Method for Designing Scalable Microservice-based Application Systematically: A Case Study

Ahmad Tarmizi Abdul Ghani, Mohamad Shanudin Zakaria
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
Malaysia

***Abstract*—Microservice is a new transformation of Service-Oriented Architecture (SOA) which is gaining momentum in both academic and industry. The success of microservice began when giant companies like Netflix used them as a service architecture for the purpose of serving customers. Monolithic architecture used by Netflix previously was no longer able to cope with business development and it is difficult to scale to meet user demands. Although Netflix has been successful with microservice architecture, there is no systematic method introduced to produce microservice. Academic studies related to microservice are still in the early stages and have not yet reached maturity. Microservice is seen to require a method that helps organizations to systematically design microservices and replicate the success achieved by Netflix. In forming a method for this systematic microservice then the methods for building an existing microservice are studied. Based on the Design Science Research method, two research artefacts have been produced. The first artefact is a systematic design of microservice that has four main steps. The second artefact is the instantiation by applying the proposed microservice design method to the case studies, namely, MyFlix. Next the evaluation is made on the new method produced by obtaining expert opinions through the process of demonstration and interviewing. The expert assessment results found that the proposed method was able to produce a systematic microservice design based on the six proposed principles and the four main steps. This method can also produce a complete feature microservice such as cohesive, loose coupling, distributed and decentralized that will contribute to the production of scalable system.**

Keywords*—Microservice; systematic method; scalable microservice design

I. Introduction

The company or organization today has an information system that has been operating for a long period of time. This system is constantly evolving and in line with changes in business processes within the company/organization. Upon reaching one level, the system can no longer grow due to physical constraints. Such a system is known as a monolithic system. The monolithic system is an undeveloped and centralized architecture. Such a system is a vendor locked-in system and requires a high cost for system upgrades to meet the increasing system requirements [2]. Although SOA has been introduced to address the problem of isolated system integration, SOA is dominated by large companies such as Oracle, IBM, and Microsoft that produce SOA in the box [2]. SOA in such a box is known as ESB and it is vendor locked and charges high cost to scale. ESB is a monolithic SOA architecture.

Companies that do not use SOA ESB however still use client-server-based (n-tier) systems. Customer-based systems are servers that are centrally and physically constrained. This is what happened to companies like Netflix, Uber and Zalando originally built a client-based system that eventually became a monolithic system. Netflix, Uber and Zalando are examples of ever-expanding companies. The company is a company that provides services to customers online. The increase in the number of users will increase the amount of access to the company's system. Therefore, the system needs to be scaled to meet increased use. If this is not done then the company cannot provide the best service to the customer (for example, the service becomes slow or inaccessible). The adverse effects on user experience will cause the user to quit subscribing to the services provided by these companies. To address the monolithic scaling constraints, organizations or companies need to exit ESB architecture and client-server architecture and find solutions to the problem. The first Netflix has taken steps to scale their systems beyond the client-server architecture. Microservice architecture has been used to develop Netflix systems that can be scaled without physical constraints. As a result of Netlix's success, microservice have begun to get attention from companies and organizations that have similar problems like Netflix. Netflix's success is welcomed by the creation of many tools to build microservices run by Spring Boot, Java EE, Prometheus, Hystrix as well as cloud technology providers such as Google Cloud, Amazon Lambda and Microsoft Azure. The framework for developing this microservice is however a tool that simplifies the transformation to a microservice-based system that is specific to a particular language for example Spring Boot is only specific to Java users. The framework is also a microservice architecture implementation that emphasizes only the aspects of microservice infrastructure such as load balancing, gateway APIs, microservice registries, microservice monitoring over functional aspects of microservice-based applications.

II. Research Background

There are several methods that have been developed for the purpose of producing microservice-based applications. Among them are Domain Based Design Methods [8], [13], [18], Choreography Methods [19], User Interface Based Methods [17], Organization Structure Based Methods [6], [10], Service Cutting Techniques [11], Microservice Dependency

Engineering [16] and Microservice Extraction Techniques [12]. The methods studied have not been emphasized by the end result of a scaled-down microservice application design that results in an cohesive, loosely-coupled, distributed and decentralized [9]. Methods of transforming monolithic applications into microservice-based applications are also found to be part of the information technology division, while the best way to design microservice-based applications should take into account organizational and business aspects [9]. The methods used are still not systematically processed. Systematic method means a method with a system. Systematic methodology means it has a step-by-step guide that must be made by the developer to produce the desired design [14]. Each step has a clear and detailed explanation of the required input, the process to be done and the resulting output. Systematic methods are important because they can guarantee the development of scalable microservice-based applications instead of producing monolithic applications that are not scalable.

III. Research Methodology

The research method used in this study is the Design Science Research (DSR). This method has been used as this study is in the form of design science. Design science uses scientific methods to analyse the structure of the technical systems and its relationships with the environment. It also aims to publish rules in developing systems using the system elements and their relationship [14]. The output of the DSR study is construct-shaped artefacts, models, methods and instantiations. The main output generated from this method is a method of designing a systematic microservice-based application. The following are the phases in the DSR study method used for this study [15]:

A. Problem Identification and Motivations

The research problem has been identified by using Systematic Literature Review (SLR). All the key papers regarding methods and techniques for transforming monolithic system into microservice has been reviewed.

B. Defining the Objective to a Solution

Once research problem has been identified, the objective of the research also need to be defined. In order to solve problems in existing microservice design method, the objective has been set to design a method that can be used to design scalable microservice-based application systematically.

C. Design and Development

Based on the objective to develop a method that can be used to design scalable microservice-based application systematically, the design principles need to be identified first. Then, steps of the method are developed based on the principles.

D. Demonstration

Method developed then has been applied to a case study and then being demonstrated to the experts. The experts are the software developer with more than 10 years of experience. The method used for the demonstration was walkthrough method.

E. Evaluation

The method then evaluated by the experts. There are 5 experts have been interviewed. The experts were first demonstrated with the walkthrough of the application of the proposed method to a case study. The experts then are being interviewed with questions regarding the effectiveness of the method in designing scalable microservice-based application.

F. Communication

Results from the research is then reported in publication to disseminate the knowledge about the proposed method.

IV. Method for Designing Scalable Microservice-based Application Systematically

The proposed method for designing a scalable microservice-based application systematically consists of four major steps (Fig. 1):

A. S1: Model the Organization Unfolding Structure

Step S1 is modelling the organization's unfolding structure. The organization's unfolding structure should be modelled before any other steps can be implemented. The resulting structure represents the actual organizational structure of a company or system. The organizational structure of a company described here does not mean a hierarchical organizational structure but a viable organizational structure [1]. An organization is also a system where the system definitions here are interconnected and working together as an entity [7]. So, in order to control a system then the model should be produced because “every good controller for a system is a model for the system” [5].

B. S2: Model the Business Process

The second step is to model the business process after the model of the organization's unfolding structure/system has been created. The unfolding structure model allows organization/system transformation activities to be identified. This transformation activity means any activity to convert input to output. This activity is also known as the main activity of the organization [7]. Key identified activities will then be mapped to the relevant business processes. Thus, an organizational business process should also be modelled and mapped to the main activities. The difference between the business process and the main activity is the responsible business process of the value chain whereas the main activity is in regards to regulation [7]. The business process can be modelled in many ways. One of the methods chosen in this proposed method is to model with the promise theory. The promise theory is used as it is a new approach in building a distributed and autonomous system [3]. Promise theory emphasizes equilibrium in building a distributed system. It can also prevent the occurrence of situations where dynamics override semantics where conditions can be controlled to a dynamic system and maintain system stability [4]. Since microservices are distributed systems then modelling business processes with promise theories can generate autonomous and centralized microservice and it separates between intent and implementation.

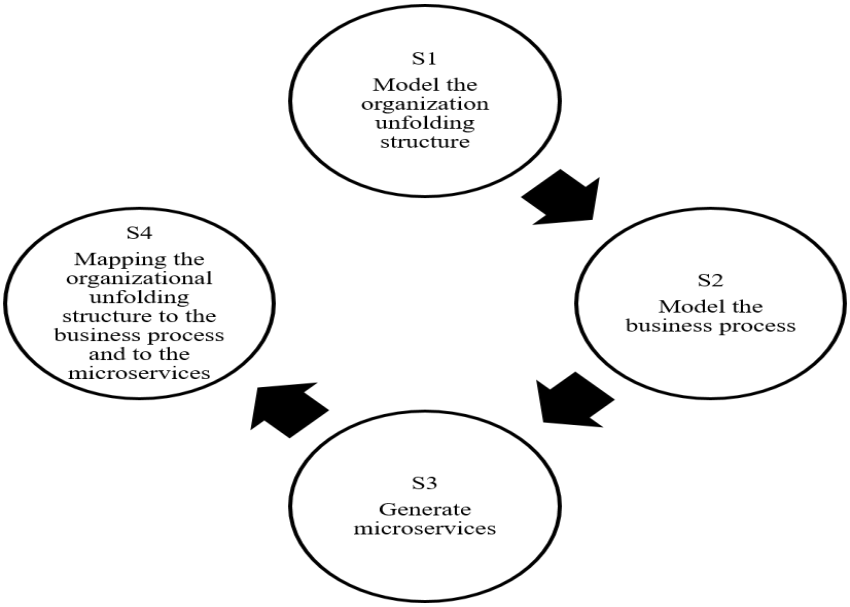

Fig. 1. Steps in the proposed method.

C. S3: Generate Microservices

Based on the business process model using the promise theory that results in the S2 step then the microservice candidate can already be generated. Generating microservice involves earning the name of the microservice candidate as well as the Application Programming Interface (API). APIs are channels for microservice to service users. It is similar to the function in programming that is the function name, parameter/input to the function and output to be generated by the function. The resulting microservices and APIs are not specific to any programming language and they aim to make the resulting microservices implementable in any programming language.

D. S4: Map the Organization's Unfolding Structure to the Business Process to Microservices

The last step is to update the unfolding structure. Artefacts generated from the S2 steps (business process) so that S3 (microservice) should be placed together with the technology branch in the unfolding structure. The unfolding structure needs to be updated to ensure that the unfolding structure is up to date as it will be a reference to any party wishing to subscribe to the existing microservice.

The principles behind the design method are as follows:

1) Principle 1: Systematic

In order to achieve the desired results then there should be a systematic method of systematic methodology, demonstrating step by step in a clear and detailed manner resulting in the desired results [14]. This proposed method emphasizes the creation of systematic design of microservice-based applications so that the desired design results are consistent with the characteristics of a microservice-based application primarily from a distributed and decentralized system aspect.

2) Principle 2: Reducing complexity

System construction is a complex process. The proposed methodology is not to increase the complexity of system construction but to reduce complexity to a degree that can be controlled by system developers [7]. Similarly, the resulting microservice-based application will be a more organized and structured system.

3) Principle 3: Balance

The system based on the microservice generated through this method will be more stable in terms of semantic and dynamic [4]. The paradigm is taken from the promise theory of the importance of the system interacting voluntarily as opposed to coercion. Microservice-based applications need to be modelled with promise theories in order for the system to achieve a balance between the services that interact with each other. It also makes the microservice more reliable.

4) Principle 4: Documentation

The measures taken in this method will produce artefacts that are also documentation of developed microservice-based applications [4]. Documentation is especially important as it will be a reference or a trail to developers who develop the system by himself or a new developer trying to understand the system.

5) Principle 5: The alignment of organizations, businesses and information technology

Microservice-based applications aim to enable the system to evolve in parallel with changes in businesses and organizations [7]. Businesses are changing so fast that they need to meet the needs of new customers. Changes to organizations and businesses mean there will be changes to the business processes that need to be supported by information technology. This proposed method allows microservice to be produced more quickly, less complex, systematic and also stable to meet evolving business needs.

6) Principle 6: Scaled microservice design

The main feature in microservice production is to produce a scalable, cohesive, loosely-coupled, distributed and decentralized service [9].

S1. Model the organization's unfolding structure from complexity drivers
S1.1 Model organizational technology structure
S1.2 Model the organizational geographical structure
S1.3 Model the organizational client/provider structure
S1.4 Model the organizational structure of time
S1.5 Combine the models from step S1.1 - S1.4 to become the organization's unfolding structure
S2. Model the business process
S2.1 For each branch from the root of the organization's unfolding structure resulting from S1.5
S2.1.1 Start from the bottom level technology branch n to level 0
S2.1.1.1 For each level, model the business process for each branch of technology at that level
S2.1.1.1.1 If the level is the leaf level, model the business process by taking into account all types of complexity drivers that will use it at the higher level
S2.1.1.1.2 If it is not a leaf level, model the business process by taking into account all types of complexity drivers that will use it at the higher level and all types of complexity drivers used at the lower level
S2.2 Convert the resulting business process into the form of promise statement
S2.3 Model the promise diagram based on promise statement from step S2.2
S3. Generate microserves
S3.1 List the microservice candidates from step S2.3
S3.1.1 Determine the API for each microservice candidate based on the promise model from step S2.2
S3.2 Model a microservice dependency diagram
S4. Map the organization's unfolding structure to the business process to microservices

Fig. 2. Algorithm of the Proposed Method.

The algorithm for designing the scalable microservice-based application (Fig. 2).

V. Case Study

Chief Technology Officer at F.com (real company name has been hidden) stated the company's desire to develop an application called MyFlix to download videos and watch videos online to paid subscribers. MyFlix applications must be developed using microservice architecture as a company that provides the similar services Netflix has adopted microstructure architecture to develop their system. The company chooses microservice because the architecture is seen capable of being properly scaled and suited to use cloud technology for scaling and also load balances. This service is expected to be supplied to customers around the Southeast Asia region from Malaysia and Singapore. This system should be designed so that it can accommodate high demand at one time without affecting service delivery to the user.

The customer will use the service by becoming a member and making monthly payments through the bank account. The business model used is to serve customers and receive monthly payments is a familiar model for online services today. The process to enable users to register and get the service should be smooth so that users can access the system for immediate service.

The service provided by this company is a search and download service online using cloud technology, stores the video in cloud technology and users can watch the video directly from the internet without having to download video files into their computer or mobile device.

The services provided are intended to enable users to quickly download video files from video providers because cloud technology has high download capability compared to downloading using custom broadband. The user can also download as many videos at the same time and the system will behave on behalf of the user without the user waiting for each download ready. The system also needs to provide the facility to delete video files that are no longer needed. Another feature is the convenience of searching video files on the Internet and continuously downloading them into the system. The system also needs to convert the video format into a format that can be displayed on a web browser or mobile device.

The load on the system should be propagated and better balanced to prevent any failure on the system. The system will later be seen primarily on search, downloading and viewing videos. Therefore, these critical parts need to have a good load balancing and scaling strategy so that the service can continue to be supplied to the user at a user acceptable rate.

Developing a new system is especially risky, especially in this case providing online services that involve a high number of users, a wide range of tools used, users from multiple locations as well as peak times where users access multiple systems. Each one can affect the supplied service. If the system is centrally developed by putting all the services in a single server then there is definitely a service that gets high demand will affect other services. Important service segregation so service that has high demand will not affect other service. Building a microservice-based application requires a different approach than a centralized and non-distributed traditional approach. It requires methods and approaches in which the system can be designed to produce distributed and decentralized services and is able to provide developers with an understanding of the behaviour of the built-in microservice system. It requires a balance between semantics and dynamics.

A. S1: Model the Organization's Unfolding Structure from Complexity Drivers

To simplify the S1 step implemented then the software tool has been used. The software used is Treeline which is the software to generate the main diagram using XML. By using this software all sub steps in S1 can be done directly by using the software. To make it easier for users to replicate the tree structures generated using the software then it also includes an XML schema for each tree-producing step. Next, users can modify tree structures such as adding, updating and deleting to produce an ideal tree structure.

B. S1.1: Modeling Organizational Technology Structure

The model of the MyFlix technology model that has been produced (Fig. 3). The model illustrates MyFlix's organizational structure/system structure comprising three main activities (technology), Membership Services, Browsing Services and the Download Service. Every major activity has sub activities as in Membership Services containing Registration and Payments, Listed Videos Available Video List, Play Video, Download Video Files and Delete Video and Download Services comprises Search technology, List Search Results, Create Playlist, Download and Change Format .

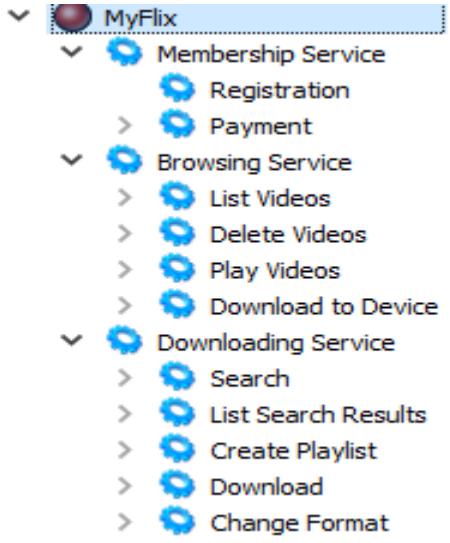

Fig. 3. MyFlix technological model.

C. S1.2: Modeling Organizational Geography Structure

The model of the MyFlix geographical that has been produced (Fig. 4). The model describes the geographic structure of the MyFlix organization/system comprising a geographic drive South East Asia. South East Asia consists of two sub geographies namely Malaysia and Singapore. The geographic drive explains the context where MyFlix's service/technology will operate.

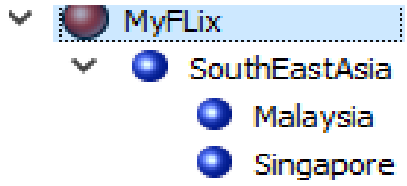

Fig. 4. MyFlix Geographical Model.

D. S1.3: Modeling of Client and Organization Provider Structure

The model of the client tree and the supplier of MyFlix (Fig. 5). The model illustrates the structure of the MyFlix organization/system client consisting of two types of client i.e. Free and Paid. MyFlix's organizational structure/system is comprised of four types of suppliers, Free Video, Paid Video, Shared Video and Subscribed Video.

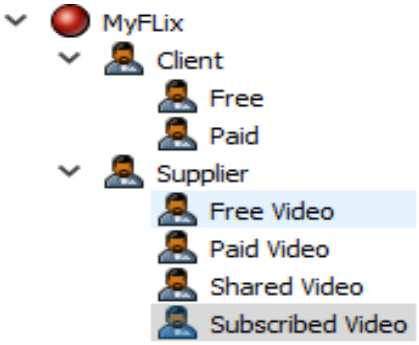

Fig. 5. MyFlix client and supplier model.

E. S1.4: Modeling Organizational Time Structure

The MyFlix tree time model that has been produced (Fig. 6). The model describes the structure of the organization's MyFlix system comprising years and followed by months. This year is the year of MyFlix operation beginning in 2018. Sub year time consists of 12 months from January to December.

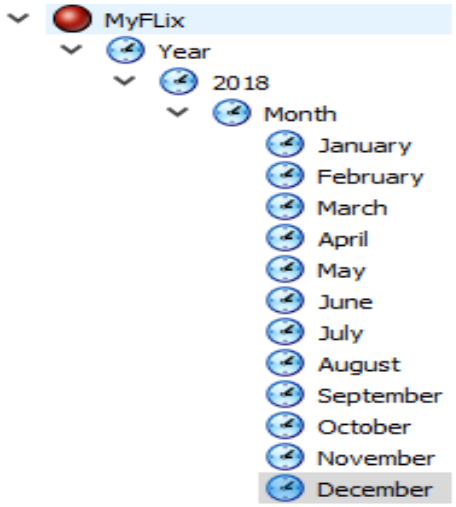

Fig. 6. MyFlix time model.

F. S1.5: Integrate Models from Step S1.1-S1.4 to Become Organization Unfolding Structure

The model resulting from steps S1.2-S1.4 is then used to produce what is known as the unfolding structure of the MyFlix organization/system (Fig. 7). The purpose of this model is to look at the recursive structure inherent in the organization that aims to balance the load. It is much different from the structure as it appears in the hierarchical organizational chart to show unstable power (higher upwards more powerful). The structure of the unfolding structure of the MyFlix organization/system resulting in an overview of the organizational structure is not based on the power but on how an organization is formed based on the elements identified in steps S1.1 to S1.4. In the figure it can also be seen how MyFlix organizations are scaled up in the early stages based on technology, geography, client and suppliers as well as time. This makes the design of MyFlix application scalable.

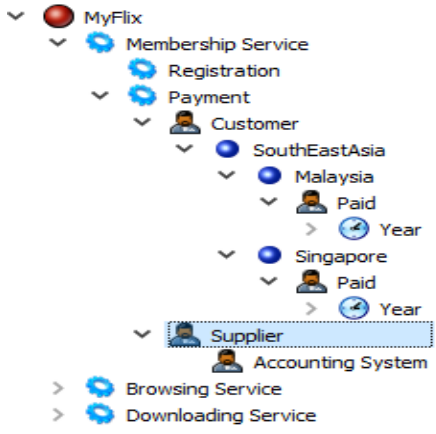

Fig. 7. MyFlix unfolding structure.

G. S2: Model the Business Process

This step is aimed at identifying the business processes involved in each branch of technology. This step can be done by referring to the existing business process or creating a new business process and matched to a recognized branch of technology. Here is the identified business process for the MyFlix technology branches. Step S2 consists of sub steps starting from S2.1 to S2.2.

H. S2.1 for each Branch from the Root of the Organization's Unfolding Structure Resulting from S1.5

In this step, the level of technology of one Membership Service was selected for modelling the business process. Modelling other branches such as Browsing Services and Download Services will be implemented upon completion of the Membership Services branch. It can be run simultaneously by other developer groups. There is no branch sequence that needs to be modelled first.

I. S2.1.1 Start from the Bottom Level Technology Branch N to Level 0

The second level is the bottom level. This level is comprised of a branch of Payment and Payment technology. Then followed by the one level of Membership Services.

J. S2.1.1.1 for each Level, Model the Business Process for each Branch of Technology at that Level

K. S2.1.1.1.1 if the Level Is the Leaf Level, Model the Business Process by Taking into Account All Types of Complexity Drive that Will Use it at the Higher Level

Since the second level is the leaf level then modeling the business process is done on the technology branch of Registration (Fig. 8) followed by Payment (Fig. 9).

L. S2.1.1.1.2 if it is not a Leaf Level, Model the Business Process by Taking into Account All Types of Complexity Drive that will Use it at the Higher Level and All Types of Complexity Drive used at the Lower Level

After completion of the leaf level, the next level is the level of Membership Services (not the leaf level) (Fig. 10). The Business Services business process needs to take into account the business processes of Registration and Payments before modelling it. It also needs to take into account the upper branch of MyFlix. At MyFlix level it involves clients to organizations.

BusinessProcess	Business Process (Outward) 1. Register a new user 2. Update the user's personal information 3. Update user membership information 4. Enable users 5. Disable the user 6. User status

Fig. 8. Registration business processes.

BusinessProcess	Business Process (Outward) 1. Register the paid user payment information 2. Update paid user payment information 3. Terminate monthly charges 4. Provide paid user payment information 5. Provide paid user payment status Business Process (Inward): 1. Create a paid user payment account 2. Charge monthly on a user's credit / debit card 3. Update the paid user payment account after being charged 4. Get paid user payment status

Fig. 9. Payment business processes.

M. S2.2 Convert the Resulting Business Process into the Form of Promise Statement

Once completed modelling the business process for each focused level, the next step is to convert the business process into a promise form. For each branch of MyFlix technology, promise statements are generated for Membership (Fig. 11), Payment (Fig. 12) and Registration (Fig. 13) services. The purpose of the promise statements are to be modelled so that it can convert a coercive business processes to a more voluntary form based on the Promise Theory.

BusinessProcess	Business Process (Outward) 1. User registers 2. User enters payment information 3. User updates personal information 4. User updates payment information 5. User updates membership type Business Process (Inward) 1. Register user 2. Register paid user payment information 3. Update user information 4. Update paid user payment information 5. Get user status 6. Get paid user payment status 7. Enable user if there is a payment 8. Deactivate the user if no payment 9. Update membership 10. Stop monthly charging if the membership is changed to free 11. Get paid user payment information

Fig. 10. Membership business processes.

PromiseBasedBP	Promise (Outward): 1. Promise to allow user to register (+) 2. Promise the user to list the payment information (+) 3. Promise to allow user to update personal information (+) 4. Promise to allow user to update payment information (+) 5. Promise to allow user to update membership type (+) Promise (Inward): 1. Promise to use registration for new user (-) 2. Promise to use payment for the new payment information (-) 3. Promise to use registration to update user's personal information (-) 4. Promise to use payment to update user's payment information (-) 5. Promise to use registration to get user status (-) 6. Promise to use payment to get paid user payment status (-) 7. Promise to use registration to disable user if user does not make payment (-) 8. Promise to use registration to enable user if user make payment (-) 9. Promise to use registration for membership update (-) 10. Promise to use payment to stop monthly charges if membership is changed (-) 11. Promise to use payment to obtain paid user payment information (-)

Fig. 11. Membership promise statements.

PromiseBasedBP	Promise (Outward): 1. promise to register paid user (+) 2. promise to update paid user payment information (+) 3. Promise to stop monthly charges (+) 4. Promise to provide paid user payment information (+) 5. Promise to provide paid user payment status (+) Promise (Inward): 1. Promise to use the financial system to create a paid user account (-) 2. Promise to charge a paid user credit / debit card every month (+) 3. Promise to use the financial system to update paid user accounts after being charged (-) 4. Promise to use the financial system to obtain paid user payment status (-)

Fig. 12. Payment promise statements.

PromiseBasedBP	Promise (Outward) 1. promise to enroll a new user if the user does not already exist (+) 2. promise to update the user's personal information (+) 3. promise to update user membership information (+) 4. promise to enable user (+) 5. promise to deactivate the user (+) 6. promise to provide user status (+)

Fig. 13. Registration promise statements.

N. S2.3 Model the Promise Diagram based on Promise Statements from Step S2.2

The promise statement resulting from step S2.2 is used to model the promise diagram. This diagram is very useful to see the balance of interactions between agents (technology branches). The promise theory emphasizes equilibrium and dynamic aspects of equilibrium. It also emphasizes the convergence of interacting agents that are important to produce a stable system. The theoretical model of the promise that has been made between the Membership Service with Registration and Payment (Fig. 14). This is to indicate that there is an interaction between Membership Services with Registration and also what promises are exchanged between the two agents. The promise statement must have a symbol (+) meaning the promise given while (-) means a promise of use. It can also be seen in the figure of the interaction between Membership Services and Payments.

O. S3: Generate Microservice

Step S3 aims to produce microservice after the completion of the S2 step is implemented. The microservices to be produced are services that have APIs. The resulting microservices are general and can be implemented in any microservice framework framework such as Spring Boot.

P. S3.1 List the Microservice from Step S2.3

Microservice candidates will be taken from the model diagram produced in S2. Agencies that promise and use promise are eligible as microservice candidates. In the case of MyFlix, for the technology branch of the Membership Service, microservice are:

1) *Membership service*
2) *Registration, and*
3) *Payment.*

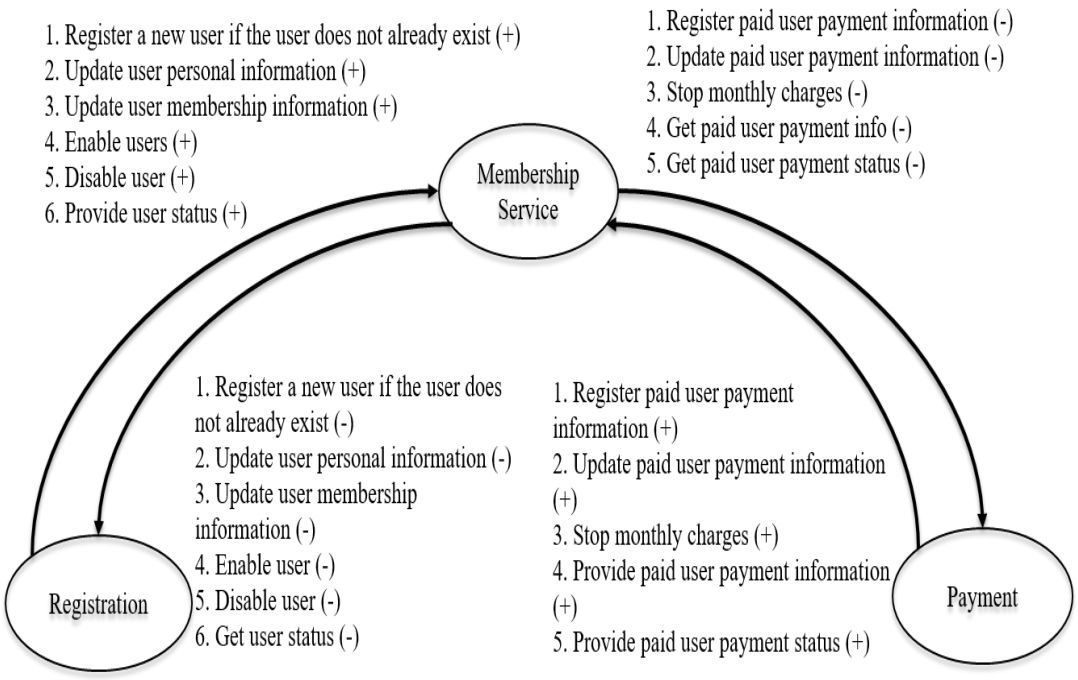

Fig. 14. Promise diagram of membership service.

Q. S3.1.1 Determine the API for each Microservice Candidate based on an Promise Model from Step S2.2

After the candidate microservice are identified in Step S3.1 then the next step is to determine the API for the candidates of microservice. Referring to the model of a promise such as API diagram can be named based on the promises of one agent to another agent. For example, Registration promises to the Membership Service to "register a new user if the user does not already exist (+)" then the API for the Registration microservice can be named with "registerUser()" but still refers to the same promise by Registration to Membership Service. The resulting APIs from S3.1.1 step for Membership (Fig. 15), Payment (Fig. 16) and Registration (Fig. 17) services.

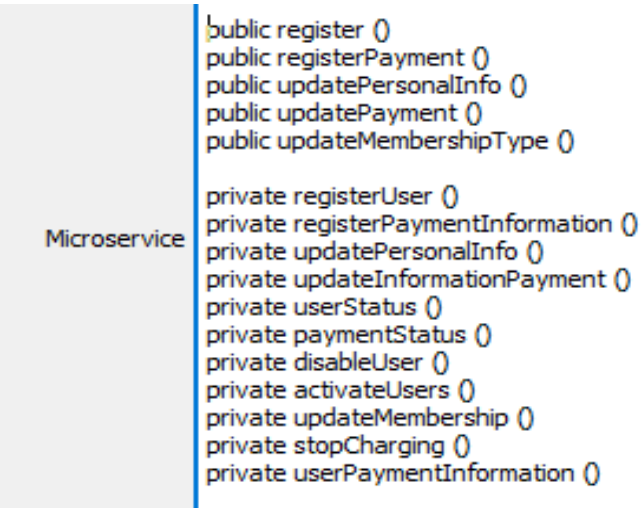

Fig. 15. Membership microservice API

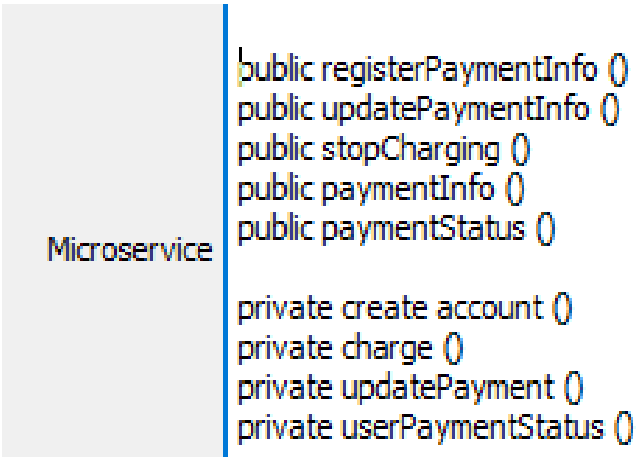

Fig. 16. Payment microservice API.

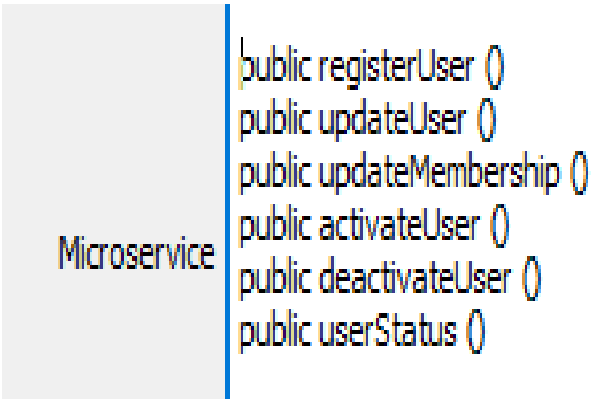

Fig. 17. Registration microservice API.

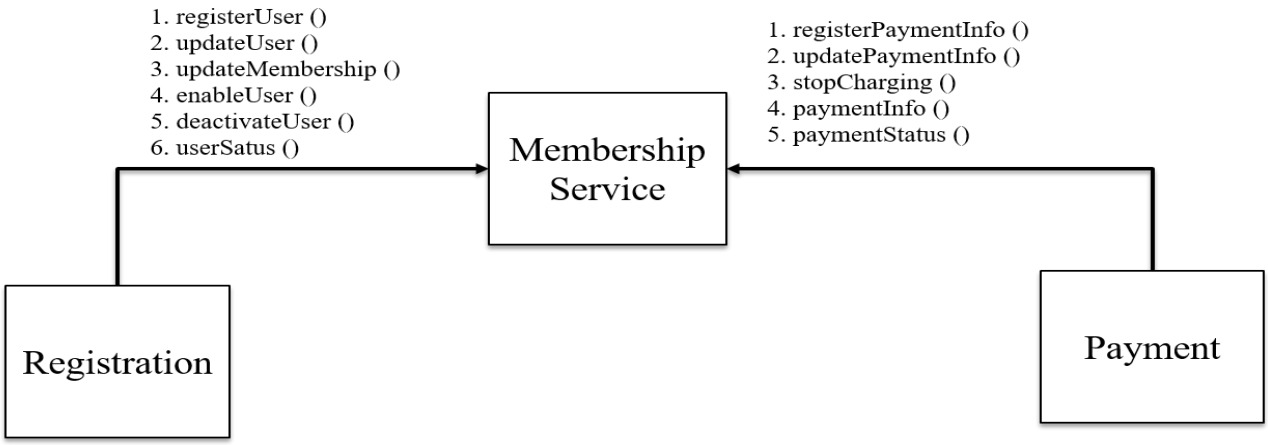

Fig. 18. Membership dependency diagram.

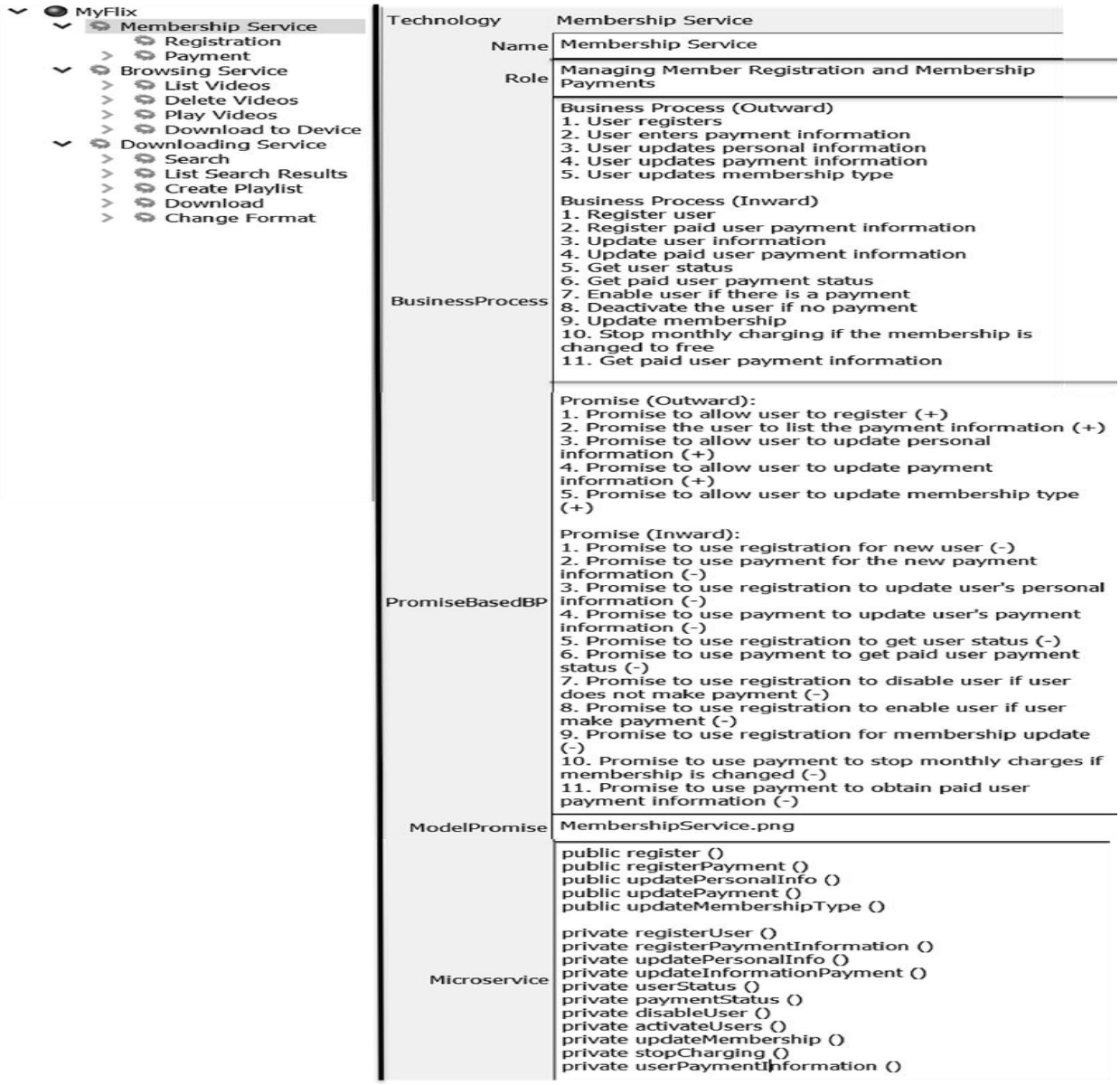

Fig. 19. Aligning organization to business processes and microservices.

R. S3.2 Model a Microservice Dependency Diagram

It is possible to see how a microservice depends on each other (Fig. 18). In microservice architecture, the two main types of interactions that are the first are synchronous and the second is as invisible. Examples of synced interactions are the use of keywords such as Get, Post, Put and Delete which are the dialect used for REST interactions. Examples of asynchronous interactions are by using technologies such as pub/sub. It is up to the builder to implement microservices either asynchronously or synchronously.

S. S4: Map the Organization's Unfolding Structure to the Business Process to Microservices

The last step is to map the unfolding structure of the MyFlix organization/system to the business process and to the microservice that have resulted from the S1-S3 step. This step is important as it will align between the organization, the bisness and the microservice. The result is a fully-documented MyFlix-based application structure and ready to be updated in the next iteration. It can also be considered as a knowledge base as well as a map to MyFlix applications which can be consulted and understood by developers anytime and anywhere MyFlix application developers are located. The mapping of the MyFlix unfolding structure with business processes and microservices for the branch of Membership (Fig. 19).

VI. Results

Experts have been interviewed to give expert opinion on the methods that have been proposed. The experts background (Table I).

The number of experts is only 5 because of the difficulty to get experts who are familiar with microservice architecture in Malaysia. However, even though only 5 experts have evaluated the method, the experts chosen represent developers from the industrial and also from the government sectors that has long experience in developing applications. Before the interviewing process, the experts have gone through the demonstration of the method of the case study in Section V. The process is called a walkthrough process. Then, interview has been conducted to get as a way to evaluate the proposed method and to get the expert opinion. Based from the interview with the experts, the results gathered from the interview are as follows:

TABLE I. Experts Background

Expert	Sector	Experience as Software Developer
1	E-Business	14 years
2	Software House	15 years
3	Government	13 years
4	Government	13 years
5	Government	11 years

1) All experts agree that this method would be able to design microservice-based application systematically.

2) All experts agree that this method would be able to reduce the complexity of developing distributed system.

3) All experts agree that this method would be able to design a stable microservice-based application.

4) All experts agree that this method would be able to effectively document the design process.

5) All experts agree that this method would be able to align the organization with the business and the information technology.

6) All experts agree that this method would be able to design a scalable microservice-based application.

Among key insights gathered from the interview with the experts are as follows:

1) The method can be used by developers in the industrial and government sectors for transforming existing monolithic application that are not scalable and expensive to maintain.

2) In developing complex system, this method is important to help developers to first design before rushing into the implementation as the latter can cause system instability and spaghetti system and the end result is a monolithic application.

3) The method not just help to align the IT to business but most importantly aligning to the organization structure. Therefore, it is important to first design a distributed and decentralized organization structure that confirm will produce a loosely coupled but cohesive business processes and microservices.

VII. Future Research

The artefacts produced in this study can be used and tested for their efficacy in diverse case studies. In this study, this solution method is tested on a case study that provides online service-based applications constrained by monolithic architecture. The outcome of a case study in this study can be a benchmark for other studies that use different cases from the case in this study.

The method that has been developed from this study is specifically to produce a microservice-based application design that can be implemented in microservice architecture. However, this method is also likely to be modified for use in research involving the creation of any distributed and decentralized system design. For example, studies in Internet-of-Things (IOT) involve any tools such as hardware and sensors that need to be designed so that they can achieve their intended purpose. The study in this area is still new and the solution method from this study is seen to have potential for use in the IOT field. Apart from the IOT field, any area requiring distributed and decentralized system designs can also take advantage of this method as in cyber security, blockchain and business modeling.

VIII. CONCLUSIONS

This study is an attempt to produce an artefact in the form of a method that can be used to design scalable microservice-based application systematically. Starting with the problem of lack of methods to produce scalable microservice-based application designs it has been suggested in this study to find a method that can be used to produce scalable microservice – based application design. The result findings have shown the method capabilities to be used to model a scalable microservice-based application design. The principles behind the method has made the proposed method to be able to deliver not just scalable design of microservice-based application but also taking into consideration the stability of the application that will be produced. However, the method produced from this research is limited only for designing a scalable microservice-based application without considering other aspects such as cost, security and human.

The contribution of this method is very important as existing methods are still not systematic and do not guarantee the development of scalable microservice-based applications. This method is designed specifically for developers to design microservice-based applications that will guarantee the development of scalable microservice-based applications. It is also important because it can align between organization, business and information technology for the sake of organization viability.

REFERENCES

[1] Beer, S. 1979. The heart of enterprise. Managerial cybernetics of organization. Wiley.

[2] Bhadoria, R. S., Chaudhari, N. S. and Tomar, G. S. 2017. The Performance Metric for Enterprise Service Bus (ESB) in SOA system: Theoretical underpinnings and empirical illustrations for information processing. Information Systems 65: 158–171.

[3] Burgess, M. 2015a. Thinking in Promises: Designing Systems for Cooperation. O'Reilly Media.

[4] Burgess, M. 2015b. In search of certainty : the science of our information infrastructure.

[5] Conant, R. C. and Ross Ashby, W. 1970. Every good regulator of a system must be a model of that system. International journal of systems science 1(2): 89–97.

[6] Conway, M. E. 1968. How do committees invent. Datamation 14(4): 28–31.

[7] Espejo, R. and Reyes, A. 2011. Organizational Systems: Managing Complexity with the Viable System Model. Springer Berlin Heidelberg.

[8] Evans, E. 2013. Domain-Driven Design Quickly. Journal of Chemical Information and Modeling, p. Vol. 53.

[9] Friedrichsen, U. 2017. Resilient Functional Service Design. InfoQ.

[10] Gucer, V., Narain, S. and others. 2015. Creating Applications in Bluemix Using the Microservices Approach. IBM Redbooks.

[11] Gysel, M., Kölbener, L., Giersche, W. and Zimmermann, O. 2016. Service Cutter: A Systematic Approach to Service Decomposition. European Conference on Service-Oriented and Cloud Computing, p. 185–200.

[12] Levcovitz, A., Terra, R. and Valente, M. T. 2016. Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems.

[13] Newman, S. 2015. Building Microservices. O'Reilly.

[14] Pahl, G., Wallace, K., Blessing, L. T. M., Beitz, W. and Bauert, F. 2013. Engineering Design: A Systematic Approach. Springer London.

[15] Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. 2007. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems.

[16] RV, R. 2016. Spring Microservices. Packt Publishing.

[17] Timms, S. 2016. Mastering JavaScript design patterns. Packt Publishing Ltd.

[18] Vernon, V. 2013. Implementing Domain-Driven Design. Pearson Education.

[19] Yahia, E. B. H., Réveillère, L., Bromberg, Y.-D., Chevalier, R. and Cadot, A. 2016. Medley: An Event-Driven Lightweight Platform for Service Composition. International Conference on Web Engineering, p. 3–20.