Transforming the Monolith E-Procurement Application

Tomy Firmansyah, Abiyyu Ganeswangga, Ahmad Nurul Fajar

Abstract: ***Digital transformation in IT organizations needs to overcome a number of key challenges: (1) Costs of change, the customer wants to significantly reduce the cost of technology, and on the other hand the customer also does not want to fund technological changes that do not produce benefits for him directly. (2) Rate of change, customers request more frequent release cycles to compensate for changes in information and service requirements. (3) Quality of change, Increased application scalability. For that reason, IT organizations in making business changes need to do what needs to be modular in developing applications and deployments. Architectural Microservices (MSA) is an alternative architecture in building an application. The MSA conversion process can start patterns that are adapted to future needs. The release cycle is carried out by implementing automation for continuous delivery and continuous integration into continuous development, transforming data services into service forms, so as to answer the challenges of digital transformation. In this paper explains how to transform architects and their design in the E-Procurement application system using MSA.***

Keywords : Microservices, Architecture, SOA, Cloud Computing, and E-Procurement..

I. INTRODUCTION

The increasingly rapid digitalization of business forces many organizations to rethink their operating models to meet the ever-increasing expectations of technology-savvy customers. At a fundamental level, companies in all industries need to ensure the services provided are available through digital channels and competitive with technological innovation. To enable this digital transformation [1], IT organizations need to overcome a number of key challenges: (1) The cost of change, the customer wants to significantly reduce the cost of technology, and on the other hand the customer also does not want to fund technological changes that do not produce benefits for him directly. (2) The pace of change, quarterly or even monthly release cycles no longer meet user needs. Customers expect more frequent launches, even several times a day, to compensate for their information and service requirements. (3) The quality of change, decision makers want to improve application scalability and provide a rich, interactive and dynamic user experience on a variety of devices, while minimizing the risk of change.

This cannot be done with a monolithic architectural approach or the de facto standard traditional multi-tier web application for building applications in the current era of Internet applications, because this is too difficult to develop, test and maintain. Multi-tier web applications typically consist of complex code bases, as a result of enormous functionality built into a single Web application. In addition, one change can affect many sub-units, so the testing is much larger and more complex, requiring testers to understand various code interdependencies or test the entire application for each change. Finally, because changing even one aspect of this monolithic application affects the entire code base, it slows the testing process and makes it error-prone. To meet the growing multi-dimensional challenges, IT companies must move from monolithic web applications that serve HTML to desktop browsers, to application architectures that enable architectural expansion.

The e-procurement system application is built using a monolithic architecture style, where the application is wrapped in a large package and dependencies between modules. The modules are a unified interconnected process. If there is a change or enhancement to one particular module that is made to adjust the business process, then this will affect the functionality of the other modules. Therefore it is necessary to consider other architectural alternatives so that the future application of e-procurement systems is more adaptive to change.

Based on the above problems, the need for analysis and design of e-procurement system applications in the transformation of application architecture to MSA-based architecture. Because this approach is a framework that can integrate existing business processes, support information technology infrastructure and also service components that can be reused and combined according to business priorities. This can facilitate the process of continued development, testing, repair and deployment.

II. LITERATURE STUDIES

A. E-Procurement

Electronic procurement (e-procurement) is the process of procurement of goods or auctions by utilizing information technology in the form of a website [5]. 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

Revised Manuscript Received on November 12, 2019.

***** Correspondence Author

Tomy Firmansyah, Information Systems Management Department, BINUS Graduate Program-Master of Information Systems Management, Bina Nusantara University Jakarta, Indonesia 11480, Jakarta, Indonesia.

Abiyyu Ganeswangga, Information Systems Management Department, BINUS Graduate Program-Master of Information Systems Management, Bina Nusantara University Jakarta, Indonesia 11480, Jakarta, Indonesia.

Ahmad Nurul Fajar, Information Systems Management Department, BINUS Graduate Program-Master of Information Systems Management, Bina Nusantara University Jakarta, Indonesia 11480, Jakarta, Indonesia.

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 [4]. Because of that e-procurement has existed far longer than the term itself which was first used after the formation of the internet in the 1990s. From the 1960s to the mid-1990s, electronic procurement mainly took the form of electronic data interchange (EDI). Nowadays, electronic procurement is often supported by internet technology and is becoming more common. The historical context is shown in the chart below :

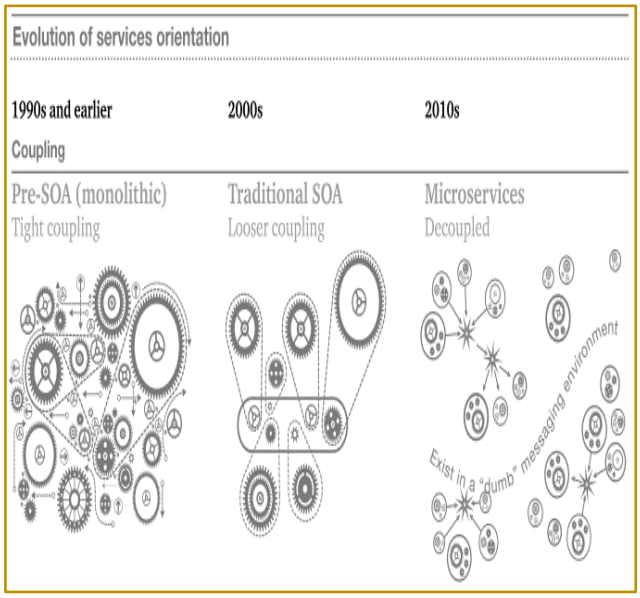


Fig. 1. The historical context of developing an E-Procurement system [22]

B. Service-Oriented Architecture

Service-Oriented Architecture (SOA) is known as Service Orientation which automates business logic as a distributed system. SOA is an architectural style for building enterprise service based solutions. SOA combines business processes and business services in organizations to provide existing systems such as database programs or applications as services. This service allows users and processes to access this functionality, and at the same time allows the exchange of data in individual processes. SOA and MSA have one thing in common: none has a clear definition. This section only looks at one possible definition. Some definitions suggest that SOA and MSA are identical. In the end, both approaches are based on service and application distribution into services. The term "service" is central to SOA. [9]. SOA is a term that is too often used and has different meanings for different people. But as a common denominator, SOA means compiling applications by decomposing them into several services (most commonly as HTTP services) that can be classified as various types such as subsystems or levels. MSA are from SOA, but SOA is different from MSA. Features such as large central brokers, central orchestrators at the organizational level, and Enterprise Service Bus (ESB) are typical in SOA. But in most cases, this is anti-patterns in the MSA community. In fact, some people argue that "The MSA is SOA done right." [10].

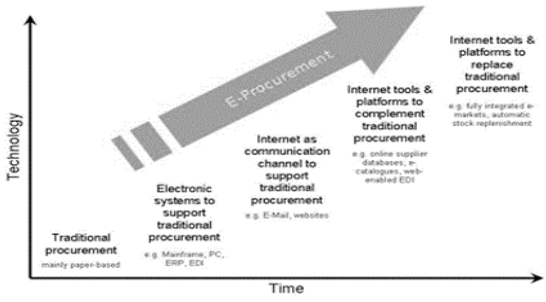


Fig: 2 Evolution of Service Orientation[13]

B. Microservices Architecture

Microservices architecture (MSA) is an approach in developing an application that consists of small services. Each service runs in its own process and communicates with a lightweight mechanism using the HTTP Resource API. These small services have minimum standards in operating several processes that can be developed with different programming languages and data storage technologies [2].

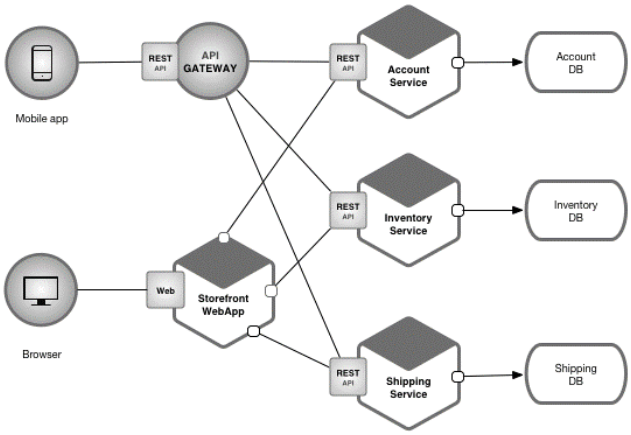
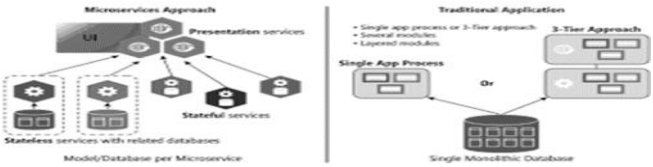


Fig. 3. Microservice Architecture [23]

The concept of MSA is more or less different from the paradigm of its predecessor, namely Service Oriented Architecture (SOA). Starting from the use of message communication protocols that are more concise (REST, etc.) compared to SOA that uses XML SOAP a lot, to the design and functional division processes that prioritize the division based on domain functionality in the organization.Here is an image to explain monolithic architecture with MSA:

Fig. 4 Microservices Approach versus Traditional Application [10]



According to [12] concluded under the MSA architecture is the latest architectural trend which has significantly gained popularity over the past few years. Although there is a lot of hype associated with MSA, and while many vendors and enthusiasts promote various architectural benefits that are said to be related to MSA, with the exception of a small number of success stories that are often

mentioned (for example, Amazon, Netflix, and Guardian). Some organizations seem to have refactored or developed from scratch large organizational systems or differentiation systems using MSA architecture. From this study of 19 experienced ICT architects, found that there is a lot of professional uncertainty associated with the adoption of MSA architecture. Although most of the interviewees understood the benefits recognized from the MSA namely increased development / deployment agility and operational scalability, they also had significant objections related to various technical and organizational aspects of the adoption of MSA.

C. Analysis Model Microservices

One of the biggest challenges in implementing MSA is how to determine the limits of each service. The general rule is that a service must do a specific thing and be designed according to business capabilities. The approach in modeling MSA is by Domain-driven Design (DDD) [16].

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 [17]. The DDD core principles consist of (1) capturing relevant domain knowledge in a domain model that may consist of structural and behavioral aspects; (2) collaborative modeling of domain experts and software engineers; (3) developing an experimental design by aligning the model and implementation closely throughout the software development process and perfecting the ongoing model; (4) encourage communication between domain experts and software engineers by jointly defining ubiquitous languages explicitly, consisting of domain-specific terms that are relevant and used in both, domain models and implementation.

DDD provides a framework for getting the most out of a well-designed set of MSAs. DDD has two different phases namely strategic and tactical. The process of defining a large-scale system structure is part of the Strategic DDD phase. This phase helps ensure that architecture stays focused on business capabilities. Then in terms of providing a series of design patterns that can be used to create a domain model is part of the Tactical DDD phase. Among them are entities, aggregates, and domain services in determining their patterns. so the tactical phase can help in designing MSA that are loosely coupled and cohesive [16].

III. METHOD

The method used in this article to identify and classify domain models in the E-Procurement application is done with the Domain-Driven Design approach [17] namely: (1) starting with analyzing the business domain by understanding the functional requirements of the application. The informal description of the domain is output at this step, which can be refined to a more formal set of domain models. (2) Determination of domain bounded contexts. Each interconnected context contains a domain model that represents a particular subdomain of a larger application. (3) Implement Tactical DDD patterns to define entities, aggregates, and services domains. (4) Use the results from the previous step to identify MSA.

IV. RELATED WORKS

S. Balakrushnan et al. the case study in applying MSA revealed that it was compatible with the needs for reasons including: (i) making it possible to implement complex processes with very lean teams, (ii) being able to build teams quickly with a series of skills available to build each MSA, (iii) In this case can make changes to the conditions on-the-go; eg, CI and CD and DevOps empowerment, (iv) agility and flexibility, (v) Optimal CapEx and OpEx [19].

J. Fritzsch in his research shows that packet-level metrics can provide a means to evaluate potential refactoring to MSA in advance. The strength of this method lies in its rapid application on a large code base. However, there are also aspects that affect the validity of the approach: (1) Dependency between packages may not correlate 1: 1 with the interface that must be defined between services. This is caused by following classes in object-oriented design as atomic units. In conclusion, the transferability of cohesion and coupling metrics from the packet level to the micro-service is limited and needs further investigation. (2) Experiments use only one sample application, which limits the validity of the findings. In addition, the three decompositions investigated have not been thoroughly verified by architects to serve as a measure for evaluating package-level metrics.

V. DOMAIN ANALYSIS

Domain analysis is an organization that administers, monitors or controls a problem domain. The purpose of the application domain is to analyze the needs of system users.

Applying the DDD approach can help in designing MSA so that each service is compatible with functional business needs.

DDD started by modeling the business domain and create a domain model. The domain model is an abstract model of the business domain. It filters and manages domain knowledge, and provides common language for domain developers and experts [16].

Creating a comprehensive domain model in this phase requires experienced domain modelers to gain knowledge. Class syntax diagrammed in the Unified Modeling Language (UML) modeling is used to describe concepts and their relationships, constraints, etc. [18].

The result of domain analysis is the domain model, which contains relevant domain concepts, also called knowledge domains [17]. DDD emphasizes this as a focus on the core domain that is relevant for downstream application implementation [17].

The principle of MSA is that services communicate through a well-defined API. This approach is consistent with the two patterns that Evans calls the Open Host Service and Published Language. The idea of the Open Host Service is that a subsystem defines a formal protocol for other subsystems communicating with it. Published Language extends this idea by publishing APIs in a form that other teams can use to write clients. In designing an API for micro services, it uses the OpenAPI specification to define the language agnostic interface description for the REST API, which is expressed in the JSON or YAML format [5].

VI. DESIGN E-PROCUREMENT SYSTEM

Based on research [6], functional identification and technical requirements of the E-Procurement system in the procurement process in general:

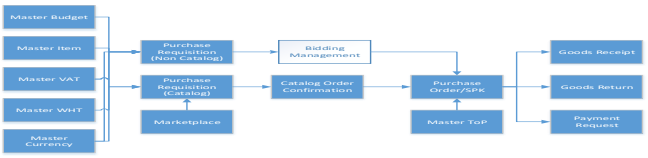
A. Data Entities

Table- I: Functional identification and technical requirements of

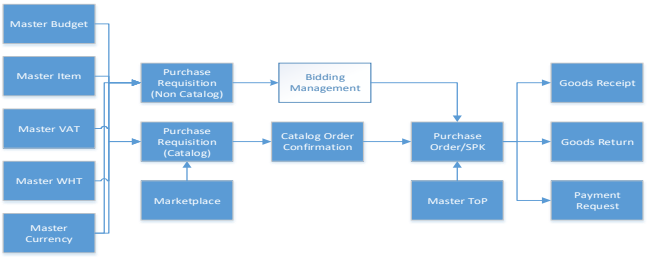
Entity	Owner	Original source	Format	Used in function
Purchased item master	Buyer	Multiple IT systems, such as one or moreinstances of ERP, supply-chain planners	Primarily electronic in various formats	. Determination of supply requirements . Determination of vendor and available supply . RFQ preparation
Supplier catalogue	Supplier	Supplier's ERP system e-Marketplace Supplier sell-side application Printed catalogues	Electronic, in various formats Printed catalogues	. Determination of vendor and available supply . RFQ preparation
Supply requirements	Buyer	Planning system (e.g. MRP/ERP, supply-chain management system or spreadsheet)	Electronic, in various formats	. RFQ preparation . Order placement
Demand	Buyer/Supplier	Generated by buyer in procurement system	Electronic RFQ and order	. Preparation of quotation . Commit capacity and inventory . Fulfil order
Inventory and capacity availability	Supplier	ERP, planning system	Electronic or hard copy	. Preparation of quotation . Commit available and capacity inventory . Fulfil order
Supplier company information and commercial terms	Supplier	Supplier's ERP system e-Marketplace Supplier sell-side application Printed catalogues	Electronic or printed	. Quotation preparation . Supplier selection
Quotation	Supplier/ Buyer	Generated by supplier in response to buyer's RFQ	Electronic	. Supplier selection . Purchase order generation
Supplier performance	Buyer	Procurement system	Electronic	. Supplier selection
Approved vendor list	Buyer	Procurement system	Electronic	. Supplier selection

B. E-Procurement Domain Model

1) Purchase Management



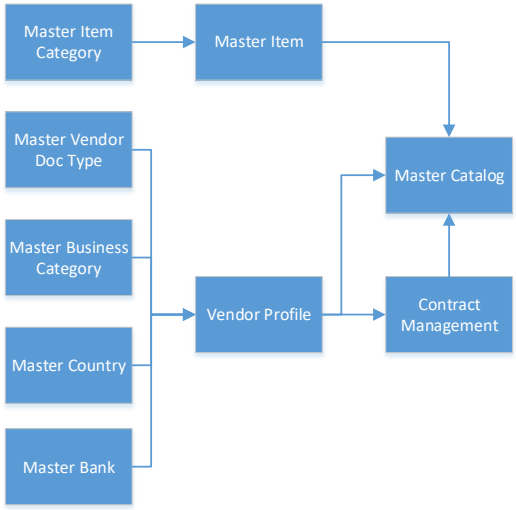
Domain Model Purchase Management



2) Bidding Management

Domain Model Bidding Management

3) Supplier Management



Domain Model Vendor Management

C. E-Procurement Application Architecture

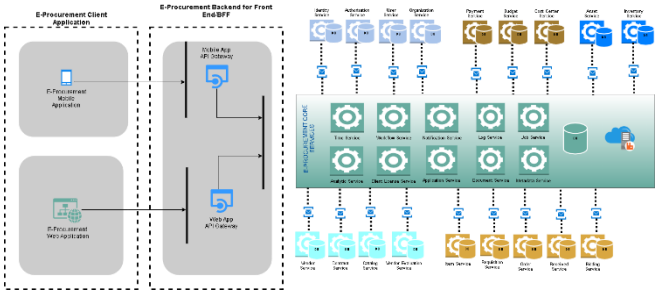


Fig5. E-Procurement Application Microservices Architecture

The MSA Design for the E-Procurement application in figure 8 above for the Gateway API is designed to treat web and mobile users differently. For the data layer, data storage technology is carefully chosen according to business capabilities. Communication between services is handled by the event bus. Leaving aside technology, this is the most commonly used integration pattern in micro-service based applications.

To handle models and large teams, DDD can be applied to

the above architecture. It deals with large models by dividing them into different and explicitly bound contexts about their reciprocal relationships and subordinate domains. This restricted context can be transformed into a separate MSA at the application design level.

Decentralized data management in the above architecture is designed with the aim of avoiding that when some services use a shared data scheme, it can create tight coupling at the data layer. To avoid this, each service must have its own data access logic and store separate data. The development team is free to choose the data persistence method that is most suitable for each service and data nature.

VII. CONCLUSION

Considering that the architecture used always brings its own challenges, the architecture under review is made to meet several objectives depending on the demand for services used. Each architecture is explained in a way that leads to the next proposed architecture. Actually, this is not about following what is the most modern and trendy, it is important to investigate whether the application fully functions efficiently using the chosen architecture. The advantages and disadvantages of MSA are illustrated to show whether an application developed will require a number of capabilities, integrity, durability, and dexterity in return for known and unknown consequences. Concludes that the MSA is able to fulfill most of the features, but this comes with many challenges.

REFERENCES

1. Cognizant, "Overcoming Ongoing Digital Transformational Challenges with a Microservices Architecture," Cogniz. 20-20 Insights, no. november, pp. 1–9, 2015.
2. Wikipedia, "Microservices," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Microservices. [Accessed: 22-Apr-2019].
3. W. P. Lestari and A. Sujarwo, "DevOps : Disrupsi Pengelolaan ICT Pendidikan Tinggi," Semin. Nas. Apl. Teknol. Inf. 2018, pp. 26–31, 2018.
4. Ecommercewiki, "Procurement Process," 2019. [Online]. Available: http://en.ecommercewiki.info/logistics/procurement_process. [Accessed: 01-Apr-2019].
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. Worldbank, "Project: Strengthening Public Procurement in Serbia," 2016. [Online]. Available: http://pubdocs.worldbank.org/en/579181449177314554. [Accessed: 20-Feb-2019].
8. Y. Chang, H. Markatsoris, and H. Richards, "Design and implementation of an e-Procurement system," Prod. Plan. Control, vol. 15, no. 7, pp. 634–646, 2004.
9. E. Wolff, Microservice: Flexible Software Architecture, First prin. Addison-Wesley, 2017.
10. C. de la Torre, B. Wagne, and M. Rousos, NET Microservices Architecture for Containerized NET Applications. Washington: Microsoft Corporation, 2019.
11. M. Kalin, Java Web Services: Up and Running, Second Edi. O'Reilly Media, Inc., 2013.
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. 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.
14. X. Larrucea, I. Santamaria, R. Colomo-palacios, and C. Ebert, "Microservices," Ieee Softw., vol. 35(3), pp. 96–100, 2018.
15. M. Wasson and N. Schonning, "Build microservices on Azure," Microsoft, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/microservices/introduction. [Accessed: 01-Apr-2019].
16. 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].
17. E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley, 2003.
18. 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.
19. 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].
20. J. Fritzsch, "From Monolithic Applications to Microservices : Guidance on Refactoring Techniques and Result Evaluation," Reutlingen University, 2018.
21. E. Risthein, "Migrating the Monolith to a Microservices Architecture: the Case of TransferWise," TALLINN UNIVERISTY OF TECHNOLOGY, 2015
22. Dailynews, "Supply chain procurement challenges" 2019. [Online]. Available: http://www.dailynews.lk/2016/06/21/business/85206. [Accessed: 12-Nov-2019].
23. Richardson, Chris (November 2018). Microservice Patterns. Chapter 1, section 1.4.1 Scale cube and microservices: Manning Publications. ISBN 9781617294549.

AUTHORS PROFILE

Tomy Firmansyah, Earned Bachelor Degree from Information Systems, STMIK AKAKOM Yogyakarta in 2012. Currently Member of Graduate Program-Master of Information System Management in Bina Nusantara University. His Email is: tomy.firmansyah@outlook.com.

Abiyyu Ganeswangga, Earned Bachelor Degree from Informatics Engineering, Telkom University in 2017. Currently Member of Graduate Program-Master of Information System Management in Bina Nusantara University. His Email is :abiyyu.ganeswangga@gmail.com.

Dr. Ahmad Nurul Fajar, Doctore in Computer Science, graduated from Faculty of Computer Science University of Indonesia (UI) in 2014. In 2001, he is graduated from Gunadarma University majoring in Informatics. Master of Science Informatics was completed in 2004 at Bandung Institute of Technology (ITB). Currently, he is the Faculty Member of Binus Graduate Program, Department of Master in Information System. , Research Interest in Software Engineering, Software Development, Information System Analysis and Design, Business Process, and Service Oriented Architecture His email is afajar@binus.edu.