# Migration of Monolith Applications to Miniservices: A Case Study from the Telecom Domain

Ümit Kanoğlu[1][a], Ali İmre[1][b] and Oumout Chouseinoglou[2][c]
[1]Türk Telekom, Ankara, Turkey
[2]Department of Industrial Engineering, Hacettepe University, Ankara, Turkey

Keywords: Service Computing, Miniservices, Microservices, Miniservice Migration.

Abstract: More organizations are considering the transformation of their existing monolithic applications to microservices in order to increase competitiveness and utilize the benefits of new software architectures which meet their business needs. However, due to detailed and extensive requirements of the microservice architecture (MSA), organizations either implement microservices at different granularity levels or decide not to undertake this migration even though the business need is evident. Miniservices have been proposed as an intermediate alternative between monoliths and microservices, with a larger scope of services and more relaxed architectural constraints. This paper introduces the concept of miniservice architecture (MnSA) to the industry domain, proposes a methodology to be implemented for the migration of a monolith application to MnSA and shows the applicability of this methodology with a detailed case study from the telecom domain.

## 1 INTRODUCTION

Large enterprises over the years of their operation generally accumulate a software inventory of large and elephantine monolithic systems. Continuously changing business requirements and environments lead to these monolithic systems getting inevitably larger and implementing changes while trying to meet budget and schedule constraints but also satisfying quality, availability, and reliability levels can be very challenging (Levcovitz, Terra, & Valente, 2015) (Dragoni, et al., 2017). One of the newest proposed alternatives to the monolithic approach is the microservice architecture (MSA), based on the service-oriented computing and defined as "an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms" (Lewis & Fowler, 2014). Even though MSA has been proposed as a solution for the inefficiencies and problems caused by the monolith architecture and a strong industry interest in migrating legacy systems to MSA exists (Di Francesco, Lago, & Malavolta, 2019), MSA can also be a high-cost, disruptive, and often less predictable undertaking for many enterprises due to the lack of experienced development teams (Christudas, 2019). As a deficient implementation of MSA would not fully resolve the already existing problems of a monolithic application, the miniservice architecture (MnSA) has been proposed as an intermediary architectural approach. MnSA, compared to monolithic architecture aggregates functionality related to a specific domain, whereas compared to MSA, MnSA has a larger scope and more relaxed architectural constraints, and may or may not use independent data (Christudas, 2019) (Thomas & Gupta, 2017). In other words, a miniservice can be described as a structure that may consist of functionalities which could be decomposed to more than one microservice and share databases (DB) with other miniservices, thus allowing a more relaxed design and development for the organization with the realization of the advantages of the MSA. The introduction of the concept of miniservices has a specific importance as findings from the industry show that not every implementation of microservices adheres to the characteristics of "pure microservices", and service granularity is one of the characteristics

[a] https://orcid.org/0000-0001-8245-1355
[b] https://orcid.org/0000-0001-9477-0660
[c] https://orcid.org/0000-0002-8513-351X

Kanoğlu, Ü., İmre, A. and Chouseinoglou, O.
Migration of Monolith Applications to Miniservices: A Case Study from the Telecom Domain.
DOI: 10.5220/0009408103900397
In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020)*, pages 390-397
ISBN: 978-989-758-421-3

not followed in many microservice applications (Bogner, Fritzsch, Wagner, & Zimmermann, 2019).

As in the case of microservices (Baresi, Garriga, & De Renzis, 2017), when planning a migration of a monolith system to an MnSA or when designing an application from scratch considering an MnSA, discovering the level of granularity and cohesiveness that would be adequate for the miniservices is an important problem that needs to be addressed by the people responsible for the migration. To the best of our knowledge, this is the first paper that proposes a methodology for the migration of an existing monolithic application to an MnSA, and displays the applicability of the proposed approach with a case study. However, it should be noted that the proposed methodology is not a technical decomposition but a high-level design and analysis, based on the business needs of the organization evaluating the migration to miniservices. The contributions of this research are therefore threefold; we introduce the alternative of MnSA as proposed by Christudas (2019) and Gartner Research report (Thomas & Gupta, 2017) for organizations that want to migrate to MSA but cannot handle the detailed and extensive requirements of microservices, we propose a migration methodology to MnSA and finally evaluate its applicability with a case study.

The aforementioned case study was conducted on an enterprise application in Türk Telekom, one of the largest telecommunication companies and the first integrated telecommunication operator in Turkey, with a history of 178 years. As of September 2019, Türk Telekom provides telecommunication services to 14.6 million land line, 11.3 million broadband, 3.6 million TV and 22.8 million mobile subscribers. In this paper, the Customer Problem Management (CPM) (TMForum Frameworx, 2019) application from the enterprise application inventory of Türk Telekom that serves all the aforementioned customers has been selected, and the migration of CPM to MnSA is examined and documented as a case study, with a specific focus on the lessons learned throughout the project management of this migration process. The CPM application was developed initially as a monolithic system with the aim of centralizing in an end-to-end way numerous different service failure management processes running on different systems at Türk Telekom. CPM is providing the Service Failure Management Product which facilitates the failure management in different broadband services of Türk Telekom (e.g. xDSL, PSTN, IPTV, Data, P2P, etc.), allows reception of failures from different channels, manages the sending and receiving of failure notifications, stores and monitors end-to-end records of failures, facilitates problem management, provides inter-team workflows, and has the capability of generating executive reports that may support future investment decisions and further decisions at strategic level. As CPM manages the failures submitted by customers, its accessibility level is required to be at the highest. Moreover, as a result of the continuously introduced new products and services, and rapidly changing legislations and technology in the telecom domain, CPM is required to meet the changing functional requirements of both internal and external customers. However, as CPM is a monolithic application, the deployment of any changes to the live environment is only possible by stopping the whole CPM and all associated services.

The rest of this paper is organized as follows: Section 2 provides a brief literature review of related studies and approaches from the microservices domain, Section 3 describes the proposed methodology, Section 4 summarizes the migration of the CPM application to an MnSA as a case study, and finally Section 5 concludes the findings, gives the evaluation of the proposed approach by a team of developers, summarizes the lessons learned and refers to the planned future work.

## 2 LITERATURE REVIEW

The migration literature on MnSA is almost non-existent as the concept of miniservices is considerably new compared to microservices. Similarly, no case studies documenting the success of migration to an MnSA exist. However, a number of studies have addressed the problem of decomposing an already existing monolithic application to microservices or documented case studies of migration from monolithic applications to MSA. Even though there are notable differences between the concept of MSA and MnSA, we believe that similar studies in the domain of microservices would shed light to better understanding the implementation of miniservices.

Several software products exist that have been developed as solution to the problem of microservice identification, such as Service Cutter (Service Cutter; Gysel, Kölbener, Giersche, & Zimmermann, 2016) or graph based analysis by using Neo4J GraphGist (Bastani, 2015). A systematic mapping study by Di Francesco, Lago and Malavolta (2019) by examining 103 studies regarding microservices displays the current trends in the microservices research and the gap between academy and industry in this domain.

Baresi, Garriga and De Renzis (2017), by utilizing a clustering approach, propose a solution to the problem of identifying microservice granularity based on the semantic similarity between predicted and available functionality of the application, with a success rate of 80% in correct identification of microservices. Levcovitz, Terra and Valente (2015) propose a six-step technique to identify microservices on existing monolithic systems and successfully display the applicability of their proposed approach on a 750KLOC real-world monolithic banking system. Their proposed approach has as initial step the evaluation and classification of the DB tables of the monolith system into business subsystems, thus not being entirely applicable to a miniservice viewpoint where the organization may not require the decomposition of the DB. Use cases or user interfaces have also been proposed (Richardson, 2014) to be used for the decomposition of monolithic applications to microservices. Fritzch et al. (2019), by examining the MSA migration of 14 systems from different business domains, identify as an important drawback the fact that most organizations preferred to rewrite their codebase instead of splitting it as they had highly complex legacy systems. The authors argue that in some cases this was because of the absence of a suitable decomposition approach, and therefore a major technical challenge was finding the right service cut. Mazzara et al. (2018) present a real-world case study regarding the migration to MSA of a mission critical system from the banking industry. The authors identify and propose a repeatable migration process that can be used to convert a monolithic application coming from the financial domain (with the characteristics of legacy systems and batch-based processing on heterogeneous data sources) into an MSA.

## 3 PROPOSED METHODOLOGY

As part of this study, a methodology to assess the feasibility to migrate an existing monolithic application to an MnSA is proposed. This migration methodology is not a technical decomposition but instead a high-level design and analysis, to be utilized based on the business needs of the organization when evaluating the migration to miniservices, and is in accordance with the ITIL framework (What is ITIL, 2019).

The steps followed in the migration process are given in a generic fashion and in the most general form in Figure 1. In the first step of the migration process a development team that is knowledgeable about MSA and the application to be migrated is formed and the application is logically decomposed to miniservices. During this decomposition process the mission and aims of the application, code isolation in an MnSA basis, the integration of the application with other applications, a viewpoint supportive of continuous integration, delivery and development (Microsoft, 2019) and the DB structure are considered. Following this step, all updates done in the previous year regarding the application in question are collected from the respective change logs and they are assigned to each proposed miniservice. For each update an effort is estimated based on the components of the miniservice that they are assigned to and the historical effort data of this update. In the third step the previous year's updates are evaluated based on the changes they resulted on the DB DDL. If the updates have resulted in too many DB changes, the decomposition of the DB at a miniservice level is evaluated and discussed. At the fourth step the proposed miniservices are examined with respect to

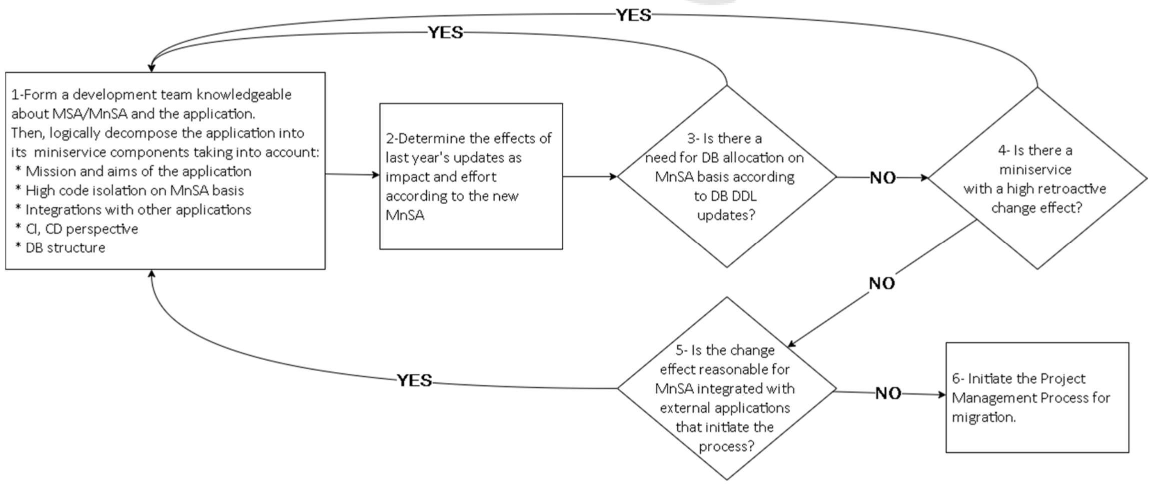
YES
YES
1-Form a development team knowledgeable about MSA/MnSA and the application. Then, logically decompose the application into its miniservice components taking into account:
* Mission and aims of the application
* High code isolation on MnSA basis
* Integrations with other applications
* CI, CD perspective
* DB structure
2-Determine the effects of last year's updates as impact and effort according to the new MnSA
3- Is there a need for DB allocation on MnSA basis according to DB DDL updates?
NO
4- Is there a miniservice with a high retroactive change effect?
NO
5- Is the change effect reasonable for MnSA integrated with external applications that initiate the process?
YES
NO
6- Initiate the Project Management Process for migration.

Figure 1: Steps of the followed migration process.

the number of updates that they have been assigned and if there are miniservices with excessive number of updates, a new logical MnSA decomposition is reconsidered. In the next step, the integration of the miniservices with external applications is examined: if the impact of the external applications concentrates on a single miniservice the decomposition process is executed again to form a new MnSA decomposition. In the final step, if the decomposition of the application to the proposed miniservices is acceptable with respect to the aforementioned steps by the team, the project management process within the organization is initiated.

It should be noted that further steps may be added to the proposed approach with respect to the specific characteristics of the application being evaluated and the overall requirements of the organization. The main aim, however, should be realization of the architectural and technological advantages of the MnSA through the migration of the application to this new architecture.

## 4 CASE STUDY

### 4.1 Current Architecture (AS-IS)

In this section, the migration process from monolith to MnSA of CPM (TMForum Frameworx, 2019), an enterprise application that resides in the Customer domain of Türk Telekom as specified by the TMForum Application Framework (Application Framework - TAM, 2019) is documented. The existing version of CPM was developed according to monolithic architecture; that is development was conducted on a single codebase in Java and the application runs with a single DB. The monolith CPM application consists of approximately 200 KLOC, 2,681 Java classes and 305 DB tables, and has approximately 3,000 active users who execute an average of 30,000 transactions per day. This version of the CPM is being used for the last five years (2014-2019) and has accumulated a DB size of approximately 20 TB. When the planned and realized application updates are examined it is observed that during the last year of operations, 85 update requests that required a total of 1,377 person/day effort were realized and put to live, resulting to 22 planned shutdowns of the live system that lasted approximately a total of 35 hours. Figure 2, in order to better describe the importance of the CPM application, displays the enterprise applications within Türk Telekom that CPM is related to and works in an integrated fashion. As CPM is a monolith,

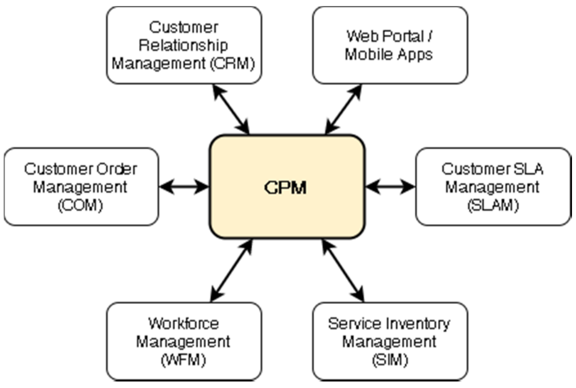
Customer Relationship Management (CRM)
Web Portal / Mobile Apps
Customer Order Management (COM)
CPM
Customer SLA Management (SLAM)
Workforce Management (WFM)
Service Inventory Management (SIM)

Figure 2: Relationship of CPM with other enterprise applications in the organization.

even a single update or change in the code requires the update of the whole application. Due to the predefined enterprise application update policies, during planned updates all servers and access to CPM are shutdown. Moreover, in case of a problem in the implemented update the application in all servers needs to be rolled back.

### 4.2 Proposed Architecture (To-Be)

Considering the problems existing with the monolith CPM, a migration of the application to MSA was initially evaluated. With the employment of an MSA approach it was intended to deploy updates more rapidly and in a more flexibly fashion, decrease the downtimes resulting from updates, and eliminate the rollback process complexity in case of problems after the deployment of updates. According to the MSA principles each microservice is required to be able to execute independently, needs to be associated with a single functionality or task and data independence is mandatory. However, as a result of the evaluation it was decided that an MnSA approach would be more appropriate for the migration from the monolith architecture, and the migration methodology described in Section 3 was followed.

The miniservices were identified by a migration team of five software professionals, consisting of one project manager, one solution architect, one operations manager and two software developers. The migration team have decomposed the monolith CPM to miniservices by considering the existing codebase, their experiences with the functionality of CPM and the requirements of the organization by this migration. One of the most important requirements considered was the minimization of the downtimes resulting from application updates. Following the steps identified in Figure 1, the technical components of the application were grouped and the miniservices were decomposed logically in such a way to support continuous integration (CI) and continuous deployment (CD). Following these steps, two major MnSA versions were obtained, namely MnSA-1 and MnSA-2, respectively shown in Figure 3 and 4.

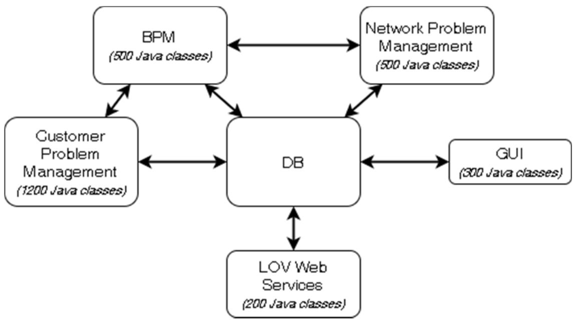

Figure 3: Proposed MnSA-1.

In order to evaluate the efficiency of the proposed MnSA's, the updates realized during the last one year were associated with the miniservices proposed in each MnSA alternative. Table 1 and Table 2 show how each one of the 85 updates and their actual effort is associated with the MnSA alternatives developed by the migration team. This miniservice and update association was realized by evaluating the update details by examining the update logs, the functionalities that an update was related to, how these functionalities were distributed to each miniservice and by using the expert knowledge of the team on how the update effort would be distributed to the detailed components of each update.

Table 1: Analysis of the Proposed MnSA-1.

| Changes | Number | % | Effort as Person/ Day | % |
|---|---|---|---|---|
| Total CPM + DB Changes | 59 | 69 | 1058 | 77 |
| Only BPM Changes | 3 | 4 | 16 | 1 |
| Only LOV Changes | 6 | 7 | 20 | 1 |
| Only GUI Changes | 5 | 6 | 65 | 5 |
| Only NPM Changes | 1 | 1 | 13 | 1 |
| Multiple Changes except CPM | 11 | 13 | 205 | 15 |
| Total Changes except CPM + DB | 26 | 31 | 319 | 23 |
| **Total Changes** | **85** | **100** | **1377** | **100** |

When MnSA-1 in Figure 3 is examined, it is observed that 69% of the updates and 77% of the realized effort is associated with the Customer Problem Management miniservice that deals with the problems transferred by other enterprise applications and users to CPM. The migration team evaluated these update and effort numbers to be too high, the methodology given in Figure 1 was repeated taking into account this observation. As a result, the Customer Problem Management miniservice of MnSA-1 was further decomposed to two miniservices, proposing Customer Problem Entry (CPE), a new miniservice that deals with the external integrated enterprise applications. The new MnSA is shown in Figure 4. When Table 2 is examined, it is observed that with MnSA-2, 62% of all updates can be realized without shutting down the CPE miniservice which is vital for the reception of problems communicated by other enterprise applications and users.

According to the MnSA-2 proposal, the monolithic CPM is decomposed consequently to seven miniservices: CPE encompasses the web services that receive user problem notifications by other enterprise applications, Customer Problem Management miniservice manages the overall customer problems process, Network Problem Management (NPM) miniservice are the web services that allow the receipt of problems sent from other enterprise applications regarding the problems in network devices, Business Process Management (BPM) miniservice manages the business processes, List of Values (LOV) miniservice encompasses web services that allow the query of a list of values that exist in the problem management process and define complaint types and problem causes by the external WFM enterprise application, GUI miniservice is the interface of the CPM application, and finally the DB miniservice.

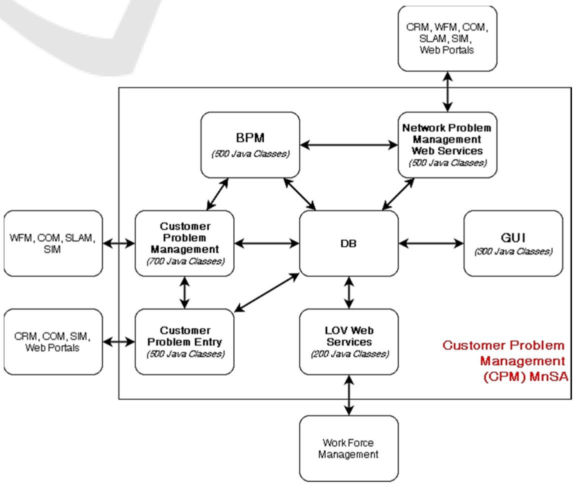

Figure 4: Proposed MnSA-2.

Table 2: Analysis of the Proposed New MnSA-2.

| **Changes** | **Number** | **%** | **Effort as Person/ Day** | **%** |
|---|---|---|---|---|
| Total CPE + DB Changes | 32 | 38 | 565 | 41 |
| Only CPM Changes | 13 | 15 | 133 | 10 |
| Only BPM Changes | 3 | 4 | 16 | 1 |
| Only LOV Changes | 6 | 7 | 20 | 1 |
| Only GUI Changes | 5 | 6 | 65 | 5 |
| Only NPM Changes | 1 | 1 | 13 | 1 |
| Multiple Changes except CPE | 25 | 29 | 565 | 41 |
| Total Changes except CPE + DB | 53 | 62 | 812 | 59 |
| **Total Changes** | **85** | **100** | **1377** | **100** |

As the data integrity has been evaluated by the migration team to be of high importance and the percentage of updates that have affected the DB during the last year have been identified as only 12%, it was decided that there is no necessity of decomposing the DB at the miniservice level, thus justifying the reasoning of using an MnSA approach instead of MSA. Figure 4 also shows the number of Java classes that are associated with each proposed miniservice.

## 4.3 MnSA Migration Process

In order to migrate to the proposed MnSA-2, the codebase of the monolithic CPM is required to be decomposed with respect to the proposed miniservices. At this point, it was assessed by the migration team that there is no need for changing the tools and techniques used by the development team for the realization of this migration, but it would suffice to manage the development, compile and deploy steps of each miniservice separately. However, if the application servers are migrated to a container structure then the development tools should be updated to alternatives that would be compatible with the container structure. It is proposed that in the ideal case, similar to the MSA, the communication between miniservices should be realized with messages, however, it is argued that if required this communication can be done through RESTful web services. Moreover, a separate miniservice development team may be formed for each miniservice according to the MnSA approach. In the current case study, it was decided by the migration team that a single development team would manage all proposed miniservices, and that as the organizational experience regarding MnSA increases and the teams mature on the migration and management of miniservices future migration efforts may be undertaken with the specialization of separate teams at the miniservice level.

## 4.4 MnSA Operational Processes

As required by different legislations, the change management processes in Türk Telekom are being governed by the ITIL framework (What is ITIL, 2019). A foreseen further gain of the MnSA migration, from the operational viewpoint, is making the change management process more agile. Due to the services that will be designed in a distributed fashion on a cluster structure, it will be possible to make deployments related to the CPM services without affecting the operation of other enterprise applications integrated to the CPM. Moreover, with the use of the automatic test run tools placed on top of CI and CD pipeline it is expected that defects will be resolved more rapidly and new updates will be deployed to the live system in a timelier manner.

However, together with the gains at the operational level, it is expected that the use of new tools and techniques with the novel MnSA would result in experiencing several difficulties at the operational processes, in accordance with the findings of Fritzch et al. (2019). These difficulties are foreseen to be related to the configuration and use of new tools for server management (e.g. Kubernetes), intra-service communication (e.g. Kafka and Zookeeper), and log management (e.g. Elasticsearch, Graphana, Prometheus) by the operational users at the starting level. Moreover, code management of miniservices that have been decomposed with some predefined logic by the migration team and are built on top of a container structure is expected to be more difficult compared to the monolithic architecture alternative. Finally, as the DB in the proposed MnSA is still designed in a relational way, no changes at the operational are expected with respect to this miniservice.

## 5 CONCLUSION

This paper introduces the possibility of migrating monolith applications to MnSA at the industrial setting, consequently proposes a methodology for the migration of monolithic enterprise application software to an MnSA, and applies and documents the implementation of this methodology on a monolithic enterprise application at Türk Telekom to display its applicability at the organisational domain.

As the case study of the proposed methodology, the CPM, an enterprise application of five operation years from the Türk Telekom application inventory is selected. A migration team consisting of five software professionals with experience at both microservices and the CPM application was formed. In order to decompose the monolith application to miniservices, as a first step the codebase of the existing application is investigated. By following the steps of the proposed methodology and taking the codebase structure into account, the basic functions and the processes of the application, together with the experience and knowledge of the migration team on microservices, the first architecture, that is MnSA-1, is proposed. The development of MnSA-1 required a 16 person/hour effort by the migration team. To evaluate the efficiency of the proposed MnSA, the implemented updates of the last year on the CPM application were collected from the change logs of the organization and each update was allocated by the migration team to a miniservice in the proposed MnSA, the effect and effort of each update on each miniservice was estimated. The migration team spent a 24 person/hour effort for this allocation process. Having evaluated the findings of this allocation it was decided that the proposed MnSA can be improved as specific miniservices are overloaded, thus with an extra effort of 24 person/hours the second architecture was decomposed, that is MnSA-2. A project plan was developed and was submitted to the Project Management Office of Türk Telekom together with the feasibility analysis and expected gains of this migration. The migration of CPM to MnSA according to the proposed MnSA-2 has been accepted to the 2020 Master Plan and will be evaluated with respect to the overall Türk Telekom budget and priority requirements.

In order to assess the effectiveness and the possible gains of migrating CPM to MnSA as proposed by migration team, an internal questionnaire was prepared and distributed to the software team of CPM. The development team consists of 16 individuals with an average of professional experience in software development being 12 years and an average experience at the organization being 9 years. The respondents are all university graduates (10 bachelors, 6 graduate level), with 12 of them having a diploma in Computer Science or Computer Engineering. The development team members were asked six questions based on the proposed benefits of microservice migration in the literature and especially selected from the work of Taibi, Lenarduzzi and Pahl (2017), and the responses are given in Table 3.

The questions were specifically adapted to the concept of miniservices. When the responses are examined, even though the proposed architecture by the migration team is not a pure MSA but instead an MnSA, it is evident that the developers believe the application will benefit from this transformation in all six areas, namely agility, deployment risk, distributed development, technology flexibility, server scalability and increased resiliency.

Table 3: Responses of Development Team members regarding the benefits of migrating to MnSA.

| **Question:** *"Migration to this MnSA would...* | **No idea** | **No** | **Some** | **Very** | **Extremely** |
|---|---|---|---|---|---|
| *...increase the agility of the application"* | 1 | 1 | 1 | 10 | 3 |
| *...decrease the risks associated with deployment"* | 2 | 1 | 1 | 10 | 2 |
| *...would facilitate distributed development"* | 3 | 0 | 1 | 6 | 6 |
| *...would allow technology flexibility"* | 1 | 0 | 1 | 8 | 6 |
| *...would increase server scalability"* | 3 | 0 | 0 | 8 | 5 |
| *...would increase server resiliency"* | 5 | 2 | 0 | 7 | 2 |

In the documented case study, the MnSA was developed with the use of approaches and techniques introduced for microservices and MSA, as it is apparent that these approaches may be used for miniservices and migration to MnSA. Moreover, the proposed MnSA is based to the requirements of the organization and a logical decomposition of services. Subsequent to this decomposition, the application itself was decomposed to separate codebases. Considering the requirement of data integrity, a DB decomposition was not proposed as each miniservice will be working with separate parts of the single DB and moreover historical updates that were examined show that DB updates are rare. Each proposed miniservice may be deployed as a separate application or can be managed accordingly with a migration to a container structure. With the migration to this structure the aim was to flexibly manage the integration dependencies between applications, decrease the service shutdowns and associated risks during deployments, and thus adding agility to the development process. Moreover, it is argued that with a container server structure, improvements in availability and scalability are realized. The most evident gain of the proposed MnSA is that CPE, one of the basic miniservices of the proposed MnSA and which is associated with 62% of all updates realized during the last year, can now be updated and deployed without stopping the operation of other services within the CPM application. This would significantly decrease the downtime of the overall CPM application, which was recorded to be 35 hours during the operations of previous year.

The development of the methodology presented in this paper has contributed to the existing organizational know-how on microservice migration and MSA/MnSA transformation. Moreover, the proposed methodology was presented and communicated with different IT units of Türk Telekom to disseminate the microservice and miniservice migration awareness. As a result of the conducted presentations the Enterprise Architecture Unit is considering the addition of the developed methodology to the already existing two-phase migration framework as a third and final step to be implemented in future MSA/MnSA transformation processes within Türk Telekom.

Depending on the success and organizational reception of the proposed MnSA migration of the CPM application, further enterprise applications from the application inventory of Türk Telekom are considered to be transformed following the proposed methodology. As a future study, the applicability of the MnSA migration methodology on applications with different characteristics and requirements, and the findings of these migration processes may be documented and presented to provide a deeper insight on the topic of miniservice migration. Moreover, we are planning to investigate and model a methodology for the required transformation of server infrastructure, business processes and organizational processes to meet the migration to an MnSA.

## REFERENCES

*Application Framework - TAM.* (2019). Retrieved 12 10, 2019, from TMForum: https://www.tmforum.org/application-framework/

Baresi, L., Garriga, M., & De Renzis, A. (2017). Microservices identification through interface analysis. *European Conference on Service-Oriented and Cloud Computing.* Cham: Springer.

Bastani, K. (2015). *Using Graph Analysis to Decompose Monoliths into Microservices with Neo4j.* Retrieved 12 24, 2019, from Kenny Bastani: https://www.kennybastani.com/2015/05/graph-analysis-microservice-neo4j.html

Bogner, J., Fritzsch, J., Wagner, S., & Zimmermann, A. (2019). Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality. *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)* (pp. 187-195). IEEE.

Christudas, B. (2019). *Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud.* Springer.

Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software, 150*, 77-97.

Dragoni, N., Giallorenzo, S., Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering* (pp. 195-216). Cham: Springer.

Fritzsch, J., Bogner, J., Wagner, S., & Zimmermann, A. (2019). *Microservices Migration in Industry: Intentions, Strategies, and Challenges.* arXiv preprint arXiv:1906.04702.

Gysel, M., Kölbener, L., Giersche, W., & Zimmermann, O. (2016). Service Cutter: A Systematic Approach to Service Decomposition. *European Conference on Service-Oriented and Cloud Computing* (pp. 185-200). Cham: Springer.

Levcovitz, A., Terra, R., & Valente, M. (2015). Towards a technique. *III Workshop de Visualização, Evolução e Manutenção de Software (VEM)*, (pp. 97-104).

Lewis, J., & Fowler, M. (2014). *Microservices: a definition of this new architectural term.* Retrieved 12 24, 2019, from martinfowler.com: https://martinfowler.com/articles/microservices.html

Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., Larsen, S., & Dustdar, S. (2018). Microservices: Migration of a mission critical system. *IEEE Transactions on Services Computing.*

Microsoft. (2019). *CI/CD for microservices architectures.* Retrieved 12 1, 2019, from Microsoft Azure: https://docs.microsoft.com/en-us/azure/architecture/microservices/ci-cd

Richardson, C. (2014). *Microservices: Decomposing Applications for Deployability and Scalability.* Retrieved 12 24, 2019, from InfoQ: https://www.infoq.com/articles/microservices-intro/

*Service Cutter.* (2019). Retrieved 12 24, 2019, from Service Cutter: https://servicecutter.github.io/

Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing, 4*(5), 22-32.

Thomas, A., & Gupta, A. (2017). *Innovation Insight for Miniservices.* Retrieved 12 10, 2019, from Gartner Research: https://www.gartner.com/en/documents/3615120

*TMForum Frameworx.* (2019). Retrieved 11 12, 2019, from ILSA: http://www.ilsa.kz/etom/main/tamapplication2.htm

*What is ITIL.* (2019). Retrieved 12 16, 2019, from ITIL: https://www.axelos.com/best-practice-solutions/itil/what-is-itil