Smart Computing and Systems Engineering, 2021
Department of Industrial Management, Faculty of Science, University of Kelaniya, Sri Lanka

Paper No: SE-10 **Systems Engineering**

Theoretical framework to address the challenges in Microservice Architecture

Dewmini Premarathna*
Department of Software Engineering
University of Kelaniya, Sri Lanka
dewminic@kln.ac.lk

Asanka Pathirana
Department of Software Technology
University of Vocational Technology, Sri Lanka
asanka.pathirana@gmail.com

Abstract **- Microservice Architecture (MSA) is a recommended way to introduce the application software in a modularized manner instead of the traditional Monolithic Architecture (MA) approach due to the inherent advantages. The MSA is very much effective considering the true benefits of scalability, flexibility, cost-effectiveness, etc. However, there are significant challenges in the use of MSA as well in the viewpoint of the seniors in the field of Software Engineering (SE). So, the objective of this research is to introduce a theoretical framework to be followed by the SE industries to address the challenges they face in providing MSA-based software solutions. In this research, the literature of MSA is evaluated in detail to understand the influencing factors to cater to the requirements of the software developments. In methodology, two research questions are derived based on the hypothesis of not getting adequate benefit in the process of adopting MSA for software application development; 1. What are the challenges to implementing applications incorporating MSA? 2. How to achieve the exact needs of the clients via MSA? For this study, based on purposive sampling the five SE professionals are selected for interviews to understand the true impact on identified factors through literature for development challenges and client satisfaction. Further, thematic analysis is conducted for evaluating those extracts of the interview qualitatively. Nevertheless, the online questionnaire is distributed among a wide range of SE professionals in the domain of MSA implementation for overall understanding about significant factors filtered out through the literature and the interviews, and those were analyzed descriptively. Based on the findings, a theoretical framework is introduced for successful implementation of MSA assuring the clients' requirements. Eventually, this study confirms how MSA adaptation with the theoretical framework is effective for both organizations and clients.**

Keywords - development, framework, microservices, modularize

I. INTRODUCTION

At present, the software industry is a bit more complex due to the evolvement of the technology, progressive demand of the clients, affordability of the customer, complex business requirements, etc. ultimately, the nature of the solutions is also complex catering to different requirements of different audiences[1], [2]. As a result, the software industry is possibly subdivided into different main categories such as product-based, service-based, solutions-based, and research-based. However, the most important consideration of any software solution is its architecture influencing the quality of the final outcome. There are many ways to introduce solid architecture to incorporate specific requirements of the software solution giving priority to exact requirement(s). But any architecture software industry can decide whether it is a single component or a combination of several modules.

Among the many available software architectures, the MSA is a priority consideration for introducing solution architecture either partially or completely because the MSA allows introducing the solution as a collection of smaller services[3]. On the other hand, the MA provides the entire software solution as a single service but it comprises drawbacks in implementation and maintenance perspectives [4], [5]. However, MSA has been introduced to addresses those issues effectively.

Moreover, the solutions-based software industries mainly interact with clients to cater to their emerging requirements, and the software solutions are developed by providing the priority for the client requirements [6]. However, the technical decisions over architecture are made by SE professionals. In some situations, the technical background is also communicated with the client, but with the facts in long run, the final outcome of the particular phase of the development is more focused on [7]. As a result, the client may be suffered in the long run due to extended maintenance and extra efforts is different.

It is compulsory for the client to have an entire understanding of the lifecycle of the use of particular software for making a strategic decision towards selecting the right application software [6], [7]. In other words, the effectiveness of the business process should be improved with the involvement of software solutions by increasing productivity to achieve business objectives. The MSA is a priority consideration for such initiatives so the important factors of MSA are identified in detail in different aspects such as maintainability, scalability, reusability, etc. [3], [5], [8]–[10].

There is a trend in the industry to use MSA due to its benefits, but there is uncertainty whether the organization and client are acquired the true benefits of MSA. MSA also has its own drawbacks associated with distributed services, partitioned databases, infrastructure resources allocation which add extra complexity to the software analysis, design, development, and deployment [10]. The unacceptable or improper usage of MSA also prevents getting its advantages towards the organizations. These reasons may cause the software industry to suffer from various shortcomings throughout the Software Development Life Cycle (SDLC) process. There is a possibility this is indirectly transferred as a cost to the client. As a result, the client ends up with high costs in long run [7]. This paper focuses on those situations and proposes how to satisfy both organizations and clients with the use of MSA on their projects.

Section II discusses the literature about MSA incorporating the reviewed research papers. In section III, the methodology is described and the research design is also extracted from the methodology in the same section. The results and discussions are laid down in section IV, whereas the recommendations are illustrated via theoretical framework next in section V. Then the conclusion is made finally in section VI towards delivering the true benefits of MSA for everyone.

II. LITERATURE REVIEW

The literature is to understand the real value of the MSA and analyze whether those values are properly utilized by the software industry towards delivering appropriate benefits for the clients according to the specific requirements. The main focus here is to have a strong understanding of MSA, its benefits, and its challenges. Literature is mainly categorized into design & implementation, security, deployment, and reporting to understand the benefits and the challenges associate with MSA.

A. Design & Implementation

Incorporating MSA for the software solution is comprised of a mix of both the benefits and the drawbacks as per the requirement of the situation of a client, so it is always challenging to make appropriate use of the required microservices by software engineering professionals [11]–[13]. Some features are required to implement essentially and some others are inherently available with MSA. The important high-level features of MSA are briefly described as follows.

1) Scalability: Scaling is a very important aspect of MSA and it is highly supported for utilizing resources as per the dynamic requirements [14]. To achieve scaling, the solution is introduced as a collection of small services assisting to easily allocate resources upon the requirement of the specific service. Resources such as memory, CPU, disk usage, can be shared within services and more resources will be allocated to those who need it, thus reducing cost [11].

2) Flexibility: MSA has great flexibility in selecting programing language and introduce new human resources into the project effortlessly [8], [10]. If the solution requires more services to develop, the industry has the flexibility to selecting resources at any given time irrespective of its programming skills and which language is used to developed other services [11]. So this is a great advantage that you couldn't achieve from MA.

3) Unit Testing and Integrating Testing: The effect of the unit testing is not much different in MSA and MA domains, but MSA comprises some repeated works (Rahman, and Gao, 2015). Further, the integration testing is relatively more complex in the use of MSA because the involvement of dependent services is significant with respect to MA [4]. As a result, MSA requires more time and effort to complete such testing.

4) Service Discovery: The main function of service discovery is to incorporate new services into the solution [4], [8]. It seems service discovery is an essential element to implement with the solution for large-scale microservices-based solutions because it should automatically detect the services added into the echo system and give zero downtime to the entire system.

5)Circuit Breaker: Circuit breaker is also an essential feature for a solution and it is the approach to isolate the faults automatically to prevent system failures due to an issue with one service. So the main functionality of the circuit breaker is to check the availability of independent services and to start sending requests again upon the availability of the dependent services up [14]. So, this is an additional overhead that developers need to do.

B. Security

In one viewpoint, there is a benefit over security when it comes to the MSA, if one service is open for vulnerability, it is a matter of disabling that and allow the system to run as usual with minimum impact [15]. In another viewpoint, MSA influences security negatively due to network security risk because each microservice communicates over the network via messages. As a result, the internal attacker is in a position to easily find out the message format and try to sabotage the system. Following aspects discuss more details about security aspects.

1) Web Application – Front End: With the MSA there is a need of considering web applications development as small features called micro-front-ends (MFE). So, with the MFE architecture, if one function breaches security or opens for vulnerability, it is easy to disable such functions and the application is available to users with less effect of user experience. On the other hand, the security of each function needs to be validated separately and all the developers who work in parallel on services must have strong knowledge of web application security such as disabling auto-filling on the text fields, masking sensitive inputs typed on the text fields, handling cookies securely in the browser level, keep token like sensitive information in an encrypted format, etc [15].

2) Application Level – Back End: For the micro-frond end architecture there are a set of microservices are available to support backend services as well (Rahman, and Gao, 2015). As mentioned earlier it is an advantage to isolate service open for vulnerability and allow the system to work smoothly. But achieving security standards for each microservice is a more time taking task. Developers, designers, and architects need to think about factors like enabling HTTPS (transport layer security) for intercommunications, secure database connectivity, loading secrets and keys from secure stores such as vaults solutions, etc [15]. Further, some application needs to comply with client's security requirement such as banking guidelines for banking solution. Hence applying these things to all the microservices is required extended time and effort.

3) Source code: Microservices source code is kind of repeating the same security approaches in multiple places. So, the requirements like keeping passwords in encrypted format in property configurations are going to be a big overhead to the network because it is needed to load from a centralized secure store; like vault solutions [15]. Hence, when it is compared with MA source codes security with MSA, has significant complexity.

4) Database: The best practice of MSA is to keep separate databases for each service because when it is required to scale up, the database also can be scaled up separately [8], [15]. If the common database is used for all the microservices, scaling only services is not enough and a bottleneck can occur from the database side. From the security perspective database, administrators have to apply/configure security for databases separately and which requires more time and effort.

5) Vulnerability Assessment and Penetration Testing: For a production-ready application, a final check is to assess vulnerabilities and do a security test which is called penetration testing which covers CSS attacks, SQL injection, CSRF, basically all the security standards are defined by OWSAP application security verification standards [15]. So, the preparation of testing and carry out testing on each developed service and the deployed environment is required extended effort than it is deployed with MA.

C. Deployment

Deployment of MA is very easy because it is required to deploy one or two applications in an application server and high availability can be achieved through horizontally scaling two or three nodes and required a minimum of two databases for failover/replication [5]. But in MSA things are different, it is required more tools like Docker and Kubernetes and the industry needs to build a required skill set to do a successful deployment. The entire deployment process is in five main topics.

1) Docker: Docker is a containerized technology that acts as a small machine and its configuration can be defined by the DevOps engineer or architects to match with particular service requirements. So, each microservice developed for a solution can be configured as containers and can run as small servers [16].

2) Kubernetes: On average, software solution is comprised of a considerable amount of microservices and if those run as Docker containers the same number of small machines are running on top of the infrastructure and managing them might be an arduous task. Hence Kubernetes technology has introduced the capability of managing docker-containers efficiently [8], [16]. So when compared with MA this requires more works to achieve sustainable MSA deployment.

3) Continuous integration and deployment (CI/CD): CI/CD is a most important concern on any development means it helps to automate the building of application and deploy in test, staging, and then production environment [16]. So, the CI/CD process incorporates automation of the build process from development to production environment. When it comes to the MSA building process it requires more configuration.

4) Observability: Observability requirement is consisted of log analytics, distributed tracing, and metrics monitoring. There is a special toolset and most industries use ELK stack for log analytics, elastic APM for metrics, and Zipkin for distributed tracing which is an essential tool for MSA [10]. So, to troubleshoot the issues this setup is required for every deployment, and this is involved more works.

5) Service mesh: Service mesh is a dedicated communication layer that ensures reliable and safe communication between services with high observability[14]. It can handle high-volume communication and uses existing persistent connections to improve performance [17]. Implementing service mesh is not mandatory with MSA but it can add benefits in service discovery, load balancing, encryption, observability, traceability, authentication, and authorization [14]. As service mesh supports circuit breaker, it is no need to develop that feature separately at the sourcecode level.

D. Reporting

Reporting in MSA is a bit complex. Because required data for reporting is in individual microservices [3]. Followings are three approaches that can be used in report generation, and each has its own drawbacks.

1) API-based Reporting: In this approach, reporting service will extract data through API calls from each service and it increases network traffic [8]. Further, the system tends to unresponsive service calls due to hanging if users extract data for long period.

2) Database-based Reporting: In this approach single report service connect to each database owned by other services [8], [12], [14], [15]. Drawbacks that are arisen with the API approach can be overcome with this, but then it breaks the basic principle of MSA because one service is tightly coupled with all other services. If the developer changes any logic or implementation which affects the data structure on a particular service, the report service also needs to be adjusted to address the changes.

3) Message Queue(s) based Reporting: This can be considered as the best approach where each service sends an event to a message queue and report service saves the message into its own database [8]. Then data is available for the reports without affecting any service. Although it is the best approach, extra complexity is added to the environment since additional message brokers need to be managed.

III. Methodology And Research Design

The methodology is introduced for having an overall understanding of the use of microservices to fulfill the requirements of the clients. Then the experiment design is introduced based on derived methodology.

A. Methodology

The background analysis is the initiation for this research with the use of experiences and available literature until enough background understanding is obtained. Then the overall understanding of the influencing factors is achieved for continuing with the interview with SE Professionals. The purposive sampling is used to filter out the 5 key experts who work with MSA due to their comprehensive understanding of MSA, and such data is evaluated based on a thematic analysis approach. Then the questionnaire is introduced incorporating background information and interview findings, and it is shared among the different stakeholders to obtain their opinion in a broader sense. Then responses for the questionnaires are collected for descriptive analysis due to their quantitative nature. As a result, this research approach is a mixed method. Further, findings are organized to recommend a theoretical framework for SE Professionals to use for the betterment of themselves as well as their clients.

B. Experimental design

As per the above methodology, the flowchart in Fig.1 is introduced as an experimental design, and the outcome of this research is a theoretical framework for SE professionals to use as guidance.

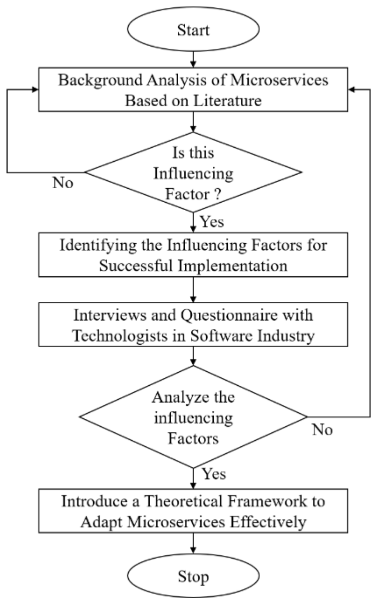
Start
Background Analysis of Microservices Based on Literature
Is this Influencing Factor ?
No
Yes
Identifying the Influencing Factors for Successful Implementation
Interviews and Questionnaire with Technologists in Software Industry
Analyze the influencing Factors
No
Yes
Introduce a Theoretical Framework to Adapt Microservices Effectively
Stop

Fig. 1. Experimental design

According to the flow chart in Fig.1, background analysis was carried out to understand key components of the MSA. Then checked whether that is sufficient to influence the solution that going to propose in this study. This cycle was carried out till the background understanding is enough for the solution. Once it is sufficient, further that evidence was confirmed by using interviews and questionnaires. Zoom was used to conduct the interview with SE industry professionals and the survey was delivered as a Google form. This process was repeated until the gathered information is being satisfied to introduce the theoretical framework to adapt to MSA effectively.

IV. Result And Discussion

The interviews with SE professionals are extracted with important information on the use of MSA focusing on the benefits towards the client, and those qualitative data are evaluated based on thematic analysis. Further, the questionnaire is shared among the stakeholders of the software industry to have an overall understanding of their view towards the same goal as in Fig.2 and those quantitative data is analyzed descriptively.

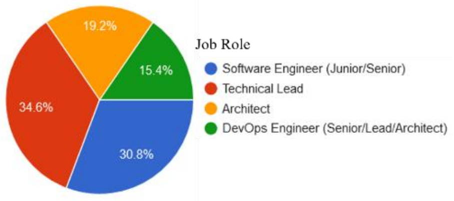

Fig. 2. Contributors for the Questionnaire.

A. Design & implementation requirements

As per the interviews conducted, the following extracts are emphasized to convince the importance of the initial design incorporating the relevant services.

"Representative of the client is a key stakeholder in the software design process" - (SE Professional 1).

The above statement is true once the client is from a non-technical background. However, the level of technical knowledge is reflected on such initiatives as clients can represent themselves throughout the software development lifecycle analysis phase once there is adequate understand of the technology. Such initiatives are positively influenced in addressing the challenges of the MSA implementation.

"Bad designing would cause buying more time for developers"- (SE Professional 2).

Design is important for having a shared understanding between the development team and client from a technical perspective and it streamlines the software engineering development process with clear requirements avoiding reworks. As per the above quotation, it is clear that improper design wastes time due to a poor understanding of the requirements, and it slows down the development process.

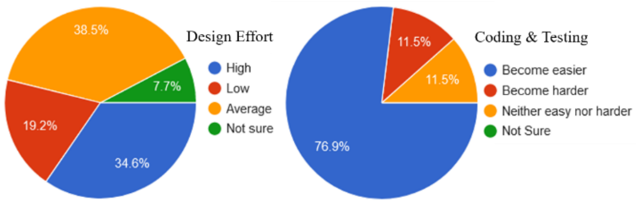

Fig. 3. Design and development phase.

Based on the above understanding, the findings of the survey have also convinced the situation as per Fig.3. 73.1% of responses on design effort are in the average or above level so their primary focus is also on the design. Although there is an extra effort in the designing phase if the industry can manage reusable service repository to reuse the predefined services, it is positively influenced to save more time from coding and testing to deliver true benefit to the client.

B. Security requirements

As per all the interviewees, the required level of security should be achieved via MSA initiatives. The following extract is about the security requirements of the client applications.

"As the services are isolated, securing those are relatively easy but each service should be addressed separately to enrich the level of security" (SE Professional 2).

According to the above statement, the security of each service is assured individually in MSA with the extra effort for the implementation. Eventually, the vulnerability of individual service is not influenced by the others so it is possible to achieve an improved level of security at the end as per the above statement.

Based on the survey findings, Figure 4 also illustrates that better security is achieved in MSA having 80.7% responses on/above the medium level of security. However, the nature of the communication of services by using messages introduces issues as described in section II, and it reflects here having 11.5% responses for low security.

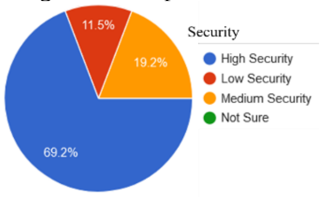

Fig. 4. Security feedback.

C. *Deployment container requirements*

The requirement for the deployment container is emphasized in the interviews as following.

"Better monitoring strategies should be considered during the deployment with properly planned infrastructure, otherwise, maintenance will be hard." (SE Professional 3).

As per the statement, it is clear the SE professionals struggle with monitoring, deployment, and infrastructure utilization support provided by MSA influencing the cost factor of the client negatively due to the maintenance. But it also mentions these concepts need to be properly planned, which means there is a way that we can control the above aspect to improve and give a cost-benefit for the client.

Further, the survey extracts the following information as in Fig.5 with respect to the deployment infrastructure, and 73.1% of responses represent on/above average complexity so it is an important finding on the true complexity of the deployment. As a result, deployment complexity should be addressed with proper tools then clients receive the benefit. Further, infrastructure resource utilization is average/above considering 84.7% of responses in that aspect, so client solutions should be finalized with that understandings.

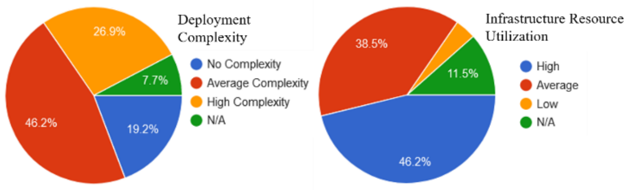

Fig. 5. Deployment Complexity and Infrastructure.

D. *Client requirements*

It is difficult to judge the client and it extracts per the findings of the interviews as follow

"Client is always worried about the price and quality but not the technology. It depends"

- *SE Professional 2*

Understanding the above statement is also clearly illustrated in Fig.6 based on survey findings on how industry experts answered their thoughts about the client expectations. Most of the senior leadership accept clients' most expectation for cost reduction and they do not rely on the underlying technology while some also think infrastructure resource utilization and product quality is equally important.

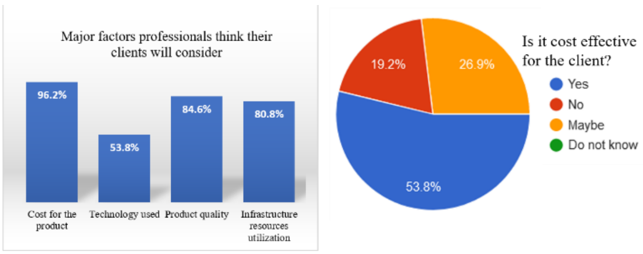

Fig. 6. The perspective of SE professionals about their clients

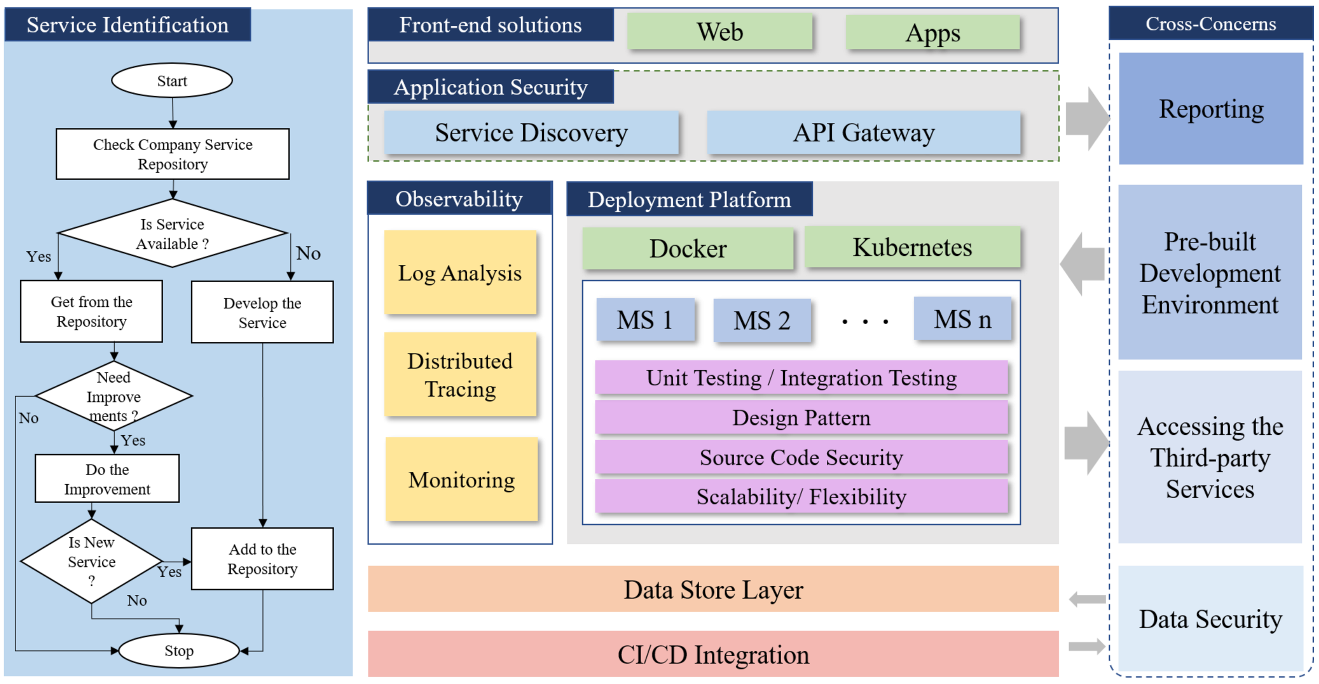

Fig. 7. Theoretical framework for Microservices(MS) developments

As per overall understand, though MSA has challenges, the industry continues on MSA solutions. But those challenges indirectly support to increase in the cost of the projects.

V. RECOMMENDATION

By considering the overall aspects of the use of MSA in different aspects, it is recommended for the organization to follow the basics before moving with MSA addressing the specific requirements of the client in their design of the solution. Incorporating an overall understanding of the findings, a theoretical framework is introduced as a guideline for SE professionals to use for evaluating different possible options available for discussion among all the stakeholders

In Fig.7, the high-level theoretical framework is introduced based on the above literature, survey result, and answers to the interview questions. Further our individual experience is also used when retrieving some components on the introduced framework. SE Professionals can use this framework as a guideline for discussion catering to the exact need of the client appropriately can be adapted. The framework breaks into the following 8 major components; Service Identification, Front-end Solution Layer, Application Security Layer, Observability, Deployment Platform, Data Stored Layer, and CI/CD integration layer. Moreover, the framework is consists of Reporting, a Pre-built Development Environment, Accessing Third Party Services, and Data security.

A. *Service indetification*

As per the interview carried out with industry persons, it is cleared design need get more times and hence developers might facing some issue with delivering implementation on time. Hence Service Identification process is introduced to the theoretical framework so that similarly services can be reuse without spending time on re-developing the same thing. It is a process that an organization should define. Based on the requirement SE professionals need to break down the solution into microservices, once finalized the services that they need to check that defined services are in the organization service repository which is a centralized code management system (e.g. GitLab) and have a full set of functions that microservice can do. So that the few services are utilized from the repository and save the development time. Also identified new services should be developed as reusable components and need to add into a centralized service repository to use by other projects.

Then to speed up development and minimize the re-work pre-build development environment should be available, for example, logging, auditing kind of common concerns should be addressed by developing a library to match with each programming language and need to build into the development environment.

B. *Front-end solution layer*

This layer consists of applications where the end-user interacts. Web and mobile APP can be considered as main applications and sometimes another backend system may be a front-end application. At a high level, any application or system sending requests to the framework can be considered as a front-end application. So when developing these front-end applications if the organization can consider this as micro-front ends, it can be reused in various similar needs so that it will reduce time and effort.

C. *Application security*

Based on the literature review and result of interview answers, it is clear that providing security to each individual microservice is time-consuming work. Hence Application security layer is introduced to the framework so that security can be managed centrally. This layer mainly consists of API gateway and Service discovery. The main function of API gateway is to filter out malicious requests, authenticate and authorize requests before they reach the deployment platform. Also, throttling can be managed from this layer where it can be configured number of concurrent requests allowed for a particular API call. Hence focusing on each individual service's security can be avoided and it will be a huge effort and time-saving for the organization and also benefits can be transmitted to into client as well.

Service discovery controls what are the services available in the deployment platform. So, if any service is added to the platform, it will not be visible to the outside (front-end layer) till that service is added to service discovery. So the service discovery is playing a major role to add services into the platform and remove services from the platform and in that way, it will control service level accessibility.

Once this layer is established there is no additional effort to do with each microservice development and deployment so it will help to overcome the drawback of MSA security concerns and finally it saves a lot of money for the organization.

D. *Observability*

Then the most important part of the framework is to set up the one-time deployment platform and observability. According to the understanding, we gathered from the data analysis it was recognized log analytics and health monitoring of microservice is very important. Observability was added to the theoretical framework to achieve that aspect. There are a lot of open-source tools to configure observability to do log analytics, distributed tracing, and monitoring which includes performance monitoring and stats monitoring. So, when developing the microservices, developers should not worry about the observability and the underlying deployment framework will provide the observability, so that application support after production deployment won't be a hassle anymore and it addresses most of the challenges discussed in the literature. Finally, it benefits the organization in terms of resource and cost.

E. *Deployment platform*

Although MSA is used to strengthen the solution, one key factor we extracted from the interview is if better monitoring strategies were not accomplice when deploying microservice there can arise maintenance issue. To overcome that deployment platform is introduced with the theoretical framework. The deployment platform consists of Docker, Kubernetes, and MSA design partners like circuit breakers and toolset to support the event sourcing especially to full fill reporting requirements. At a high level, individual microservices deploy in docker containers and these docker containers are managed by Kubernetes. Also, Service mesh can be introduced to facilitate and manage service to service communication with fault tolerance way. So if an organization can set up this one-time deployment environment, services deployment will be very easy and all the difficulties face once traditional services deployment will be overcome. Further infrastructure wise it will be huge cost saving when it considers the large scale of solution deployments.

F. Data store layer

There should be a unified centralized place to store data related to developed microservices. Data store layer is added to the theoretical framework to have a completeness over the entire solution when developing MSA. In this layer, an organization can define any relational database like MySQL, PostgreSQL, Oracle, or any NoSQL databases like MongoDB. So this database server is centrally managed and needs to create individual databases inside the server to cater to each microservices unique requirements. So in that case developer, no need to worry about the database management part and a dedicated team will be taken care of the data store layer and which will benefit in every means.

G. Cross-concern layer

In this framework, reporting (auditing), pre-built development environment, accessing the third-party services and data security can be considered as cross-concern where this requires in most of the microservices. So, if an organization can develop common frameworks for these items there won't be any repeated tasks be carried out. For example, if a service requires a report, it should be a matter of enabling a flag in the configuration file or annotate a particular function so that it will automatically start to send some events into reporting service. So, with minimum development effort developer will be able to enable a particular feature. Similarly, if an organization can come up with a common implementation that will give more benefit than the traditional way of development while addressing a lot of challenges faced in the practical implementation of MSA.

H. CI/CD integration layer

This layer will reduce the deployment time which was another concern raised by SE professionals. CI/CD defines continuous integration and continuous deployment. So, with this CI/CD implementation from source code development to applications deployment into production can be automated including executing unit tests, integration tests, code quality checks, code security checks, vulnerability checks, penetration testing, etc. To do this, a pipeline needs to be created and configure according to the requirement. Jenkins is a well-known tool for build automation. So once deployed in the production, the service discovery module should be capable of adding a new service into its registry. By automating this complex deployment process it will be a huge cost saving for any organization when working on multiple projects because now you have a centralized deployment platform to deploy and test the services before delivering to the client which will be benefited for the client in terms of the project cost and it also supports over to overcome deployment complexity currently faced by industries.

VI. Conclusion

There are a lot of researches carries out about MSA and none of them has introduced proper implementation guidelines. So in the literature review, identifies the features of MSA architecture and also what are the limitations, drawbacks, or challenges involved with them. Also, the conducted survey with a specific set of questions identifies how the industry accepts those challenges. Not only that but also conducted interviews with SE professionals by asking specific questions further implies the challenges they see when implementing with MSA. Based on all the inputs, although there are many benefits associated with MSA, it can be unnecessarily complicated due to the different ways in which it is used and some of its limitations. Proper use of technologies with MSA can alleviate those difficulties. But people in the software industry have different levels of knowledge and they provide solutions according to their point of view. Therefore, in some implementations, it is not possible to get the real benefit of it. But if they have some guidance to adapt, they can minimize the difficulties that arise in SDLC. In this research, proposing a theoretical framework as a solution to address each issue theoretically and which will be easily implemented in the practical world as well. Anyone can use it to upgrade every aspect of their organization's SDLC. It will make both organizations and clients are added benefits in time reduction, cost reduction while giving high-quality software with high maintainability.

References

[1] A. Araujo and H. Moura, “Comlexity within Software Development Projects: An Exploratory Overview,” 2015. [Online]. Available: http://lattes.cnpq.br/0902980235660943

[2] J. C. Munson and T. M. Khoshgoftaar, “Measuring Dynamic Program Complexity,” IEEE Software, vol. 9, no. 6, pp. 48–55, 1992, doi: 10.1109/52.168858.

[3] D. Shadija, M. Rezai, and R. Hill, “Towards an understanding of microservices,” Oct. 2017. doi: 10.23919/IConAC.2017.8082018.

[4] Óbudai Egyetem, IEEE Hungary Section, M. IEEE Systems, Hungarian Fuzzy Association, and Institute of Electrical and Electronics Engineers, 18th IEEE International Symposium on Computational Intelligence and Informatics : proceedings : 2018 November 21-22, Budapest.

[5] F. Tapia, M. ángel Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, “From monolithic systems to microservices: A comparative study of performance,” Applied Sciences (Switzerland), vol. 10, no. 17, Sep. 2020, doi: 10.3390/app10175797.

[6] Z. Racheva, M. Daneva, and A. Herrmann, “A conceptual model of client-driven agile requirements prioritization: Results of a case study,” 2010. doi: 10.1145/1852786.1852837.

[7] N. bin Saif, M. Almohawes, and S. M. Jamail, “The impact of user involvement in software development process,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 21, no. 1, pp. 354–359, 2021, doi: 10.11591/ijeecs.v21.i1.pp.

[8] M. V. L. N. Venugopal, “Containerized Microservices architecture,” International Journal of Engineering and Computer Science, vol. 6, no. 11, Nov. 2017, doi: 10.18535/ijecs/v6i11.20.

[9] R. de Jesus Martins, R. B. Hecht, E. R. Machado, J. C. Nobre, J. A. Wickboldt, and L. Z. Granville, “Micro-service Based Network Management for Distributed Applications,” in Advances in Intelligent Systems and Computing, 2020, vol. 1151 AISC, pp. 922–933. doi: 10.1007/978-3-030-44041-1_80.

[10] R. Boncea, A. Zamfiroiu, and I. Bacivarov, “A scalable architecture for automated monitoring of microservices,” 2018. [Online]. Available: http://www.antonkharenko.com

[11] A. de Camargo, R. dos Santos Mello, I. Salvadori, and F. Siqueira, “An Architecture to Automate Performance Tests on Microservices,” in ACM International Conference Proceeding Series, Nov. 2016, pp. 422–429. doi: 10.1145/3011141.3011179.

[12] A. D. M. del Esposte, F. Kon, F. M. Costa, and N. Lago, "InterSCity: A scalable microservice-based open source platform for smart cities," in SMARTGREENS 2017 - Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems, 2017, pp. 35–46. doi: 10.5220/0006306200350046.

[13] N. Herzberg, C. Hochreiner, O. Kopp, and J. Lenhard, "Proceedings of the 10th ZEUS Workshop," 2018. [Online]. Available: https://www.researchgate.net/publication/324517504

[14] S. S. de Toledo, A. Martini, and D. I. K. Sjøberg, "Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study," Journal of Systems and Software, vol. 177, Jul. 2021, doi: 10.1016/j.jss.2021.110968.

[15] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in microservices architectures," in Procedia Computer Science, 2021, vol. 181, pp. 1225–1236. doi: 10.1016/j.procs.2021.01.320.

[16] A. Raj, K. S. Jasmine, and P. G. Student, "Building Microservices with Docker Compose."

[17] "What Is a Service Mesh? - NGINX." https://www.nginx.com /blog/what-is-a-service-mesh/ (accessed Jul. 15, 2021).