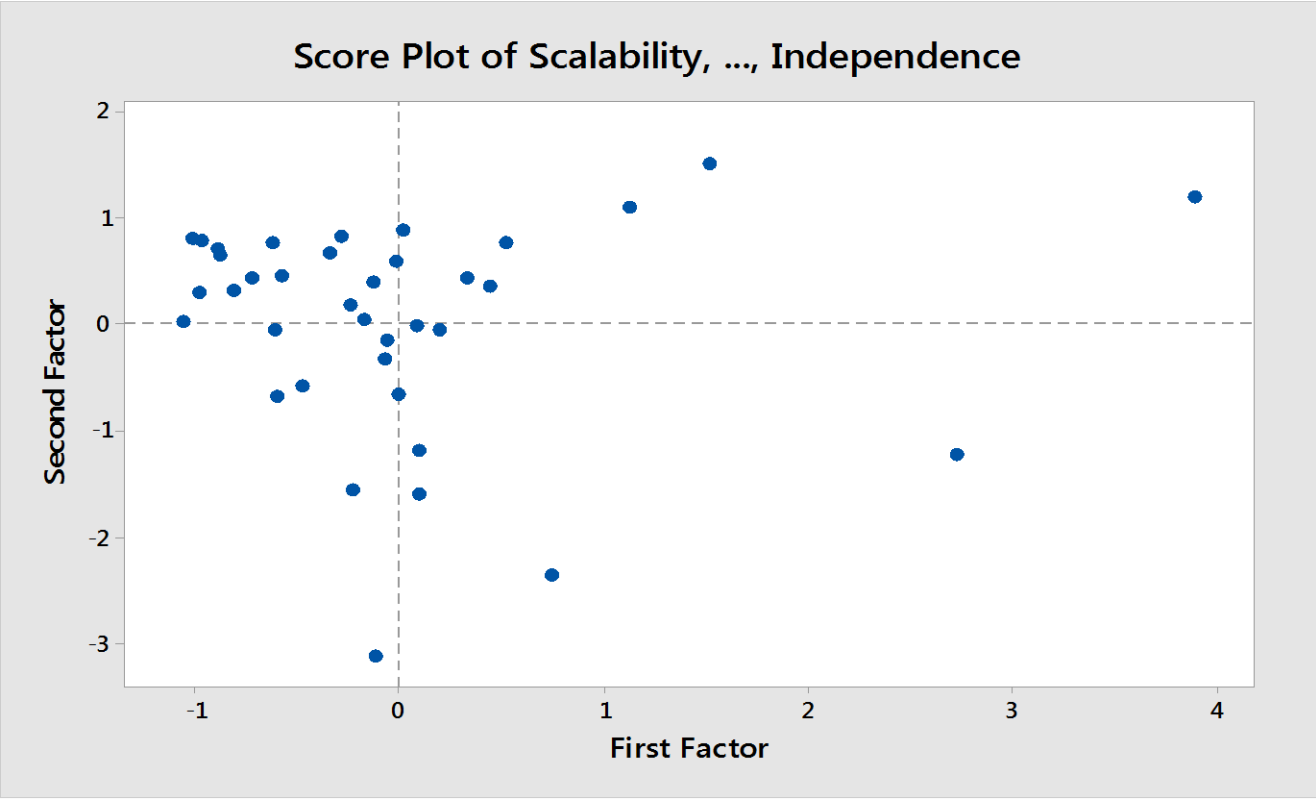

Fig. 6. Score plot.

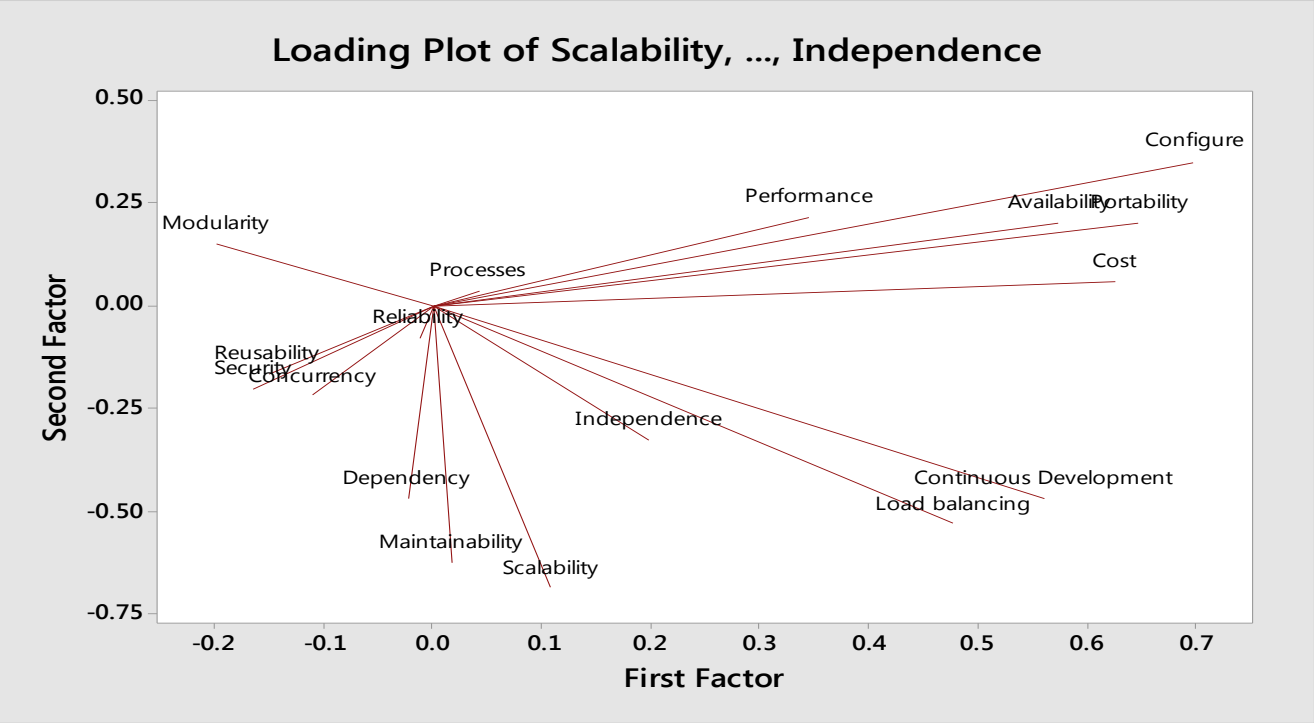

Fig. 7. Loading plot.

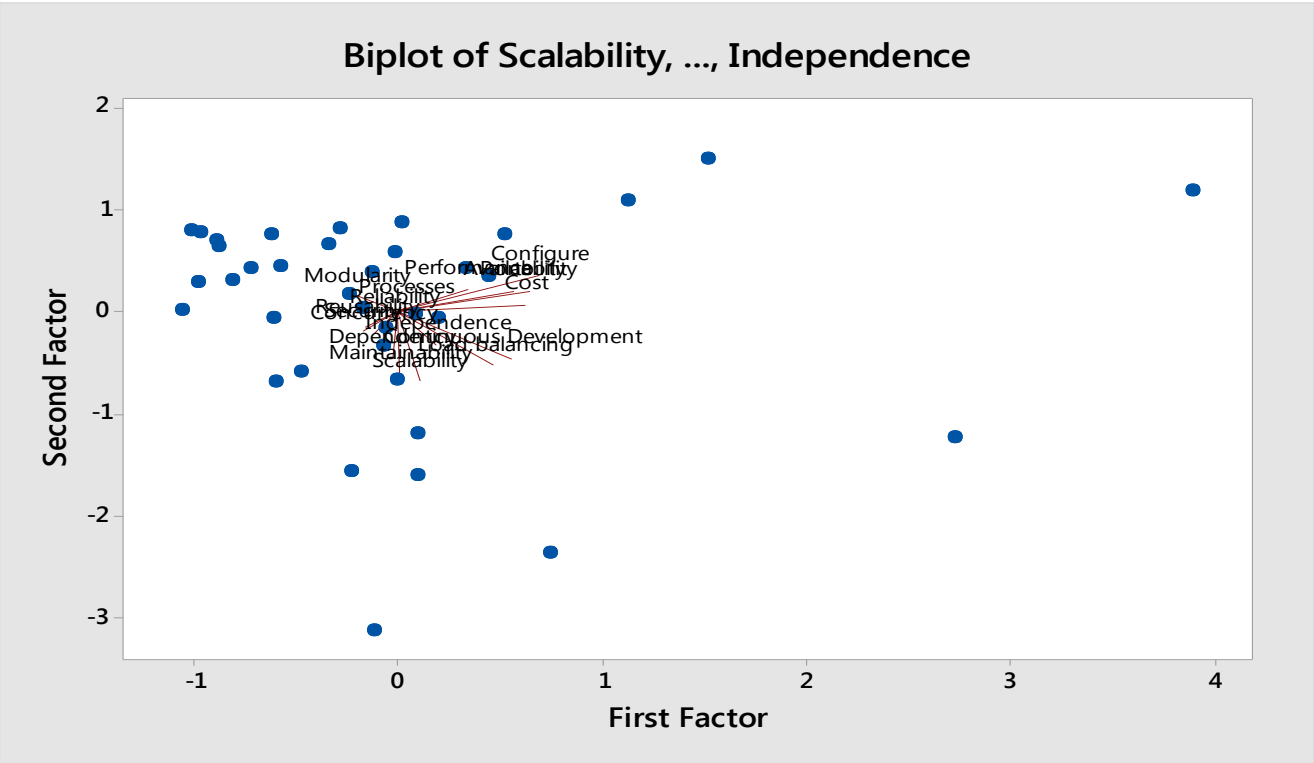

Fig. 8. Biplot.