A Method for Analyzing and Designing Microservice Holistically

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

***Abstract*—Microservice is a new architecture that is getting attention in the development of service systems. However, microservice is still at the early stage and the acceptance of this architecture is overwhelming. Microservice architecture is a promising architecture in delivering loosely coupled, decentralized, and scalable system that utilizes the latest technology, such as container and cloud computing. However, the traditional method for analyzing and designing system will not be able to fully utilize the capability of the microservice architecture. Therefore, a new method for analyzing and designing the microservice holistically is being proposed in this paper. The Design Science Research methodology has been adopted in designing the proposed method. The artifact, which is the result of the research, is the proposed method. The proposed method has shown its potential in being used to analyze and design the microservice holistically and to benefit from the microservice architecture capabilities.**

***Keywords*—Microservice; service design; promise theory; viable system model; Viplan method**

I. Introduction

The growth of technology such as cloud computing, the Internet of Things (IOT) and mobile technology has created a new challenge in designing information systems, particularly the service systems [13]. People are using and getting services out of these technologies. These technologies have challenged the constraints of space and time as they are being utilized exponentially and the trend is growing. Previous knowledge with regard to system analysis and design has come to an end as the knowledge caters only for systems that are designed in a manner where they are tightly coupled, non-scalable and centralized.

Present day systems must be designed in the form of a more loosely coupled manner that can work together as a unit [14]. There is no force in making these separate systems to work together. It is only how the behavior of each separate system (or subsystems) is perceived, and then by using the promise (such as the contract used in the web service) provided by every system, that a whole new system will emerge. This whole new system, which is made up of loosely coupled parts (or subsystems), interact and work together voluntarily and will have its new behaviors [7]. However, this new system cannot be controlled directly because it consists of so many parts that work together. In order to help control the system, the use of constraints to the behavior which can be configured as shown by the system is thus performed [8]. The use of constraints will enable the system to have self-control and self-regulation instead of the need to be controlled and monitored by humans. This is called the autonomous system.

Humans have the temptation of controlling everything, and so does the service system. It is easy to control small, not-so-complex, and predictable systems. However, with the growth of the present day service system, it is getting more impossible for humans to control the complex system. Hence, humans must give up the control and let the system to control itself. It is rather like a need to adopt the knowledge from the cybernetics area in order to design this new type of analysis and design in creating the autonomous system [1]. The results of letting down of human control of the system are that the system will become more autonomous, scalable (time and space), faster, reliable and durable.

II. Background

In designing this new method, a number of existing theories whether old or new, and also theories in other fields have been revisited and adopted. Among those theories are the information theory, the control and communication theory, variety engineering [2], and self-organization [3] theory from cybernetics, the Viable System Model [4], and the Viplan method [10] from the organizational theory and the promise theory [5].

The reason why those theories are adopted is because each of the theory has its own role in developing the method that is going to be proposed in the next section. Overall, this method is based on the Viplan method which provides the foundation in building the holistic self-organized system. The Viplan method itself is based on the Viable System Model which is the model for building an autonomous system which is able to adapt to its environment and can change accordingly in order to survive. The Viplan method is an established method that has been used in designing viable organizational system based on VSM. The Viplan method on the other hand, provides knowledge on how organizations can be structured to be viable based on the identity of the organization, and also the identification of the primary and the support activities in the organization that will respond to the environment. Then, the business processes can be identified and mapped to the primary and the support activities. Apart from that, the way information is passed among the business processes must also be recognized since information is important in ensuring the viability of the whole organization. There is no centralized control in the organization developed using the Viplan method.

The latest theory adopted for the design of this new method is the promise theory, which was pioneered by Burgess [8]. This theory is developed based on the knowledge of quantum physics, and has been successfully implemented as a configuration software known as the CFEngine. The concept of the promise theory is to break things into parts and to make it work together as one. It is a bottom up paradigm where different parts can interact with each other and work together as new whole system. The promise theory gives special attention on how to design a system that is not controlled by force and is able to work voluntarily. The only control that exists within the agent is the constraint from the promise made by the agent to the promisee. Moreover, this theory does not require for a centralized control.

The Viplan method and the promise theory are the two main theories that provide the design concept in developing the method proposed in this paper. The proposed method has then been improved and tested in analyzing and designing the microservice system. Microservice is a new paradigm in designing service systems. Systems used to be designed in the context of the client server architecture [11]. Then, the concept of the Service Oriented Architechture (SOA) with the purpose of making a loosely-coupled service system was created [14], [18]. However, humans still wanted to control the system they have developed, and thus the type of SOA that was adopted was the Enterprise Service Bus which failed to be controlled by the humans and resulted in a non-scalable system [6]. Microservice on the other hand, is to ensure that the service system is scalable, regardless of the constraints of time and space [22]. Furthermore, microservice does not require for any direct human control or any centralized control. In order to achieve this, there is a need for a method that can be used to analyze and design the microservice holistically.

III. Related Work

This research is related to other ongoing researches regarding how to break the monolithic system and to identify the microservice boundary, and transforming monolithic system into microservice system [14]-[16], [21], [24]. Monolithic system is a system developed in one long script that have thousands line of codes. The code will be modified if there is a need and the changes made will affect the whole system when redeployed because the monolithic system is tightly coupled in nature. This style of system development usually implemented using programming language such as PHP and Ruby. Monolithic style of system is also not an exception to the implementation using object oriented programming language such as Java and .Net. Monolithic system is a centralized architectural style of information system. The problem will occur when the demand to the system by the user are beyond the threshold level of how the system can handle and the centralized style of system has the problem to scale and to load balance the system gracefully [9].

Other related works are on composing microservices to make the separated microservices to interact and work in cooperative manner. The composition of microservice is related to service orchestration [20] and service choreography [23] and piping [12]. The organization structure is another related research that play important role in developing microservice [11], [17].

IV. Research Methodology

The research methodology used in this research is the Design Science Research methodology. The methodology consists of five phases [14]:

A. *Awareness of the Problem*

This research is aware of the problems faced in analyzing and designing architecture-based microservice systems. The existing method is no longer suitable in analyzing and designing microservice-based architecture since microservice architecture is a decentralized and loosely-coupled type of architecture [19].

B. *Suggestion*

The use of a more holistic method of analyzing and designing microservice-based systems is suggested so that it is able to reap the full benefit of the architecture such as scalability, decentralization, loosely coupled and autonomous.

C. *Development*

Existing theories are surveyed and tested in order to be used as the foundation for the proposed method. Among the theories are variety engineering, the Viable System Model, the Viplan method, and the promise theory.

D. *Evaluation*

The method was evaluated using different case studies to improve the design.

E. *Conclusion*

The final design of the method was produced and presented in this paper as a way of communication. The final design of the method is considered as an artifact of the research, which is the contribution of this research to the body of knowledge.

V. Method for Analyzing and Designing Microservice Architecture Holistically

The proposed method consists of the following steps. All the steps are described using an example of a case study for analyzing and designing a microservice for a local university in Malaysia.

A. *Identifying the Organizational Identity and Objectives*

The organizational identity and objectives in this case are for a university. They can be formed using the TASCOI formula. The importance of this step is to ensure that all members of the organization understand what the organization identity is and where it is heading to. Every member must have the same understanding of what the transformation is that is being done by the organization.

1) *Transformations*

The transformations made by the university are:

- To equip students with the most current and quality knowledge to face the working world.
- To increase the research impact in the niche areas.

2) Agents (who will perform the transformations)

- Lecturers
- Researchers
- Administration staff
- Support staff

3) Suppliers (who will supply the input for the transformations)

- Secondary schools
- Polytechnics
- Industries
- Government sectors
- Other higher education institutions
- Other research institutions

4) Customers (who will benefit from the transformations)

- Funders
- Industries
- Government sectors
- The general public

5) Owner (who owns the organization)

- University Board of Directors, Vice Chancellor, Deputy Vice Chancellors

6) Intervener (who can intervene the organizational transformations)

- The Ministry of Higher Education
- Accreditation agencies
- Other higher education institutions

B. Modeling of the Organizational Complexity Drive

The purpose of this step is to model the existing transformation that is carried out by the organization based on five complexity drives which are the technology, geography, time and customer/supplier.

1) Technological drive

Fig. 1 is the example of technological model. The purpose of the technological drive is to see what transformation technology is adopted by the organization in converting the input into the output.

2) Geographical drive

The geographical drive looks at how the organization is spread over at different locations (Fig. 2)

3) Time drive

The time drive (Fig. 3) is important in assessing how time plays an important role in the organizational complexity.

4) Customer/supplier drive

The customer/supplier drive helps to identify the suppliers who supply the input into the transformation process, and to identify the customers who benefit from the organizational transformations (Fig. 4).

C. Modeling the Organizational Unfolding Structure

All the complexity drive modeled in the previous steps are then combined into one model that is called the organizational unfolding structure. The purpose of this model is to model how the organization is structured from all types of drives. The primary activities can then be identified from this unfolding structure and all the other unfolding structures under each primary activity. The unfolding structure is a recursive structure (Fig. 5).

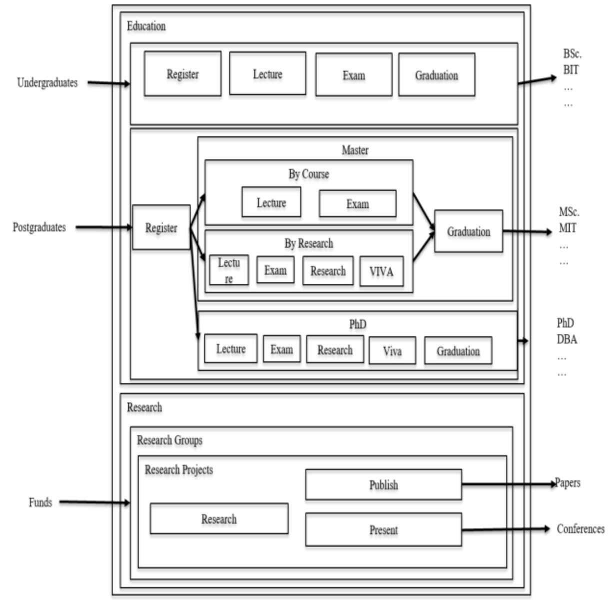

Fig. 1. Technological model.

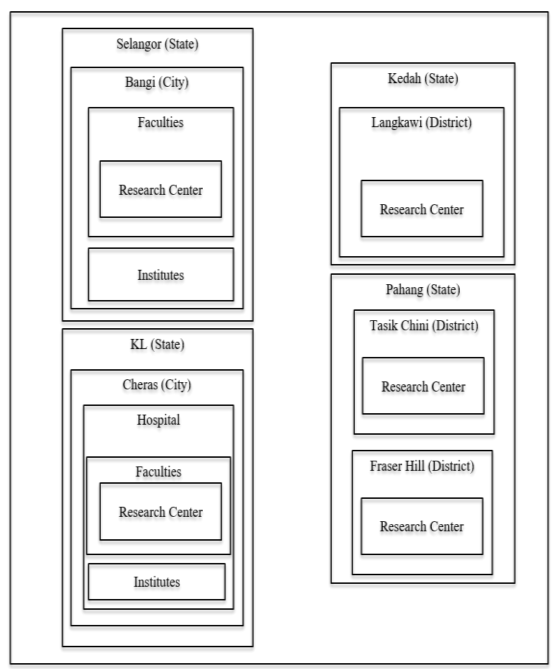

Fig. 2. Geographical model.

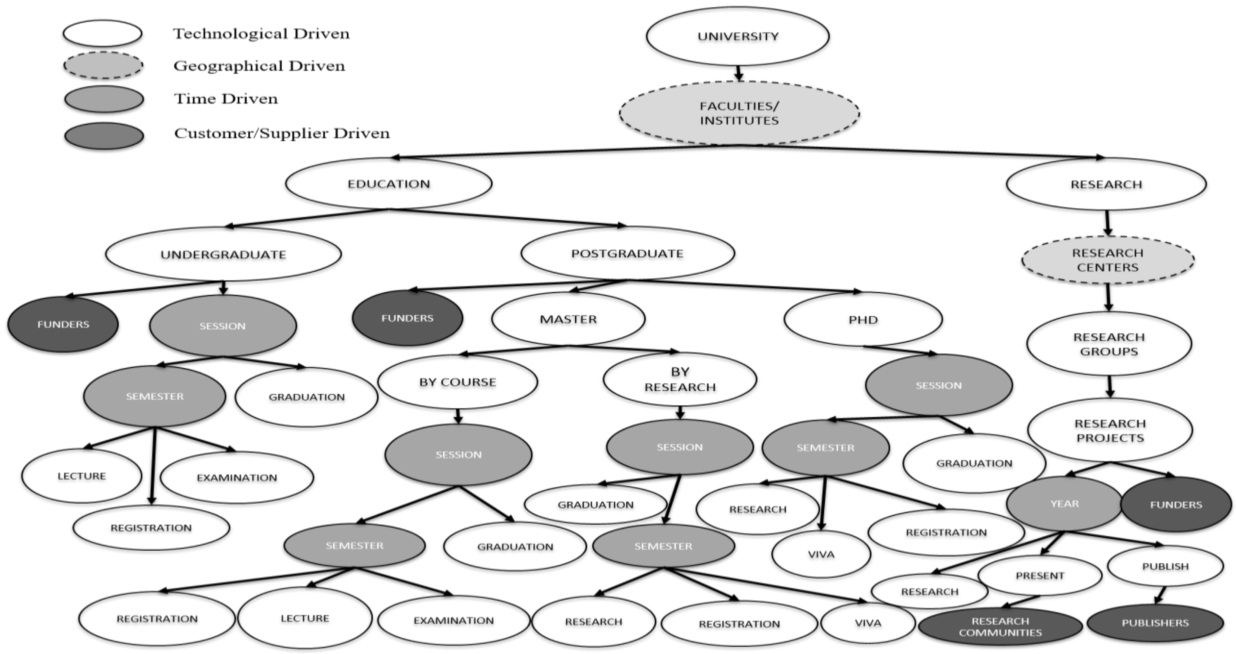

Fig. 3. Organizational unfolding structure model.

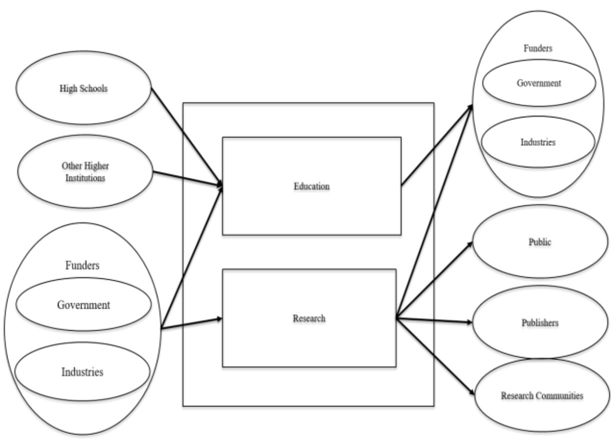

Fig. 4. Customer/supplier model.

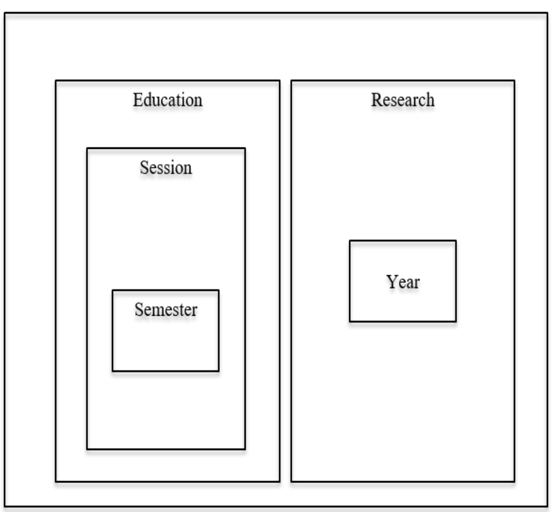

Fig. 5. Time model.

D. Building the Cross Table between the Primary Activities and the Support Activities

Primary activities are activities that are directly involved in the transformation process. Meanwhile, the support activities are performed by those other than the transformation activities but help to enable the transformation processes to take place. A cross table is used in mapping the primary activities to the relevant supporting activities (Table I). The primary activities are extracted from the unfolding structure identified in Step C. The primary activities in the table are focused on the unfolding structure under the title "Education".

TABLE I. CROSS TABLE BETWEEN THE PRIMARY ACTIVITIES AND THE SUPPORT ACTIVITIES

Primary Activities	**Support Activities**					
	Registrar	**Academic Department**	**Treasury**	**Computer Center**	**Security**	**Library**
University	X	X	X	X	X	X
-Faculties/ Institutes	X	X	X	X		
--Education	X	X	X			
---Undergraduates	X	X	X			
----Funders			X			
----Sessions	X	X	X			
-----Graduation	X	X	X			X
-----Semester	X	X	X			
------Lecture		X	X			
------Registration	X	X	X		X	X
------Examination		X	X		X	

E. Identifying the Business Processes

Business processes are all processes involved in the transformation process. The purpose of step one until step five is to reveal the structure of the organization (viable structure based on the Viable System Model). Existing or new business processes can thus be identified and then mapped to the corresponding primary activities. There are four main business processes identified which can be mapped under "Education" as the following:

- Students' registration
- Students' lecture
- Students' examination
- Students' graduation

F. Modeling the Business Processes using the Promise Theory

This is the step where the business processes are modeled in detail. The following are the detailed business process for "Students' Registration".

The "Students' Registration" business process involves three based events:

1) Pre-registration

- Fill in students' details
- Create Student IDs
- Assign faculties and programs
- Send offer letter to the students

2) Registration

- Students pay their registration fees
- Register students
- Create students' payment account
- Create students' ID cards
- Create students' library accounts

3) Course Registration

- Every faculty publishes the courses available for the semester
- Students choose courses which match their requirements and also the requirements of the faculty and the university
- Students register for courses
- Students drop courses
- Students pay course fees

Based on the above events, the business processes are then modeled using the promise theory

a) Pre-registration Promises (Fig. 6)

+D1: promise to input candidate information into batch files based on the faculties and programs into the system.

+D2: promise to process the batch files and to save the information into the database.

+D3: promise to create students' ID based on the saved information.

-D1->-D3: promise to accept/use the corresponding promises.

b) Registration Promises (Fig. 7)

+D1: promise to pay the registration fees to the university bank account.

+D2: promise to show proof of payment.

+D3: promise to activate students' status as active once provided with proof of payment.

+D4: promise to create students' payment account once students' status is activated.

+D5: promise to create students' ID card once students' status is activated.

+D6: promise to create students' library account once students' status is activated.

+D7: promise to update students' payment provided that the payment accounts have been created.

-D1->-D7: promise to accept/use the corresponding promises.

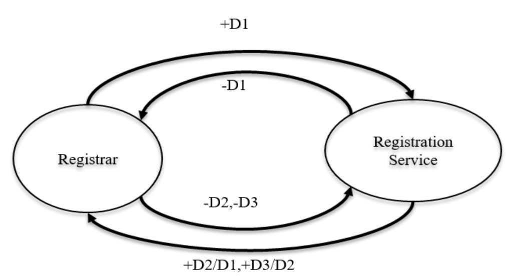

Fig. 6. Pre-registration promise model.

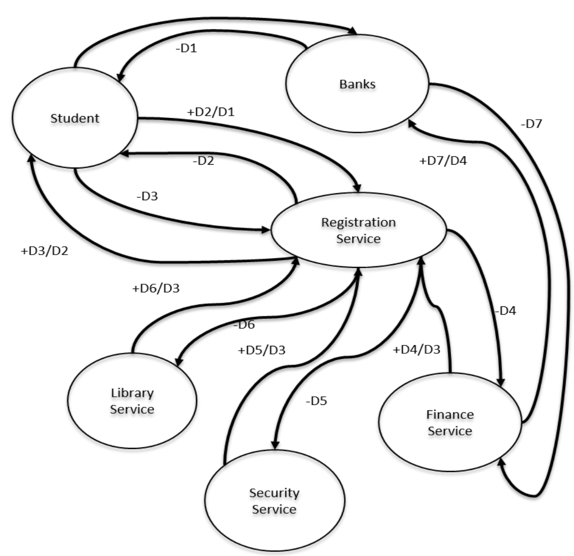

Fig. 7. Registration promise model.

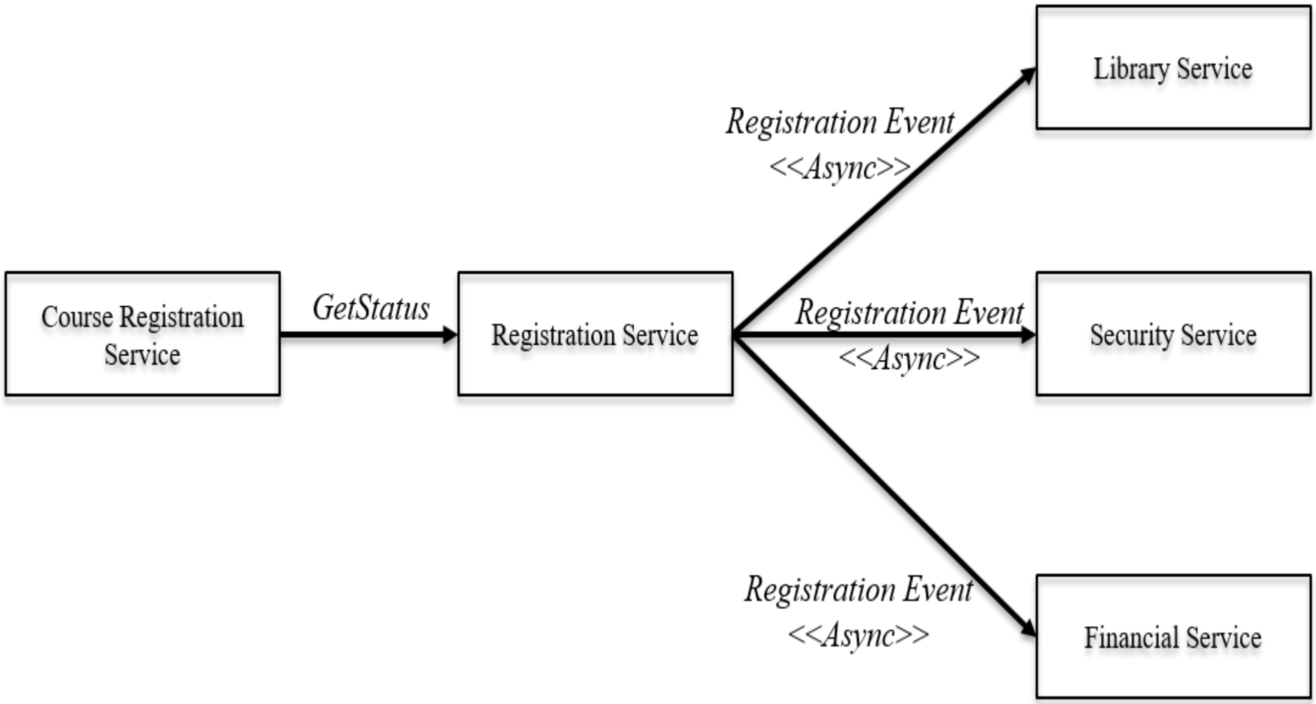

Fig. 8. Microservice dependency graph.

c) Course Registration Promises (Fig. 8)

+D1: promise to provide the registration status.

+D2: promise to publish courses offered for the semester.

+D3: promise to select courses offered which are relevant to graduation requirement.

+D4: promise to register for the selected courses.

+D5: promise to drop courses after the registration.

+D6: promise to establish the amount payable based on the registered courses.

+D7: promise to pay the course fees to the banks.

+D8: promise to update payment.

-D1->-D8: promise to accept/use the corresponding promises.

G. Identifying the Microservice Candidates

Based on step six above, five microservice candidates have been identified. The microservice will then be implemented in the designated server or container in the cloud. The microservice candidates are:

- Registration Service
- Course Registration Service
- Finance Service
- Library Service
- Security Service

H. Modeling the Microservice Dependency Graph

Fig. 9 shows dependency between the microservices which are modeled in this step. At this stage the system designer determines how to set the dependency between one service to another. The interaction is either asynchronous or synchronous. The detailed implementation of each service can be referred back to the modeling of the business processes with the promise theory in step six.

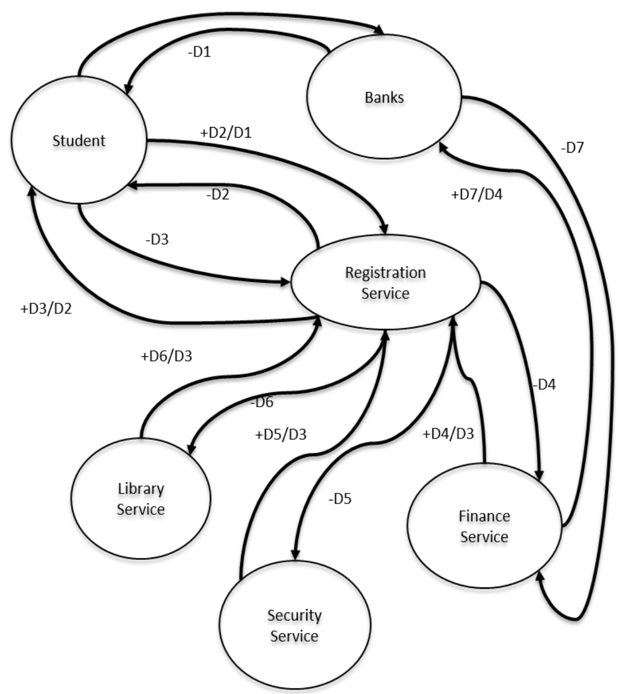

Fig. 9. Courses registration model.

VI. Future Work

There are other improvements that can be made to the existing work and the possible improvements are:

- To improve the existing proposed method and to find other use cases that beneficial in designing better microservice.
- To create software tools that can automate the process of designing and creating microservices.
- To design and model microservice infrastructure such as api gateway, load balancing, monitoring, logging, configuration and microservice optimization.
- To design a microservice framework specific to implementation such as .Net language.
- To do research on microservice simulation to study the behavior of designed microservices.

VII. Conclusion

The method proposed in this paper has demonstrated its capacity to be used in analyzing and designing the microservice holistically. Without this method, the development of microservice-based systems would still be using the traditional method of system analysis and design. The proposed method has contributed to the development of service systems that are more loosely coupled, decentralized, scalable and autonomous. It has also been designed to take into account the latest technology such as the container technology and cloud computing. Future research is to improve the method designed in this paper by using other case studies in different domains.

References

[1] Ashby, W. R. 1956. An introduction to cybernetics. University paperbacks. J. Wiley.

[2] Ashby, W. R. 1958. Requisite variety and its implications for the control of complex systems. Cybernetica.

[3] Ashby, W. R. 1962. Principles of self-organizations: Transaction. Pergamon Press.

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

[5] Bergstra, J. A. & Burgess, M. 2014. Promise theory: principles and applications. Promise Theory. Createspace Independent Pub.

[6] Bhadoria, R. S., Chaudhari, N. S. & 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.

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

[8] Burgess, M. 2015b. Thinking in promises: designing systems for cooperation. O'Reilly Media.

[9] Dragoni, N., Lanese, I., Larsen, S. T., Mazzara, M., Mustafin, R. & Safina, L. 2017. Microservices: how to make your application scale.

[10] Espejo, R. & Reyes, A. 2011. Organizational systems: managing complexity with the viable system model. Springer Science & Business Media.

[11] Gucer, V., Narain, S. & others. 2015. Creating applications in bluemix using the microservices approach. IBM Redbooks.

[12] Gutierrez, F. 2016. Spring Boot in the cloud. Pro Spring Boot. Springer.

[13] Hanson, M. D. 2000. The client/server architecture. Server Management.

[14] Newman, S. 2015. Building microservices. O'Reilly.

[15] Holmes, B. & Nicolaescu, A. 2017. Continuous architecting: just another buzzword? Full-scale software engineering/the art of software testing.

[16] Johanson, A., Flögel, S., Dullo, C. & Hasselbring, W. 2016. OceanTEA: exploring ocean-derived climate data using microservices.

[17] Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J. & Josuttis, N. 2017. Microservices in practice, part 1: reality check and service design. IEEE Software.

[18] Rotem-Gal-Oz, A. 2012. SOA patterns. Running Series. Manning.

[19] Rusek, M., Dwornicki, G. & Orłowski, A. 2016. A Decentralized System for Load Balancing of Containerized Microservices in the Cloud. International Conference on Systems Science.

[20] Toffetti, G., Brunner, S., Blöchlinger, M., Dudouet, F. & Edmonds, A. 2015. An architecture for self-managing microservices. Proceedings of the 1st International Workshop on Automated Incident Management in Cloud.

[21] Vernon, V. 2013. Implementing domain-driven design. Pearson education.

[22] Von Alan, R. H., March, S. T., Park, J. & Ram, S. 2004. Design science in information systems research. MIS quarterly.

[23] Yahia, E. B. H., Réveillère, L., Bromberg, Y.-D., Chevalier, R. & Cadot, A. 2016. Medley: an event-driven lightweight platform for service composition. International Conference on Web Engineering, hlm. 3–20.

[24] Zimmermann, O. 2015. Do microservices pass the same old architecture test ? or : SOA is not dead – long live (micro-)services.