Domain-Driven Microservice Architecturefor Designing Modern Applications

Fawaz Alsolami

Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Summary
For the last several years, information technology (IT) industry uses monolithic architecture in their solutions, but it has several limitations in meeting the demands of modern businesses of the digital world such as high coupling, difficulty in scaling of the application, adding newer technologies to the application, etc. To address the requirement of the modern business need of the digital world, the IT industry is looking to an improved architecting methodology which can be easy to scale and is reliable too. Microservices are unique techniques for designing and architecting software systems that have recently gained its popularity in the software industries as it addresses modern business demands of the digital world. The Microservice Architecture divides a system into small and independent modules, each responsible for performing a specific functionality. It emphasizes scalability, independence and the cohesiveness of each of its components. In this paper, the domain-driven design process is put in place to keep the technical stuff at such a level that everyone involved understands the model, including the customer who, in many cases, does not know much about software development. This paper intended to discuss this new trend in software architecture, its adaptation process, striking characteristics and how it is different from monolith architecture. Further, a case study was conducted to explore domain-driven microservice development and deployment by applying separation of concern to specific boundaries within microservices as suggested by domain-driven design.

Key words:
Microservices, Domain Driven design, Microservice Architecture, Distributed Software Architecture, service-oriented architecture.

1. Introduction

In recent years, almost all major computer-based systems have been distributed and are cloud-based. To develop these distributed and complex system, the IT industry uses different software architecture. The architecture of a software illustrates a high view of the system in terms of framework, components, modules and their interactions. The architecture of the software assists software engineers and analyst to design a high view of the system.

For several years, the IT industry has been using monolithic or SOA-based architecture in their solutions, but in order to address the modern business demands of the digital world, this approach has a huge limitation, overtime applications have a tendency of growing and this nature make the system complex, huge and highly coupled. To address the requirement of the modern business need, the IT industry is looking for an improved architecting methodology which can be easy to scale and is reliable too.

Microservices adopts Agile development to the project rather than the enterprise-wide reuse supported by SOA. This architecture is flexible, and it provides developers with the liberty to independently develop and deploy services for delivering enterprise application [1][2] [3].

Microservice architecture is a technique of developing software solutions as a collection of small, independent, modular deployable services run on a unique process and use lightweight APIs to communicate with each other to serve the required business goal [4]. Domain Driven Design (DDD) is a software design methodology that can help to construct a pattern. It gives tools and building blocks needed to discover and develop a model and turn it into a working application. Besides, it also provides a comprehensive and systematic approach to software design and development. It is common in most projects, that different stakeholders speak different languages as per domain, for example, the technical team uses a language and words that are more technical for describing the system which is different from the language and words used by the businesspeople. This language barrier between different stakeholders results in a situation where it is very hard for them to get a conversation about how a real system works and how system functionalities are understood and written. The key to this approach is to communicate the user directly to define a ubiquitous language. A domain may have its ubiquitous language. For example, in a hospital domain, there may be several sub-domain, appointment and reservation domain, inpatient and outpatient domain, health insurance domain and so on. Further, the objective is as technical and business team communicates the same language when coming up with system models or code like naming classes, data members, methods, certain events as described in the business domain, etc. Domain Driven Design offers a bounded context concept which is like a solution of a system in term of its domain i.e. large software application is broken down into multiple services based on domain and sub-domain. Each of these multiple services also known as Microservices focuses on individual business goal for that particular domain. [5][6].

Manuscript received June 6, 2020
Manuscript revised June 20, 2020

The motivation for this research is focused on the lack of software architecture tools that help designers and implementation of domain-driven design using high-level software models based on input specifications written in natural language and taking into account the specific context of the work problem. A case study is conducted to evaluate the effectiveness of the proposed Model. The study results showed significant improvement in the architecture of the system when using the domain driven approach.
This paper is organized as follows: Section I describes the introduction of the paper, sections II discusses the Microservices and overview of Monolith and Microservices Architecture, section III examines a case study on adapting monolithic to Microservice architecture using the domain driven approach. Section IV summarizes the findings and lastly, Section V concludes.

2. Overview of Monolith and Microservices Architecture

2.1 Monolith Architecture

Monolithic architecture as shown in Fig. 2 is the setup used for traditional client server-side systems. The entire system's functionality is based on a single application. Software applications are composed of components that can be providing services from different deployment systems such as web services.
Domain-specific functionality in a monolithic software framework is usually divided into functional layers such as presentation, business, and persistent. This type of application development has been strongly influenced by hardware, mainly deployment platforms, on if physical or virtual devices.
Most of the IT projects infrastructure are static even when virtualized and the software application written for this infrastructure are sized and design for hardware specific, any hardware failure could result into a tailspin of the entire application.
For example, a typical three-tier client-server architecture, where each tier is decomposed to minimized hardware dependency and counter agility and interdependency as shown in Fig. 1. A three-tier architecture for an Internet banking system. The client is the presentation tier, the Web-server is business and application processing tier and Database is persistence tier. Each of which can be design developed and deployed independently and has its own monolith.

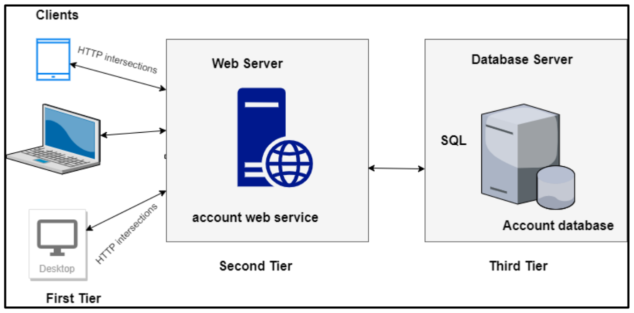

Fig. 1 Three-tier architecture for an Internet banking system [7]

The main issue with this architecture is inefficiency to decomposed services separately within each independent tier. Each tier has different services that provide services for the tier, these services are mainly tightly coupled by the developer of the system.

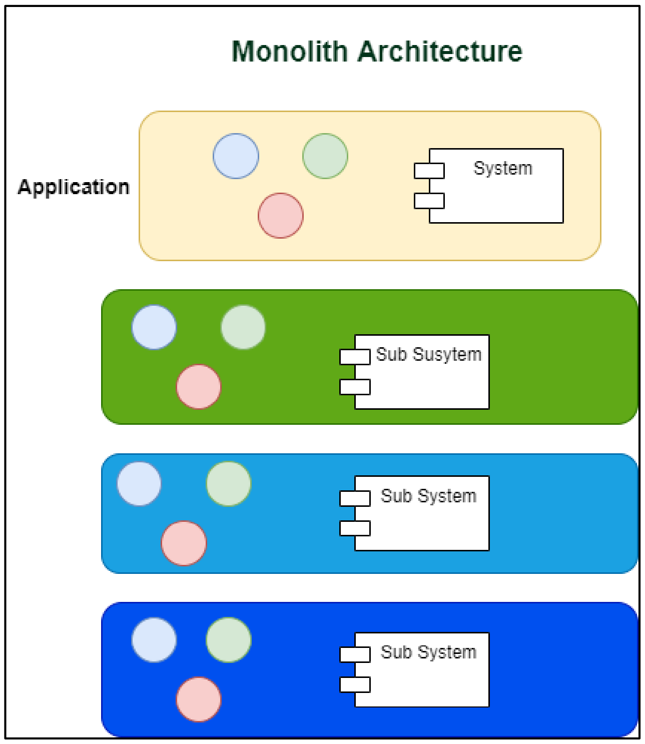

Fig. 2 Conceptual Architecture for a Monolithic based application [8]

A change made even it is minor in any services resulted in the redistribution and redeployment of the whole tier. Rigorous tested is required for any changes made as the risk of service failure is very high due to the tight coupling of services within a tier.
if we see legacy hospital management business model which is based extensively upon monolithic applications for almost every function involved like appointment, in-patient, accounting to filing insurance plans, providing services to

the patients, and providing required features to doctors and paramedical staff. Thus, a bug/error in these systems could stop the whole hospital system. Designing entire operation around one piece of critical infrastructure can cost big — but not testing that piece's performance and failover in the worst-case scenario will cost even more.

2.2 Microservices Architecture

Adaptation of monolith *architecture* approach, results in easy to start development. But as the application starts growing bigger and complicated over time, it becomes highly difficult to be agile and productive.
Microservices is an approach to architecting a software application. This is when the large software application is broken down into multiple services. Each of these multiple services is known as Microservices. Each of these services focuses on single business goal and it is developed and deployed independently. Each of the services in Microservices is loosely coupled which means it is capable of operating independently alone without coordinating with other services in the application. It makes the services development and deployment very quickly and easily with shorter time.

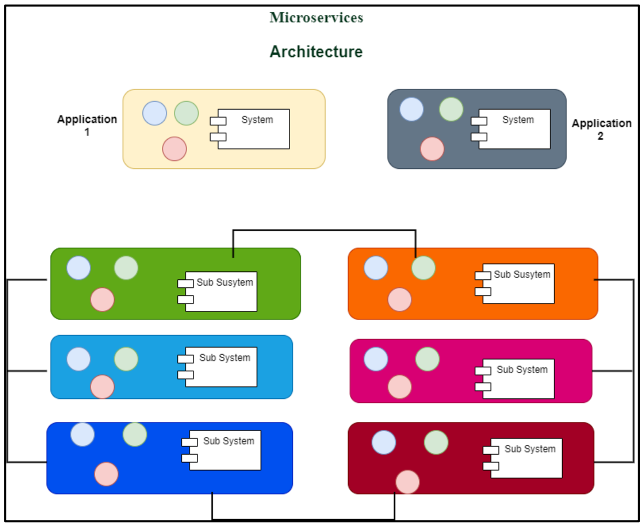

Fig. 3 Conceptual Architecture for a Microservices based application [8]

Each of the services in Microservices is small and focused this means it just addresses the single business functionality of the application and executes that business functionality only as shown in Fig. 3. The main goal is to implement each service using its own technology stacks and this makes it easy to maintain.
Microservices are language independent which means each service can be developed in any programming language and use any framework, for example, certain services can be developed quickly using a popular library rich programming language like java or android others can be developed using C or C++. Further, these services can be developed purely independently without any knowledge of other service architecture or implementation details within an application [9][10].
Microservices with the right basic setup could prove to be more internally robust, and might more easily allow for a distributed and robust cloud system to be developed. It scales out by independently deploying individual services, creating instances of these services across servers/virtual machines/containers as shown in Fig. 3.
Communication with microservices is a process for exchanging services with one another, rather than limiting them to communicating with clients and data stores and data sources. HTTP and REST are used as Request/response communication. When a client uses request / response communication, a request is submitted to a service, then the service processes the request and returns a reply. In particular, request / response communication is well suited for querying data from client apps.

3. Related Work

The current trend in the field of microservices shows that Strategic Domain-driven Design (DDD) has become a popular trend for decomposing domains into so-called Bounded Contexts. Kapferer [11] proposed a modular and extensible component architecture for a modeling framework based on Domain-driven Design (DDD). The model employs Context Mapper DSL (CML) modeling language along with tools support to reverse engineer DDD Context Maps and decompose them iteratively. The proposed framework offers to build Domain-driven Design (DDD) model and improve its productivity at the same time. The author validated the model and further improved the prototype iteratively. The author carried empirical research techniques such as action research and case studies to validate the usefulness and effectiveness of the proposed modeling framework.
Yussupov et. al. [12] proposed a multistep technique for automatically evaluating the portability of serverless applications with respect to specific platform, called SErverless Applications PORtability assessmenT (SEAPORT). It is a canonical serverless application model which offers ways for evaluating portability related to components similarity calculation and classification, along with static code analysis. The technique is compatible with prevailing migration concepts and use it as an additional part for serverless use cases. The authors evaluated the proposed technique by implementing an available open source prototype through GitHub.
Vikram [13] proposed model for the domain of Abusive Head Trauma (AHT) using Deep Learning methods and create an AI-driven System at scale using best Software Engineering Practices. AHT relates to injuries inflicted to

skull or brain as a result of violent shaking or some deliberate impact. The model categorizes AHT in different sub-domains such as Medical Domain Knowledge, Data Gathering, Data Pre-processing, Image Generation and Classification, Creating APIs, Kubernetes and Containers. Data gathering and its pre-processing were performed under the guidance of radiologists and trauma researchers. Tests were executed employing Deep Learning models. It used Deep Convolutional Generative Adversarial Network (DCGAN) for Image Generation, and Pre-trained two-dimensional and customized three-dimensional convolutional neural networks (CNNs) classifiers for the classification tasks. After training, the models were exposed using the Flask web framework as APIs, contained using Docker and deployed on a Kubernetes cluster. The study results are evaluated based on the precision of the models, the practicability of their implementation as application program interfaces (APIs) and load testing the Kubernetes cluster. Load Testing of the model shows that the auto-scalability feature of the cluster to act as a high number of requests.
Schneider et. al. [14] introduced a concept for splitting the domain model into several diagrams based on the Unified Modeling Language (UML) profiles. The prosed approach is based on dividing the domain model into multiple sub-models which allows to develop different models simultaneously, and this way helps to reduce the complexity of the modeling process. Under this concept, a software modeling tool based on Object Management Group Unified Modeling Language (OMG UM) called Enterprise Architect (EA) is used as modeling tool which allows direct conversion of the model into code. Besides, base classes are employed for entities and value objects to provide suitable functionalities for these domain concepts. The limitation of this work is that the model must be manually converted into code. Currently, this concept can only automatically generate the classes, methods, and attributes.

4. Case Study - Domain Driven Design

4.1 Domain storytelling

[15]: Domain driven design is based on domain storytelling as shown in fig. 5 which is a technique of collaborative modelling that emphasizes how people work together. Its primary purpose is to translate domain knowledge into software for business. This purpose is achieved by bringing together people with different backgrounds, and by telling and visualizing stories, enabling them to learn from each other. Domain Stories
are told from the perspective of an actor. For example, an actor may be a person, a group of individuals or a software system. In the Domain Story all actors have in common that they play an active part. Various icons are used to represent certain actors, Actors create, work with, and exchange objects like documents, physical objects, and digital objects for work.
Domain storytelling follows the numbers to read the story as shown in fig 5. Following are the details of the processing the seat request use case:

a) Customer request for seats.
b) Seat distributer agent looks up the UI screen plan in the Hajj system to find available seats.
c) LC approval manager will give approval of seats.
d) Seat distributer agent get notification of the approval and list of buses given to customer.
e) Seat distributer agent will print the trip sheets and give it to the customer.
f) Gate pass will be issue by the system and will be verified by the gate pass agent.

U*biquitous language* is used as a model to serve as a universal language between developers and the domain experts to help communicate with ease. Ubiquitous language is a set of terms and their relations between the specialists and technical experts. Boundaries are defined by applying the *bounded contexts* of the domain. Every bounded context includes a domain model representing a particular subdomain of the complex application as shown in fig 4. Domain layer is mainly used to place all business use case and business logic of the system [16][17].

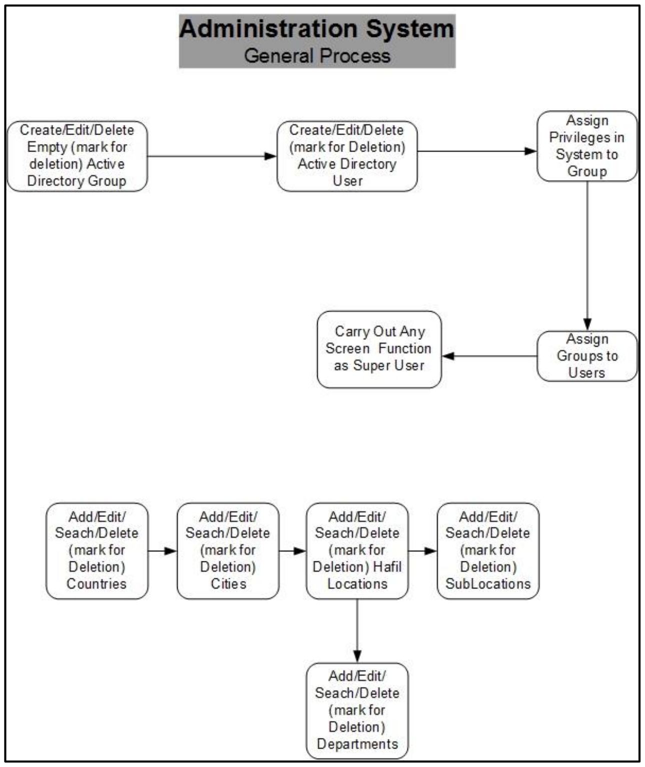
Administration System
General Process
Create/Edit/Delete Empty (mark for deletion) Active Directory Group
Create/Edit/Delete (mark for Deletion) Active Directory User
Assign Privileges in System to Group
Assign Groups to Users
Carry Out Any Screen Function as Super User
Add/Edit/ Seach/Delete (mark for Deletion) Countries
Add/Edit/ Seach/Delete (mark for Deletion) Cities
Add/Edit/ Seach/Delete (mark for Deletion) Hafil Locations
Add/Edit/ Seach/Delete (mark for Deletion) SubLocations
Add/Edit/ Seach/Delete (mark for Deletion) Departments

Fig. 4 Decomposing the administration use cases

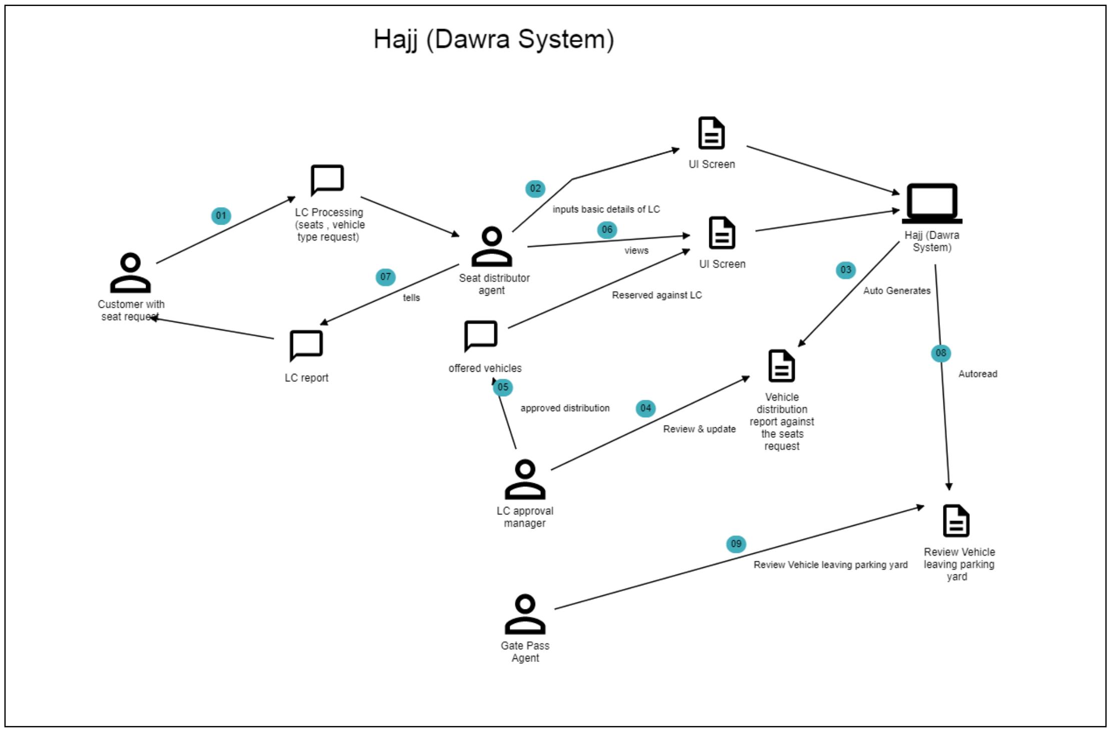

Fig. 5 Domain Story for seat distribution use case of the Hajj System

4.2 Case Study

This research presents a real-life adoption and implementation of domain driven Microservice architecture in a Saudi-Arabia based transport organization. The existing framework is based on a monolithic architecture that structures the entire project around one central piece of infrastructure. The changes are how to address the changes request that have arrived late during system execution. System architecture entire operation around one piece makes it hard to apply the changes and often in the system crashes during peak time due to errors, further the current system has lots of missing requirements. In order to establish suitable context boundaries required for splitting a monolith application into microservices, we need to analyze the responsibilities and behavior of the current system.
Following are the sub-systems of the system. Hajj Dawra apps, Employee Referral app, Contracts App, Assets App, Human Recourse Apps, etc.
Details workflow of the assets process is depicted in Fig. 6. A Dawra system is used to transport pilgrims during the Muslim Hajj season (which is annual gathering in Makkah) via buses. Dawra system accept LC which contain a total number of pilgrims mainly in 10,000s need to transport four places involved in Hajj are Mecca, Mina, Arafat, and Muzdalifah. System automatically suggests total buses required to transport pilgrims without any seat loss by choosing different type of buses like 49 / 37/ 20-seater buses etc. Receiving contract, distributing buses based on the seat request, managing trips, monitoring buses, etc. are some of the core requirements of the system.
The Employee Referral System as shown fig. 7. is another sub system that focuses a job opening in the private/public domain. The job opening system has the following exact requirements. Users with only a valid login/password will be allowed to post a job. The job list would be on intranet from where referee would be able to pick the job and refer the candidate. User will be allowed to see the job listings.
An authorized user will be allowed to edit the job details. Job details will include title, skill, specialization area, experience, job description and some other details. A job description will describe required profile for the candidate. The job will be listed on the public domain/internet/intranet for the resume referral.
Transport Employee will get an index screen with the options like add job, list job, view job and edit job. Each option will lead to a new screen from where a job can be added/listed/viewed/edited. Adding and editing of jobs would be based upon the privileges granted to personnel.

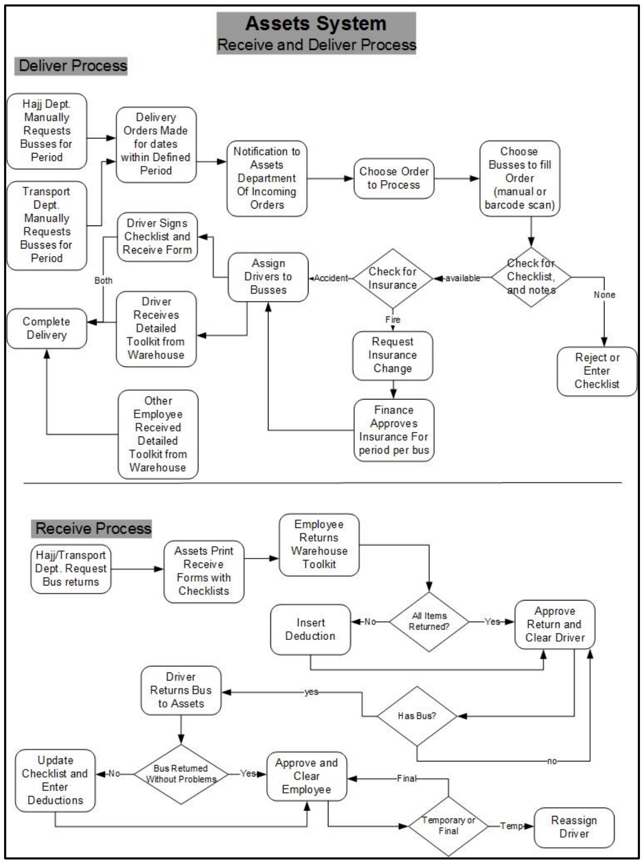
Assets System
Receive and Deliver Process
Deliver Process
Hajj Dept. Manually Requests Busses for Period
Transport Dept. Manually Requests Busses for Period
Delivery Orders Made for dates within Defined Period
Notification to Assets Department Of Incoming Orders
Choose Order to Process
Choose Busses to fill Order (manual or barcode scan)
Check for Checklist, and notes
Check for Insurance
Assign Drivers to Busses
Driver Signs Checklist and Receive Form
Driver Receives Detailed Toolkit from Warehouse
Complete Delivery
Other Employee Received Detailed Toolkit from Warehouse
Request Insurance Change
Finance Approves Insurance For period per bus
Reject or Enter Checklist
Both
Accident
available
Fire
None
Receive Process
Hajj/Transport Dept. Request Bus returns
Assets Print Receive Forms with Checklists
Employee Returns Warehouse Toolkit
All Items Returned?
Insert Deduction
Approve Return and Clear Driver
No
Yes
Driver Returns Bus to Assets
Has Bus?
yes
no
Bus Returned Without Problems
Update Checklist and Enter Deductions
Approve and Clear Employee
Temporary or Final
Reassign Driver
Final
Temp

Fig. 6. Workflow of the Assets process of Hajj system.

An employee can add a job to public/private Intranet domain. The referee can refer the resume to public/private internet domain. If a referred candidate succeeds to get the job, the finance department will reward the referee. Fig. 5. shows business functional specifications of employee referral subsystem.

ADD JOB SCREEN: This screen would enable the user to add a new job. This screen will allow entering the fields like job Id, title, technology, description and ADC.

LIST JOB SCREEN: This screen would enable the user to list all the jobs listed for the current requirement along with the number of resumes arrived for a particular job. It displays the active and archived jobs based on different criteria. This screen will also allow to add a new job or select any of the listed jobs to edit or view.

VIEW JOB SCREEN: This screen would enable the user to view the job Id, title, technology, description and ADC in details. This screen can be viewed from index screen and list screen.

EDIT JOB SCREEN: This screen would enable the user to edit any posted job, and this screen would be accessible from index and list screen.

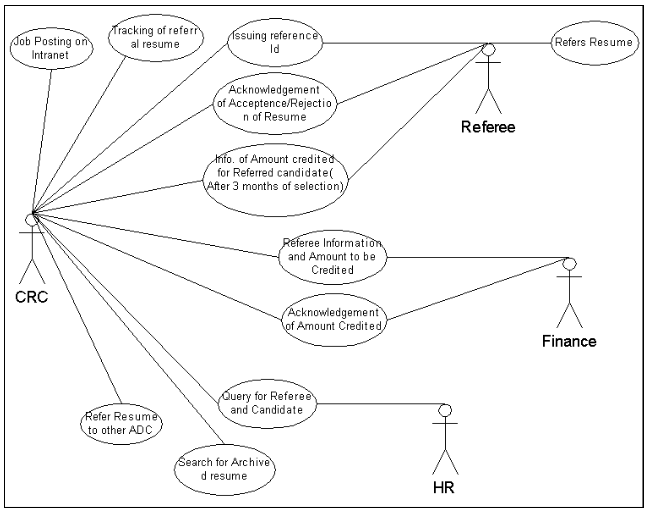
Job Posting on Intranet
Tracking of referral resume
Issuing reference Id
Refers Resume
Acknowledgement of Acceptence/Rejection of Resume
Referee
Info. of Amount credited for Referred candidate(After 3 months of selection)
Referee Information and Amount to be Credited
CRC
Acknowledgement of Amount Credited
Finance
Query for Referee and Candidate
Refer Resume to other ADC
Search for Archived resume
HR

Fig. 7 Business functional specifications of hajj employee referral sub subsystem.

Microservice is a comparatively smaller and decoupled service that relies heavily on one business functionality activity at a time as compared to monolithic applications offering all business functionality in a single unit for a complete system, which results in tightly coupling in the code base application [18]. Microservices are continually changing; services go up and down as development teams make changes, requirements can be added or updated in response to the business need of the application as shown in fig 8. Microservices architecture needs a highly durable service discovery solution to handle dynamic environment. We decomposed our system, based on domain and subdomain, by nouns and business capability.

The following is a quick description of some of those common patterns:

HAJJ SERVICE **-** The Hajj Service contains the current location (server: port) of all instances of the service being deployed. A Hajj service communicates when and if a service is available for the use, and determines whether if a service is alive.

HAJJ UI SERVICE - Provides a user interface including authentication mechanisms, routing, load balancing, and incorporates clients into API gateways.

API GATEWAY **-** A server that represents the single-entry point into an application or integration point system. It serves as a gateway to a software user Interface services.

HR SERVICE **-** A HR service communicates when a service is free and available, and decides if a service is active and alive. The HR Service contains the current location (server: port) of every instance of a program being deployed.

MAIN WEB UI SERVICE - provides web skeletons that build pages/screens by composing multiple user interface and integrates clients into API gateways.

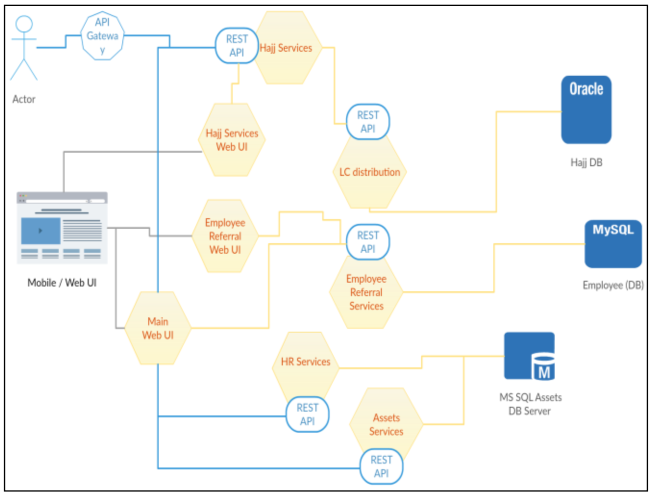

Fig. 8. Application decomposed into Microservice

5. Summary of findings

This case study examined for the duration of five to six months after the initiative to move from the monolith architecture to microservices. The results were analyzed to find where they met the expectations and where they diverged.

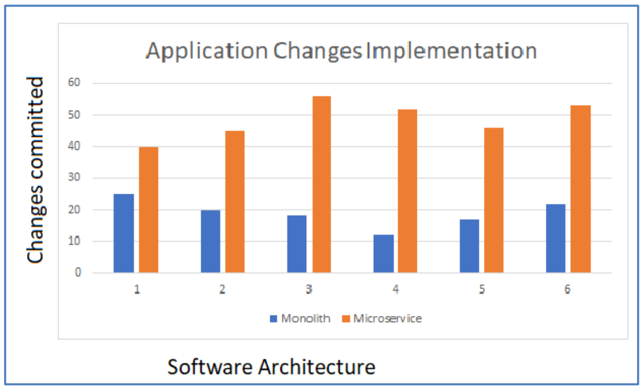

Fig. 9 Comparing the changes implemented week wise using Monolithic and domain driven microservice architecture.

The study also focused on those areas which contribute to a system's evolvability, and its rate of change. The technicality of the system was considered based on the services installed for the product, and the number of deployments for each service. It was also recorded the number of times changes were made for a product. With this business property, the number of applications, and source code perform subtitling. The results demonstrate the raw data as well as the technological commands to collect the data. Figs (9 and 10) illustrates how the number of production deployments gets divided into multiple services, before and after we started our work. Before moving to microservices, the number of deployments per week is collected over the five to six weeks duration for a monolith application. Once moved to the Microservice architecture, data are continued to be collected in the new system design for the number of deployments of the monolith service (but now with reduced functionality being filled in by extracted Microservices).

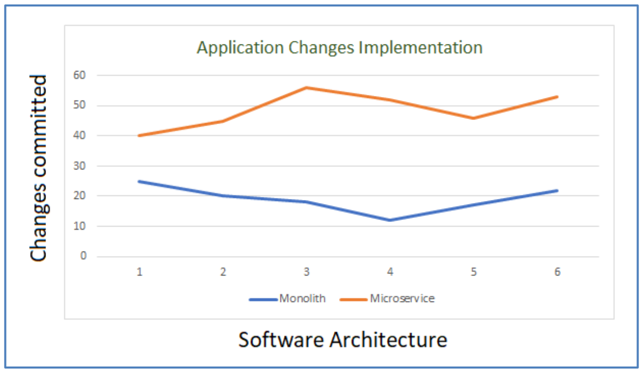

Fig. 10. Comparing the changes implemented week wise using Monolithic and domain driven microservice architecture.

It is recommended to move to the application to a Single-Page Application style and separate data from data presentation by creating a model layer that manages data and a view layer that states from the models.

Conclusion

Microservices are considered as a new style of developing enterprise architecture which is cloud-based and can be built, managed separately. Microservices helps to reduce the complexity by applying work breakdown structure, i.e., breaking down large project structure into smaller, more manageable chunks which can be developed and implemented using loosely coupled modules that can communicate with each other through simple application programming interfaces (APIs). The paper carried a case study to explore the usefulness and effectiveness of service development and deployment of Microservice. It is concluded that Microservice is an isolated service that allows changes in the system at a faster rate allowing the business to bring new features and products to market quicker.

References

[1] P. Jarman, “Microservice- A New Application Paradigm .” [Online]. Available: https://www.infosys.com/digital/insights/Documents/microservices-application-paradigm.pdf. [Accessed: 18-June-2020].
[2] G. Mazlami, J. Cito, and P. Leitner, “Extraction of Microservices from Monolithic Software Architectures,”

2017 IEEE International Conference on Web Services (ICWS), 2017.

[3] C.-Y. Fan and S.-P. Ma, “Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report,” 2017 IEEE International Conference on AI & Mobile Services (AIMS), 2017.

[4] T. Huston, “What is Microservices Architecture?,” SmartBear. [Online]. Available: https://smartbear.com/learn/api-design/what-are-microservices/. [Accessed: 17-June-2020].

[5] “Microservices at Netflix: Lessons for Architectural Design,” NGINX, 03-Aug-2017. [Online]. Available: https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/. [Accessed: 18-June-2020].

[6] “Ebay Archtecture 2015 - Deep Lessons from Google and eBay...,” Ebay Archtecture 2015 - Deep Lessons from Google and eBay on Building Ecosystems of Micr 1 of 15. [Online]. Available: http://highscalability.com/blog/2015/12/1/deep-lessons-from-google-and-ebay-on-building-ecosystems-of.html. [Accessed: 11-June-2020].

[7] I. Sommerville, Software engineering. Boston: Pearson/Addison-Wesley, 2004.

[8] Msfussell, “Introduction to microservices on Azure,” Introduction to microservices on Azure | Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices#comparison-between-application-development-approaches. [Accessed: 05-June-2020].

[9] M. Fowler, “Microservices,” martinfowler.com. [Online]. Available: https://martinfowler.com/articles/microservices.html[Accessed: 16-June-2020].

[10] A. I. Khan, M. M. Alam, N. Qayyum, U. A. Khan, "Empirical Study of an Improved Component Based Software Development Model using Expert Opinion Technique", International Journal of Information Technology and Computer Science (IJITCS), vol.5, no.8, pp.1-14, 2013.

[11] K. Stefan (2020) A Modeling Framework for Strategic Domain-driven Design and Service Decomposition. Masters thesis, HSR Hochschule für Technik Rapperswil.

[12] Yussupov, V.; Breitenbücher, U.; Kaplan, A. and Leymann, F. (2020). SEAPORT: Assessing the Portability of Serverless Applications.In Proceedings of the 10th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, ISBN 978-989-758-424-4, pages 456-467. DOI: 10.5220/0009574104560467

[13] V. Aditya.; (2020) Designing an AI-Driven System at Scale for Detection of Abusive Head Trauma Using Domain Modeling ,Arizona State University, ProQuest Dissertations Publishing, 2020. 27956820.

[14] Schneider, M., Hippchen, B., Giessler, P., Irrgang, C. and Abeck, S., 2019. Microservice development based on tool-supported domain modeling. In The Fifth International Conference on Advances and Trends in Software Engineering.be combined on Context Maps.

[15] Hofer, S.; Schwentner H. (2020), Domain Storytelling A Collaborative Modeling Method, leanpub

[16] M.M. Alam, A.I. Khan, and A. Zafar, An Empirical Study of the Improved SPLD Framework using Expert Opinion Technique., (IJEACS) International Journal of Engineering and Applied Computer Science, Volume: 02, Issue: 03, March 2017 ISBN: 978-0-9957075-4-2

[17] M.M. Alam, A.I. Khan, and A. Zafar, An Empirical Study of the Improved SPLD Framework using Expert Opinion Technique., (IJEACS) International Journal of Engineering and Applied Computer Science, Volume: 02, Issue: 03, March 2017 ISBN: 978-0-9957075-4-2

[18] A.I. Khan., R.J. Qurashi, and U.A. Khan, A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies. IJCSI International Journal of Computer Science Issues,, 2011. 8(4): p. 441-450.