International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter30872020.pdf
https://doi.org/10.30534/ijeter/2020/30872020

# Design and Implementation of Microservices System Based on Domain-Driven Design

**Ahmad Nurul Fajar[1], Eva Novianti [2], Firmansyah[3]**
[1]Information Systems Management Department, Binus Graduate Program – Master of Information Systems Management, Bina Nusantara university, Jakarta, Indonesia 11480. afajar@binus.edu
[2]Information Systems Department,Faculty of Engineering DarmaPersada University, RadenInten II Street,Jakarta, Indonesia. eva_novianti@ft.unsada.ac.id
[3] PT. EbizCiptaSolusi Indonesia, Jakarta, Indonesia. tomy.firmansyah@hotmail.com

## ABSTRACT

PT. XYZ as an IT company has developed a basic product system for e-procurement applications to meet the needs of automation in the internal procurement process of a company. The e-procurement application system developed by PT. XYZ is designed using a monolithic architecture where the application is wrapped in a large package and dependencies between modules. The modules are an integrated whole process. If there is a change or enhancement to one particular module that is made to adjust the business process, then this will have an impact on the functionality of the other modules. Therefore it is necessary to consider other architectural alternatives so that in the future the e-procurement application system is more adaptive to change. So, it can face problems that arise with old architecture. We proposed the design the transformation process using microservices architecture. We use Domain-Driven Design (DDD) approach. The results showed that the microservices architecture design can produce a system that is more adaptive to a change in which the module is designed to stand alone.

**Key words** : e-procurement, microservices architecture, monolithic architecture, domain-driven design.

.

## 1. INTRODUCTION

In this digital transformation, companies are forced to develop fast and new system in order to meet their goals. Most of companies try to avoid the intricacy and high development costs of migration process from old system to new environment. Moreover, companies need to overcome several challenges on migration process [1]: (1) The cost of development, customers want to significantly reduce the cost of technology and refuse to fund technological changes that less profitable. (2) The pace of change, quarterly or even monthly release cycles no longer meet user needs. Customers expect launches more often, even several times a day, to offset their information and service requirements. (3) The quality of change, decision makers want to improve the scalability of applications and provide a rich, interactive and dynamic user experience on various devices, while minimizing the risk of change.

According to the challenges above, migration process will be difficult, time consuming, and complex procedures. sub-units, higher risk and need testing effort, require testing procedures and probably whole application should be tested. By comparison a monolithic architecture where an application is built as a single unit, the components are therefore not independently executable, and problem with scalability, IT companies must switch their systems towards an architecture that enables the expansion and high scale computing resources with higher level of coordination of services[2].

The e-procurement system application that was built with monolithic architecture where applications are encapsulated in one large package and dependencies between modules. These modules are a unified interconnected process. If there is a change or enhancement in one particular module that is made to adjust to business processes, then this will affect the functionality of other modules.

Microservice are a small application that can be deployed, scaled and tested independently and that has a single responsibility [3]. The Microservices architecture is an architectural alternative in building an application. The Microservices architecture migration process can follow a pattern adapted to future needs. A very important migration process is to implement automation for continuous delivery and continuous integration into continuous development, transforming data services into service forms, implementing Edge servers, introducing Service Discovery and Resource Manager and Clusterization. [4]

Based on the problems above, companies need motivation for transformation from the fact of constantly maintaining a monolithic architecture that show in difficulties in keeping up in pace with new development approaches such as DevOps, hundreds of services using different interfaces, and the modularization rely on the sharing resources of the same machine. This organization need to move the complexity to the level of coordination with Microservices architecture. Using microservices architecture will help company to integrate existing business processes, support information technology infrastructure and also service components that can be reused and combined in accordance with business priorities. This can facilitate the process of advanced development, testing, repair and deployment. With this approach, the application of the e-procurement system will consist of several services that can be managed and distributed independently, this will make it easier for the system to adapt to changing needs.

## 2. RELATED WORKS

According to [5], Electronic procurement (e-procurement) is the process of procurement of goods or auctions by utilizing information technology in the form of a website. Meanwhile, according to [6], e-procurement is an electronic integration in the management of all procurement activities including purchasing requests, authorization, ordering, shipping and payment between buyers and suppliers. In other words, the process of procuring goods and services using the e-procurement system utilizes information and communication technology facilities that are used to support the public auction process electronically.In a broad sense, e-procurement involves electronic data transfer to support operational, tactical and strategic procurement [7]. According to[8], microservices architecture is servcie oriented architecture done right. According to [9], explain about Evolution of Service Orientation. Microservices architecture (MSA) is an approach in developing an application that consists of small services. [10]. The explaination monolithic. Microservices Approach versus Traditional Application has been done by [11].According to [12] concluded under the MSA architecture is the latest architectural trend that has significantly gained popularity over the past few years. The approach in modeling MSA is by Domain-driven Design (DDD) [13]. According to [14], DDD is an approach to software development that focuses on the application domain, its concepts, and its relationship to the main drivers for architectural design in determining their patterns. According to [15], the tactical phase can help in designing MSA that are loosely coupled and cohesive .

## 3. RESULTS AND DISCUSSION

A. E-Procurement Business Process

The following is an overview of the overall module flow of the e-procurement system of PT. XYZ:

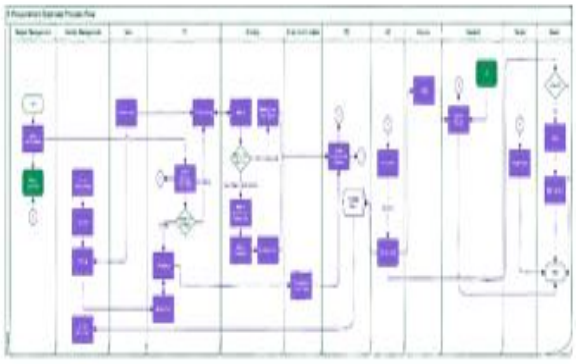

**Figure 1:** Flow E-procurement

1. Admin will make application settings such as company profile, master data, user data, groups and their access, in App Settings, Maintenance Parameters, and General.
2. Maker can choose the type of period to be used with an annual budget. Whether it's monthly, quarterly, semester, or not at all.
3. Maker registers the items to be spent on the Master Item.
4. Maker will register the vendor in the Vendor Profile. Once registered, the Maker can upload contract documents relating to the vendor in Contract Management, and upload the vendor catalog to the Master Catalog. The vendor catalog collection will be displayed on the Marketplace.
5. To make a purchase request the user concerned will fill out the Purchase Requisition (PR) form. If the item you want to buy is on the Marketplace, the PR is a PR Catalog. If the item you want to buy is not in the Marketplace, the PR is a PR Catalog. After the PR is approved, the system carries out a budget calculation, then data information is seen in the Budget Expense
6. For the PR Catalog, the Maker needs to confirm the number of items that can be fulfilled by the vendor in the Order Confirmation Catalog before proceeding to the Purchase Order.
7. For Non-Catalog PR, vendor selection must be continued through bidding. Maker will make Bidding Announcement based on agreed homework. Bidding Evaluation will be conducted, then Maker will determine the choice of the winning vendor (can be manually or automatically by the system based on the results of scoring in the bidding evaluation).
8. After there is a winning vendor, Maker will create a Purchase Order form based on the agreed price.
9. If the goods from the vendor have been received and are in good condition then the Maker will make a record on the Goods Receipt form. If the goods want to be returned due to certain conditions, it can be noted on the Goods Return form.
10. When an invoice is received from the vendor, Maker will fill the Payment Request form. After approval, this form can be printed as a payment request to the financial authorities.
11. The Asset Goods received will go to the Asset Master, which can then be printed QR Code so that it can be registered and carried out a Stock Hospitalization.

B. Domain Analysis

Using the DDD approach can help in designing microservices so that each service will later suit the functional needs of the business. Before writing any code, it requires a glimpse of the system to be developed. DDD starts with modeling the business domain and creating a domain model. The domain model is an abstract model of the business domain. This process is to filter and organize domain knowledge, and provide a common language for developers.

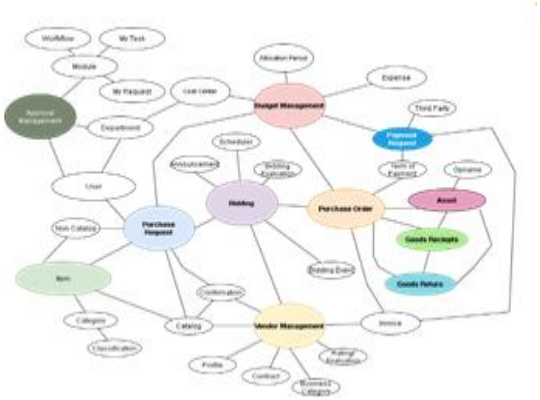

**Figure 1:** Sketch of Analysis of E-Procurement System D*omain*

From the results of the analysis of the running system, it is obtained an overview of the domain of the business process procurement area as shown below:

The domain area of the e-procurement system can be partitioned into subdomains in order to manage complexity and separate important parts from other systems. Subdomains represent areas of capabilities, define business processes, and represent system functionality. From thedomain analysis sketch above, it can be identified that the domain can be divided into several subdomains, including the following

**Table 1.:**E-procurement system sub-domain

| No | Subdomain | Domain Type |
|---|---|---|
| 1 | Budget | Core Domain |
| 2 | Purchase Request | Core Domain |
| 3 | Purchase Order | Core Domain |
| 4 | Bidding | Core Domain |
| 5 | Item | Core Domain |
| 6 | Catalog | Supporting Domain |
| 7 | Vendor management | Core Domain |
| 8 | Goods Receipts | Supporting Domain |
| 9 | Goods Return | Supporting Domain |
| 10 | Asset | Supporting Domain |
| 11 | Approval Management | Generic Domain |

In the table above, subdomains are divided into core, supporting and generic. Subdomain **Budget**, **Purchase Request**, **Purchase Order**, **Bidding** and **Vendor Management** are grouped into domain types, namely the core domain that is the core of the e-procurement system domain. Subdomain **Catalog**, **Goods Receipts**, **Goods Return** and **Assets** are grouped into supporting domains into supporting subdomains to help core domains. Whereas **Approval Management** enters the generic domain because it supports all subdomains in the e-procurement area domain.

C. Define bounded context

From the results of the domain analysis process, functions can be grouped based on whether various functions will share a single domain model or not, therefore it is necessary to analyze and identify the context domain constraints.Context limits are a central pattern in DDD. The focus at this stage is to focus on the strategic design part of DDD, all of which relate to models and large teams. DDD deals with large models by dividing them into different bound contexts and explicitly about their interrelationships. Bounded context are the defining process for applying the model. This process will provide clarity about what model is used, where it must be consistent, and what should be ignored. Contextual boundaries ensure that domain concepts outside the context of the model do not distract from problems designed to be solved. Context is an important term in DDD. Each model has a context that is implicitly defined in a subdomain. The context defines the scope of the model, limits the problem boundaries, allows the team to focus without distraction.The following figure will explain the boundaries of each existing subdomain, which later will become a reference in domain modeling and determine the services to be built:

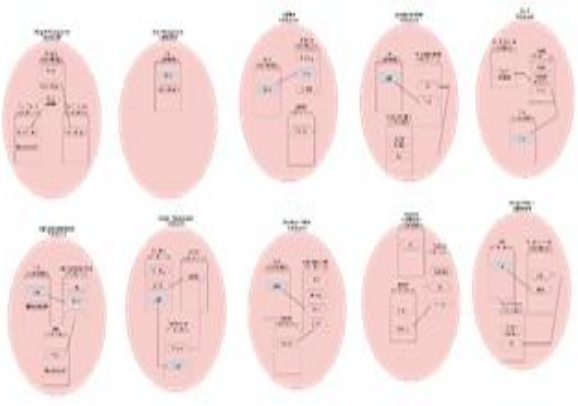

**Figure 2:** Bounded Context Diagram E-Procurement

Service identification is classified based on a series of business processes, business functions, and business function development management and microservice identification results based on the results of modeling analysis with the DDD approach. Following are the service candidates that will be developed on the microservice architecture design on the application system:

1. Item Service
   This microservice provides several functions related to managing data items such as adding, changing and withdrawing data.
2. Vendor Service
   This microservice provides several functions related to Vendor data management such as adding, changing and withdrawing data.
3. Asset Service
   This microservice provides several functions related to Asset data management such as adding, changing and withdrawing data.
4. BaseService
   This microservice provides several functions related to managing applications such as setting themes and configurations that are related to the application and controlling the authentication and authorization processes.
5. PurchaseRequisitionService
   This microservice provides functions related to the functionality of the purchase requisition process.
6. PurchaseOrderService
   This microservice provides functions related to the functionality of the purchase order process.
7. BiddingService
   This microservice provides functions related to the functionality of the bidding process.
8. CatalogService
   This microservice provides several functions related to catalog data management such as adding, changing and withdrawing data.
9. ContractService

This microservice provides several functions related to managing data contracts such as adding, changing and withdrawing data.

10. GoodsReceivedService
    This microservice provides functions related to the functionality of the Goods Received process.
11. GoodsReturnService
    This microservice provides functionality in the processing of Goods Return data.
12. BudgetService
    This microservice provides functionality in budget data processing.
13. WorkflowService
    This microservice provides several functions related to management task approval such as Start Workflow, Task Approval Action etc.

D. Architectural Design of Microservice-Based E-Procurement Application Systems. The transformation strategy undertaken in this case study is, by breaking down the monolithic system by extracting functions into services which will be implemented using the ASP.Net Core framework because it is expected that this system will be able to run across platforms such as Windows or Linux without changing in terms of functionality and business existing requirements. On the frontend or UI side, it will still use the existing project code solution. Based on the results of the model analysis and identification of the existing microservice, the following is the architecture design of the microservice-based e-procurement application system: We proposed architecture design of microservices in figure 4 below :

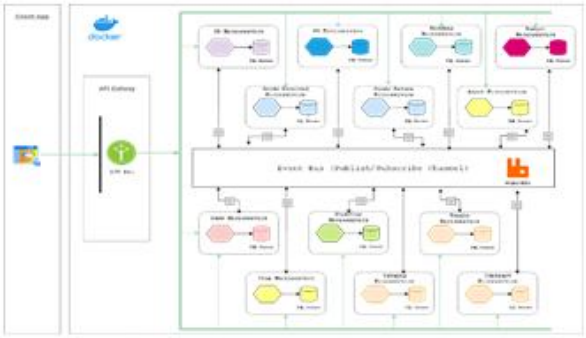

**Figure 4:** Architectural Design of Microservice-Based E-Procurement Application Systems

With the proposed microservice-based architecture design in the picture above, a Gateway API component or also called Backend for Frontend (BFF) is added, the purpose of which is to mediate between web clients communicating directly with Microservice and serving as management API or router. Each service from microservice has its own data storage approach using a pattern database per service, this helps ensure that each microservice is loosely coupled so that changes to the data base in one service do not affect other services. With a database per service approach, the design of data sharing mechanism is done through message brokers or bus events. The message broker is in charge of managing messages sent by the message sender (publisher) so that it can reach the recipient (subscriber). Each Microservice in the above architectural design is packaged in the form of an image so that it can be run or deployed in a container technology-based environment. Each service is developed by applying the command query responsibility segregation design pattern (CQRS) which separates the command to retrieve data (query) by changing the data (command).According to scenario that has been proposed by [10]. Related to ti, From the scenario in the picture above, it starts from the request to the PurchaseRequisitionService service with the POST method from the client, in this case the API Gateway is entered through the API controller with JSON format.

After the POST data has been successfully carried out, the PR service publishes the PR data via the bus event or publisher event to process the approval submission that will be processed by the Worfklow service through the event subscriber mechanism. The command handler to process the approval submission to the checker then handles the message received by the Workflow service.

After the checker performs the approve task, the Workflow service will publish the approval status through the bus event or subscriber event, which will be captured by the PR service subscriber event and the Budget service through the command handler of each service. In the PR service handler command will pass the status update PR data in accordance with the status of the action approval carried out by the checker whether rejected or approved put for the Budget service on the command handler will calculate the budget if the action approval status is approved.

## 4. CONCLUSION

Analysis and design of microservice architecture through a domain-driven design approach can help in the process of identifying microservice. Applying microservice architecture to the e-procurement application system is appropriate because the interrelationships between the modules within it can be broken down into a service or a stand-alone system that will facilitate the maintenance process and application development.

## REFERENCES

[1] Cognizant, “Overcoming Ongoing Digital Transformational Challenges with a Microservices Architecture,” Cogniz. 20-20 Insights, no. november, pp. 1–9, 2015.

[2] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen and M. Mazzara, "From Monolithic to Microservices: An Experience Report from the Banking Domain," in *IEEE Software*, vol. 35, no. 3, pp. 50-55, May/June 2018, doi: 10.1109/MS.2018.2141026.

[3] P. Clarke, P. Elger and R. V. O'Connor. "Technology enabled continuous software development," Proceedings of the International Workshop on Continuous Software Evolution and Delivery, pp. 48-48, 2016. https://doi.org/10.1145/2896941.2896943

[4] W. P. Lestari and A. Sujarwo, "DevOps : DisrupsiPengelolaan ICT PendidikanTinggi," Semin. Nas. Apl. Teknol. Inf. 2018, pp. 26–31, 2018.
[5] R. Kalakota and M. Robinson, e-Business 2.0. 2000
[6] D. Chaffey, E-Business and E-Commerce Management : Strategy, Implementation and practice, Fourth. 2009.
[7] Ecommercewiki, "Procurement Process," 2019. [Online]. Available: http://en.ecommercewiki.info/logistics/procurement_process. [Accessed: 01-Apr-2019].
[8] C. de la Torre, B. Wagne, and M. Rousos, NET Microservices Architecture for Containerized NET Applications. Washington: Microsoft Corporation, 2019
[9] G. Gruman and A. Morrison, "Microservices : The resurgence of SOA principles and an alternative to the monolith," PwC Technol. Forecast, no. 1, pp. 1–8, 2014
[10] Richardson, Chris (November 2018). Microservice Patterns. Chapter 1, section 1.4.1 Scale cube and microservices: Manning Publications. ISBN 9781617294549
[11] S. Balakrushnan et al., "Microservices Architecture – CASE STUDY: Rainyday Grocer," The Open Group, 2016. [Online]. Available: http://www.opengroup.org/soa/source-book/msawp/p9.htm. [Accessed: 01-Apr-2019]
[12] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," J. Comput. Inf. Syst., vol. 00, no. 00, pp. 1–9, 2018
[13] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley, 2003
[14] R. H. Steinegger, P. Giessler, B. Hippchen, and S. Abeck, "Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications," SOFTENG 2017 Third Int. Conf. Adv. Trends Softw. Eng., no. April, pp. 79–87, 2017
[15] M. Wasson, "Build microservices on Azure," Microsoft, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/microservices/model/domain-analysis. [Accessed: 01-Apr-2019]