Adoption Advantages Of Micro-Service Architecture In Software Industries

J. Prathap Irudayaraj, Saravanan.P

Abstract:The popularity of the developing application using microservices architecture is gaining more attention because of the independent modules, scalable, maintainable, monitorable, Continuous Integration (CI), Continuous Delivery (CD) and better fault isolation. This architectural style that breaking large software projects into small independent and loosely coupled services. So, this approach produces independently deployable service which adopts its own communication mechanism for intercommunication. This paper proposes benefits and set of guidelines while adopting the microservice architecture along with the cost benefits.

Index Terms: Software Architecture, SOA, Cloud, Microservice, benefits, microservice migration checklist

1 Introduction

He latest trend in the software development industry adhering the process of defining the software service designs in a more modularized and independently operatable, through adopting the microservice architecture. This increases the befits of maintaining and developing the software more agile. There are three major notable befits are faster delivery, improved scalability and greater autonomy. So, both organization and developer are benefited without any doubt. Software Engineering updates the well-defined scientific principles, methods, and procedures help to build an efficient and reliable software product. Microservice can be one among them. Before Microservices, the monolithic typed software development ruled the software industry. As the web-based software development started to evolve the need for the newer architecture because of the limitation in the monolithic typed applications. So, Service-Oriented Architecture-based software development began. Most of the software that is running is based on the Service Oriented Architecture(SOA).

2 Methodology

The research methodology used in the is paper is Qualitative research Methodology which performs the case study on the existing design standards like monolithic, Service-oriented Architecture(SOA) and analyzed the advantages of them. And collected the feedback and suggestion of the various category peoples like end-users, software developers, testers, business analysts and organization's management peoples. The case study research is conducted through surveys and innovation technical discussions. This approach brought an in-depth study of the existing context. Awareness of the problem The research to aware of the problem faced by the usages of the cloud architectures like monolithic and SOA and propose to adopt the Microservices for its advantages. Development The existing theories are analyzed compared with each other for their benefits. And the survey is conducted to know the awareness of the Microservices architectures.Data Collection Data were collected through interviews, online surveys and documentary materials. Set of written questionnaires were asked to know their opinions and feedback on the usages of the architectures.

- *Prathap Irudayaraj is an M.Phil. Research Scholar at D.B.Jain College (Autonomous), Thoraipakkam, Chennai, India. E-mail: prathapjayaraj@gmail.com*
- *Saravanan.P is Assistant Professor, D.B.Jain College (Autonomous), Thoraipakkam, Chennai, India. E-mail: npsindian@yahoo.co.in*

The interviews were conducted in different categories like product owner, developers, testers, QA, DevOps Engineer, Network Engineer, IT Engineer, Support Engineer. Data Analyze The collected data were categorized and grouped according to the data collection group. Data Mining techniques were performed to clean up the data and similarities were identified and again grouped. Evaluation The method was evaluated using different case studies which solve the customer and developer's problem. The final collected data were processed and matched in the cased studies of microservice adaptation and advantages. Conclusion The final design of the method was produced and presented in this paper as a way of communication and adoption guide to moving to Microservices.

3 Problem Statement

Good software requires a constant update in the technology so that users don't get bored. So, change is inevitable not only because of the requirement changes but also to adopt the recent technology stock and standards. The type of adaptation of the change is called modernization which requires understanding the existing system and transforming with the latest changes without major impact in the running software. I conducted research to know the issues in the existing architectures and feedback and opinions from different categories of people in the industries were analyzed, to get the benefits of modernization of the software.

3.1 Surveyed Audience

Software Developers Testers DevOps Engineer Product Owner Feedback from the application users Random anonymous People from Online and tech forum

3.2 Survey Questions

The below questions were circuited over the online survey, interviews to get the opinion. What is your role Share your opinion on your current development process In your current job role, what can be simplified? Do you expect the application to be updated with a modern look and feel often? How often you would like to see the changes. Specify. What is the crusial part of development? Mention all your challenges in your job role? How do you upgrade your software? Do you use any automation to speed up your work? As a product owner, what are all your concerns in developing and maintaining the software? As a developer/tester express your concern for the betterment of the software development process. Do you follow architecture styles for development?

3.3 Survey Responses

The below said points were collected using data clean up techniques using data mining from the 1002 survey responses. Want to see the changes deployed soon production. Want to adopt automation. Whole Application should not be down for an upgrade. Easily monitorable. Deployment time should be less. And the ability to increase the load balance. Interoperable and portable with third-party software should work vice-versa Don't want to invest time in training the people repeatedly for business study. Wanted to simplify so that their own organization employees are enough to maintain the software at their control. Test cases should be automated and build should deploy When there is an error, should be notified. Even the application down, need the ability to cache the content. Single developer should be able to manage the development. Both UI and servicer related works should go in parallel. Team should be able to manage the development with minimum support. The said points were collected from the survey which I've conducted in the Social Media, GitHub as feedback and opinions. These responses to be analyzed, to over come the issue by adopting the microservices-based architecture.

4 Architecture of Microservice

From the problem statements the said points were collected from survey results of 800 responses. Consolidating on the points we can say the people who expressed the concerns are using the monolithic and SOA based architectures. So I'm listing the advantages of the microservice-based architectures.

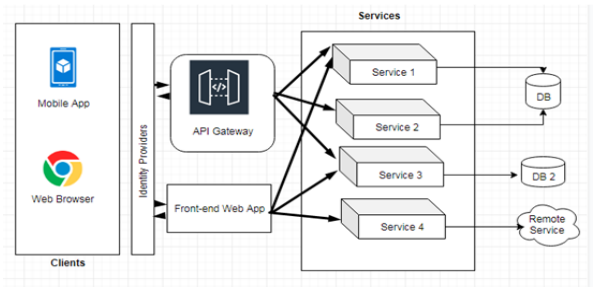

Fig 1 Architecute of Microservice

5 Advantages of adopting Micro Services

From the problem statements the said points were collected from survey results of 800 responses. Consolidating on the points we can say the people who expressed the concerns are using the monolithic and SOA based architectures. So I'm listing the advantages of the microservice-based architectures.

5.1 Technical Advantages

The advantage of adopting microservices benefits the below category technically.

6 Adaptation Advantages

6.1 Technical Advantages

The technical advantages of adopting this architecture are listed as per the category.Service Autonomy- This will break the component as undependable and isolated loosely coupled service which can be deployed independently. 2. Front-end integration - As you break the more independent services, it is easier to reuse and integrate with all front-end services and exposed as third party services. 3.Resource Monitoring and Management - Monitoring of the software is always a big problem in software maintenance. As the software is developed with more complexity and highly coupled with more layer. It because difficult to maintain and track the place of error occurrence. When you adopt the MSA based architecture each service can be watched and the service's health check status can be automated. This ability will facilitate to scale the apps on-demand and as well as raise the notification on the unstable health check. 4. Failure and recovery - When there is any failure with the system, it is easy to replace and upgrade. The breakages/outage of the service will not break the entire application. The requests that were received during the downtime can be cached and reprocessed. Also, the ability to scale the application horizontally will increase the chance of high loads to the same service. 5. Continues Integration & Delivery - As Martin Fowler says in his blog, "Being able to swiftly deploy small independent units is a great boon for development, but it puts additional strain on operations". So, as you complete your development, the DevOps process should automatically trigger the build and run the test cases. On successful completion, the build should be deployed in the configured container. 6. Cross Team Interaction - As you start breaking the independent modules, each service can be owned by the team without any dependency and give the flexibility to work simultaneously by the minimal interaction with other teams. This gives the full stack developer to own the complete modules including UI, Services layers. This is also giving the opportunity to improve the communication between developments, QA and testing teams. 7. Automation - In Microservices based architecture, automation can be embraced easily. The repetitive works like deployment, integration, unit tests, service health monitoring, error notification, self-repairing on build failure can be automated through DevOps process 8. Reduces the complexity – Keeping the whole code and maintaining will be really overburdened to the process. This makes more coupled code will create complexity in the development and deployment. Also, it stops the processing of adapting to the latest technology stack. When adopting microservice it, smoothen the processing complexity of upgrade is performed. 9. Decentralized Governance and Management – Monolithic application standards enforce the single technology platform which might look easy at the beginning but it will become the bottleneck over the period as it started growing. The monitoring of the application will be a big problem when it goes to the maintenance phase. Microservice preferred to manage its own database for each service. So, data management and decentralization makes easy.

6.2 Cost Benefits

Saves Training Cost - As we don't depend on the other services while developing the software. The adaptation of the MSA helps to remove the dependency of the domain experts. To work on the application, you don't need to be an expert in the domain. Anyone can work independently with just mock data. As the inputs and outputs are well defined. The investment in training cost can be reduced.Unused Services - The cost involved in running the low priority services can be scaled down. This helps to reduce 40% of the computing power and experienced 20-50% cost savings overall.

3. Organized Around the Business – This is the point where Microservice bases architecture plays a very important role in adding more value to the business. High demanded services alone can be increased instead of deploying the whole systems. As demand decreases, we can also reduce the server. This avoids unnecessary cost to the product maintenance and issue of a server outage because of high demand. The increase and decrease of the service instances can be also automated which gives the opportunity to adopt a carefree technique in software maintenance.

7 Migration Guide to Microservice Architecture

Before you migrate to microservice-based architecture while modernizing the existing application the below category points needs to be considered while architecting new Morden Microservice-based application. I have drafted this checklist as a yes or no type questions. The keypoint of adopting the Microservices is the notation of the independently replaceable and upgradeable capability.

7.1 Service Definition

Is the service granular enough to break without dependency? Is the service able to handle its own defined task (single business activity)? Is it a self-contained and independently develop and deploy unit? Does the service encompass all necessary resources to support the business activity? Does the service have a proper service boundary and does not expose the internal details to the outside of the service boundary? Is the service easily upgradeable? Is the service independently scalable? Is the usage documentation for each service properly writte Can a single team own the development of the services from inception to production?

7.2 Service Interactions

Does the service use a standard mode of communication like HTTP and REST-based JSON? Does the microservice have a well-defined interface for data exchange? Does the service support design for retries and fallback? Does the service take into account the failures of dependent services?

7.3 Performance, Exception Handling, and Scalability

Is the service resilient? Can the service be independently deployed and scaled? Are the health and performance of the application monitored independently? Is there a centralized logging mechanism to utilize log visualization tools? Is the performance monitored? Is load-balancing ensured ? Is elasticity handled? Are timeouts managed? Can metrics like infrastructure, file system operations, inbound requests, responses, integrations, and external services possible to monitor? Does the service capable enough to publish metrics on its uptime and response time? So that the monitoring infrastructure take the corrective measures? Does the service have timeouts with default behavior implemented?

7.4 Integration Considerations

Do you follow the enterprise standard communication methods? Do you exchange messages through the service by the required headers like reply-to queues? Do you send a communication through HTTP which are available over a pull API? Do the events raised by the service defined in JSON format?

7.5 Hosting/Monitoring

Can the service be provisioned programmatically on short notice? Can the monitoring infrastructure able to detect the service operation status? Can the monitoring infrastructure detect a degradation in the SLA based on the metrics/heartrate the service is publishing? Can the notification infrastructure raise an appropriate notification to the concerned team and alerts when it gets a failure log message? Does the monitoring system keep statistics of uptime or failures, requests serviced and show them inappropriate dashboards in a visualized format? Is the service easily discoverable? Is the service runtime status in the registry reflected if the service is started or stopped?

8 Future Work

The below can be improvements can be made to the existing work To Improve the existing proposed method and to find the other use cases that beneficial in desigining better Microservices architecture approach. As IoT related concepts are also trending, need to perform further analysis in the area. Similar case studies needs to be performed and benchmarked in IoT related domains. Further consideration checklist to utilize API gateway, load balancing, monitoring, log configuration needs to be carried The similar survey should be conducted to understand the concern of the Microservice users/developers/testers/IT Engineer/Support Engineers. Studies needs to be benchmarked with “Serverless Computing" which doesn’t requires a server to manage, instead use functions and source code directly as the unit of execution

9 Conclusion

The advantages of adopting microservice-based architecture give the opportunity of delivering the project, more efficiently and quickly by the characteristics of microservices. This also clearly indicates and removes the most of customer reported feedback from the survey. The simplified configuration is made possible of the easy maintenance of the software owning cost at their own lab. Instead of paying a huge amount for the maintenance cost. Further research to improve the method designed in the paper by using different domains needs to be performed to enhance the result.

10 References

[1] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. In Present and Ulterior Software Engineering. https://doi.org/10.1007/978-3-319-67425-4_12

[2] Linthicum, D. S. (2016). Practical Use of Microservices in Moving Workloads to the Cloud. IEEE Cloud Computing, 3(5). https://doi.org/10.1109/MCC.2016.114

[3] *“Microservices – a definition of new architectural form”* http://www.martinfowler.com/articles/microservices.html *accessed on 24 July 2019*

[4] http://microservices.io/ *accessed on 24 July 2019*

[5] Butzin, B., Golatowski, F., & Timmermann, D. (2016). Microservices approach for the internet of things. IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2016-Novem. https://doi.org/10.1109/ETFA.2016.7733707

[6] Kecskemeti, G., Marosi, A. C., & Kertesz, A. (2016). The ENTICE approach to decompose monolithic services into

microservices. 2016 International Conference on High Performance Computing and Simulation, HPCS 2016, 591–596. https://doi.org/10.1109/HPCSim.2016.7568389

[7] González, O., Sánchez, M., Universidad de los Andes (Bogotá, C., & Institute of Electrical and Electronics Engineers. (n.d.). 2015 10th Computing Colombian Conference (10CCC) : Universidad de los Andes, Bogotá, Colombia, September 21st to 25th, 2015.

[8] Jamshidi, P., Pahl, C., Mendonca, N. C., Lewis, J., & Tilkov, S. (2018, May 1). Microservices: The journey so far and challenges ahead. IEEE Software, Vol. 35, pp. 24–35. https://doi.org/10.1109/MS.2018.2141039

[9] Escobar, D., Cardenas, D., Amarillo, R., Castro, E., Garces, K., Parra, C., & Casallas, R. (2017). Towards the understanding and evolution of monolithic applications as microservices. Proceedings of the 2016 42nd Latin American Computing Conference, CLEI 2016. https://doi.org/10.1109/CLEI.2016.7833410

[10] Chandramouli, R. (n.d.). Security Strategies for Microservices-based Application Systems. https://doi.org/10.6028/NIST.SP.800-204-draft

[11] Larrucea, M. X., Santamaria, I., Colomo-Palacios, R., & Ebert, C. (n.d.). SOFTWARE TECHNOLOGY.

[12] Di Francesco, P., Malavolta, I., & Lago, P. (2017). Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017, 21–30. https://doi.org/10.1109/ICSA.2017.24

[13] Mazlami, G., Cito, J., & Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017, 524–531. https://doi.org/10.1109/ICWS.2017.61

[14] Christian Esposito, Aniello Castiglione, Kim-Kwang, & Raymond Choo. (n.d.). Challenges in Delivering Software in the Cloud as Micro services.

[15] Johannes Thones. (2015). Microservices. Retrieved from http://www.ieee.org/publications_

[16] Andy Singleton. (n.d.). The Economics of Micro services. https://doi.org/2325-6095/16/$33.00

[17] Alan Sill. (n.d.). The Design and Architecture of Micro services.

[18] J. Prathap Irudayaraj, Saravanan. P, Published a paper titled “Study on Cloud Computing and Cost-Benefit Analysis ” in INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) (www.ijrar.org | UGC and ISSN Approved), Vol. 6, Issue 1, January 2019, E-ISSN 2348-1269, P- ISSN 2349-5138. Pp 367-371. http://www.ijrar.org/papers/IJRAR19J1520.pdf. Impact factor: 5.75. UGC J.No:43602.

[19] J. Prathap Irudayaraj, Saravanan. P, Published a paper titled “Evolution of the Single Page Application in the modern web application development", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN: 2349-5162, Vol.6, Issue 3, Pp141-145, March-2019. http://www.jetir.org/papers/JETIR1903425.pdf. Impact factor: 5.87. UGC J.No: 63975.

[20] J. Prathap Irudayaraj, Saravanan. P, Published a paper titled “Comparative study on Cloud Software Architecture: Monolithic, SOA, Microservice", Journal of Applied Science and Computations (JASC), ISSN: 1076-5131, Vol.6, Issue 5, Pp2257-2261, May-2019. http://j-asc.com/gallery/289-may-2937.pdf. Impact factor: 5.8. UGC J.No: 41238.

[21] Pradeep G Pillai, Saravanan.P, Published a paper titled “ Significance of Wireless Multi- Hop AD-HOC Networks ” in International Journal of Computer Sciences and Engineering (IJCSE), Vol. 6, Issue 8, Aug 2018, e-ISSN(Online): 2347-2693. Pp 161-167.http://www.ijcseonline.org/pdf_paper_view.php?paper_id=2671&27-IJCSE-04523.pdf. Impact factor:3.022. UGC J.No:63193.

[22] Rajeswari.C, Saravanan. P, Published a paper titled “NETWORK SECURITY CHALLENGES ” in Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and ISSN Approved), Vol. 5, Issue 9, Sep 2018, ISSN: 2349-5162. Pp 279-282. http://www.jetir.org/papers/JETIR1809057.pdf. Impact factor: 5.87.

[23] S.Peachibalan , Dr.Balaji.S , P.Saravanan, Published a paper titled “Smart Grid Using Wireless Sensor Network’s and Routing Protocol” in International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), Vol. 6, Issue 8, August 2017, ISSN(Online): 2319-8753, ISSN (Print): 2347-6710. Pp 16574-16579. https://www.ijirset.com/upload/2017/august/264_70_SMART.pdf. Impact factor:6.209.

[24] Pradeep G Pillai, Saravanan.P, Published a paper titled “ A Study on Air Pollution Monitoring System using Wireless Sensor Network ” in International Journal of Trend in Scientific Research and Development (IJTSRD), Vol. 2, Issue 2, Jan-Feb 2018, ISSN(Online): 2456-6470. Pp 343-347. http://www.ijtsrd.com/papers/ijtsrd8374.pdf Impact factor:4.101.