H. VURAL

ATILIM UNIVERSITY 2020

ESTIMATION ON OPTIMAL SIZE OF MICROSERVICES

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

HULYA VURAL

DOCTOR OF PHILOSOPHY THESIS
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2020

ESTIMATION ON OPTIMAL SIZE OF MICROSERVICES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

HULYA VURAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2020

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

____

Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Doctor of Philosophy in Software Engineering, Atilim University**.

____

Prof. Dr. Ali Yazıcı
Head of Department

This is to certify that we have read the thesis “Estimation on Optimal Size of Microservices” submitted by Hulya Vural and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

____

Assoc. Prof. Dr. Murat Koyuncu
Supervisor

Prof. Dr. Ali Yazıcı
Software Eng. Department, Atilim University ________________________

Assoc. Prof. Dr. Murat Koyuncu
Information Sys. Eng. Department, Atilim University ________________________

Prof. Dr. Ferda Nur Alpaslan
Computer Eng. Department, METU ________________________

Asst. Prof. Dr. Çiğdem Turhan
Software Eng. Department, Atilim University ________________________

Asst. Prof. Dr. Atila Bostan
Computer Eng. Department, THK U ________________________

Date: 24/01/2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Hulya, Vural

ABSTRACT

ESTIMATION ON OPTIMAL SIZE OF MICROSERVICES

Vural, Hulya
Ph.D., Department of Software Engineering
Supervisor: Assoc. Prof. Dr. Murat Koyuncu

January 2020, 92 pages

Cloud computing is becoming the de-facto standard for enterprises. The monolith applications of the past do not measure up to cloud standards. As a result, less coupled, more agile Microservices Architecture style has emerged. The Microservices Architecture proposes the use of smaller services when compared to traditional service oriented architectures. Since we were introduced to Microservices Architecture, there is an ongoing debate about what the actual size of a microservice should be.

This thesis aims to identify what measures would help in finding the optimal size of a microservice. To achieve this aim, two different domain-driven design examples were taken for empirical analysis. Then those examples were modified to get more granular and less granular microservices. In the end, the more granular, original and less granular examples were compared. During the comparison, afferent coupling, efferent coupling, relational cohesion and COSMIC function points were calculated and compared. The results indicate that afferent and efferent coupling and relational cohesion values are better for deciding the optimal size. Based on the results, it is concluded that the domain-driven design can be used to obtain the optimal service granularity of microservices.

Keywords: Microservices, Size Estimation, Cloud Computing, Domain-driven Design, COSMIC Function Point.

ÖZ

MİKROSERVİSLERİN OPTİMAL BOYUTUNU HESAPLAMA

Vural, Hülya
Doktora, Yazılım Mühendisliği Bölümü
Tez Yöneticisi: Doçent Dr. Murat Koyuncu

OCAK 2020, 92 sayfa

Günümüzde bulut bilişim giderek standart ortam olmaya başlamıştır. Yekpare yazılım yapıları artık bulut isterlerine cevap verememektedir. Sonuç olarak, bağlığı az ve çevikliği yüksek yazılım geliştirmeye olanak sağlayan Mikroservis Mimari öne çıkmıştır. Mikroservis Mimari geleneksel servis mimarilerine göre daha küçük servislerin kullanımını önerir. Mikroservis Mimarinin tanımlandığı günden bugüne servislerin olması gereken boyut tartışması devam etmektedir.

Bu tezde hangi ölçütlerin en elverişli servis büyüklüğünü bulmada yardımcı olduğu araştılmıştır. Bu tez kapsamında Alan Güdümlü Tasarım kullanmış iki örnek belirlenerek her bir örneğin daha küçük servisli ve daha büyük servisli opsiyonları üretilmiştir. Sonuçta büyük servisli opsiyon, orijinal opsiyon ve küçük servisli opsiyonlar karşılaştırılmıştır. Karşılaştırmada götürgen bağlılık, getirgen bağlılık, kararsızlık, ilişkisel tutarlılık ve COSMIC fonksiyon noktaları ölçümlenmiştir. Sonuçlar götürgen bağlılık, getirgen bağlılık ve ilişkisel tutarlılık ölçümlemenin etken boyutu bulmaya daha uygun olduğunu göstermiştir. Sonuç değerlere bakarak Bölge Tabanlı Tasarımın etken boyuta ulaştırdığı gözlemlenmiştir.

Anahtar Kelimeler: Mikroservisler, Boyut Ölçümleme, Bulut Bilişim, Alan Güdümlü Tasarım, COSMIC Fonksiyon Noktaları

To the meaning of my life – Precious Lara

ACKNOWLEDGMENTS

I would like to express my thanks to my supervisor, Murat Koyuncu, for helping me publish better work.

I shall also thank to Ferda Nur Alpaslan and Çiğdem Turhan for their support during my Ph.D. journey.

Furthermore, I thank the experienced members of the faculty staff Yasemin Kurt, Ayşegül Özeke, Çağatay Öztürk and Özgür Soğukpınar Sidar for helping me out with the last minute accommodations.

In addition, I want to thank HAVELSAN for so called support which rather turned into a life lesson that toughened me up.

Finally, I want to thank my sister Hüsniye Vural Okur for unconditional support and helping me become who I am. My parents Yadigar Vural and Şehali Vural for their understanding and unconditional love. I also want to thank Lara for being my lighthouse and teaching me what really matters in life.

TABLE OF CONTENTS

ABSTRACT iii

ÖZ iv

DEDICATION v

ACKNOWLEDGMENTS vi

TABLE OF CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xii

LIST OF ABBREVIATIONS xiv

CHAPTER

1. INTRODUCTION 1

1.1 Statement of the Problem 2

1.2 Scope and Outline of the Thesis 6

2. BACKGROUND INFORMATION 9

2.1 Microservices Architecture 9

2.2 Size Estimation Methods 11

2.3 COSMIC Function Points 13

2.4 Domain-driven Design 14

3. LITERATURE SURVEY 17

3.1 Systematic Literature Mapping on Microservices 18

3.1.1 Mapping Method 18

3.1.1.1 Search Sampling 18

3.1.1.2 Search Iteration 19

3.1.2 Screening the Papers 19

3.1.3 Keywording 19

3.1.3.1 Service Models in Cloud Computing 20

3.1.3.2 Operational Areas 20

3.1.3.3 Research Types 21

3.1.3.4 Emerging Standards and Tools 21

3.1.4 Data Extraction and Mapping 22

3.2 Studies on Coupling and Cohesion Calculations of Microservices 27

3.2.1 Microservice Coupling Search 28

3.2.2 Microservice Cohesion Search 30

3.2.3 Search Results 31

4. METHOD 33

4.1 Chosen Example Applications 33

4.1.1 Original Cargo Application 33

4.1.2 Original eShopOnContainers Application 35

4.2 Generated Applications 39

4.2.1 Generated Cargo Applications 39

4.2.1.1 More Granular Cargo Application 39

4.2.1.2 Less Granular Cargo Application 40

4.2.2 Generated eShopOnContainers Applications 40

4.2.2.1 More Granular eShopOnContainers Application 41

4.2.2.2 Less Granular eShopOnContainers Application 42

4.3 Measurement Methods 43

4.4 The Overall Method for Carrying out the Case Studies 44

5. MICROSERVICES MEASUREMENTS 47

5.1 COSMIC Function Point Measurements 47

5.1.1 CFP Measurements for the Cargo Case Study 47

5.1.1.1 CFP Measurements for the Original Cargo Application 48

5.1.1.2 CFP Measurements for the MGM Cargo Application 52

5.1.1.3 CFP Measurements for the LGM Cargo Application 55

5.1.1.4 CFP Summary of CFP Measurements for Cargo Case Study. 57

5.1.2 CFP Measurements for the eShop Case Study 58

5.1.2.1 CFP Measurements for eShopOriginal 59

5.1.2.2 CFP Measurements for eShopSmall (MGM) 67

5.1.2.3 CFP Measurements for eShopBig (LGM) 71

5.1.2.4 Summary of CFP Measurements for eShop Case Study 73

5.2 Coupling Measurements 73

5.3 Cohesion Measurements 75
5.4 Analysis of Measurements 76
6. CONCLUSION 81
REFERENCES.. 84

LIST OF TABLES

TABLES

Table 3.1 Research types 21

Table 3.2 Mapping results 32

Table 5.1 New cargo (1) CFP calculation for original cargo application 48

Table 5.2 New cargo (2) CFP calculation for original cargo application 49

Table 5.3 Handle cargo (1) CFP calculation for original cargo application 49

Table 5.4 Handle cargo (2) CFP calculation for original cargo application 50

Table 5.5 Handle cargo (3) CFP calculation for original cargo application 51

Table 5.6 Track cargo (1) CFP calculation for original cargo application 51

Table 5.7 Track cargo (2) CFP calculation for original cargo application 52

Table 5.8 Total CFP of original cargo application 52

Table 5.9 Handle cargo (1) CFP calculation for MGM cargo application 53

Table 5.10 Handle cargo (2) CFP calculation for MGM cargo application 53

Table 5.11 Handle cargo (3) CFP calculation for MGM cargo application 54

Table 5.12 Total CFP of MGM cargo application 54

Table 5.13 New cargo (1) CFP calculation for LGM cargo application 55

Table 5.14 New cargo (2) CFP calculation for LGM cargo application 56

Table 5.15 Handle cargo (3) CFP calculation for LGM cargo application 56

Table 5.16 Track cargo (2) CFP calculation for LGM cargo application 57

Table 5.17 Total CFP of LGM cargo application 57

Table 5.18 eShopOriginal “Login” CFP calculation 59

Table 5.19 eShopOriginal “List items for sale” CFP calculation 60

Table 5.20 eShopOriginal “Filter on brand” CFP calculation 60

Table 5.21 eShopOriginal “Filter on type” CFP calculation 61

Table 5.22 eShopOriginal “Add item to a basket” CFP calculation 62

Table 5.23 eShopOriginal “List items in the basket” CFP calculation 62

Table 5.24 eShopOriginal “Update item in basket” CFP calculation 63

Table 5.25 eShopOriginal “Delete item from basket” CFP calculation 63

Table 5.26 eShopOriginal “Checkout” CFP calculation 64

Table 5.27 eShopOriginal “Get buyer profile” CFP calculation............................. 64
Table 5.28 eShopOriginal “Update buyer profile” CFP calculation........................ 65
Table 5.29 eShopOriginal “Update buyer password” CFP calculation 65
Table 5.30 eShopOriginal “List orders” CFP calculation....................................... 66
Table 5.31 eShopOriginal “Get order item detail” CFP calculation........................ 66
Table 5.32 eShopOriginal “Logout” CFP calculation... 66
Table 5.33 eShopOriginal “Update marketing” CFP calculation 67
Table 5.34 Total CFP of eShopOriginal 67
Table 5.35 eShopSmall “Login” CFP calculation... 68
Table 5.36 eShopSmall “Checkout” CFP calculation... 69
Table 5.37 eShopSmall “Get buyer profile” CFP calculation................................. 69
Table 5.38 eShopSmall “Update buyer profile” CFP calculation............................ 70
Table 5.39 eShopSmall “Update buyer password” CFP calculation 70
Table 5.40 Total CFP of eShopSmall 71
Table 5.41 eShopBig “Checkout” CFP calculation 72
Table 5.42 Total CFP of eShopBig... 72
Table 5.43 Measurements for cargo case study 76
Table 5.44 Normalized measurements for cargo case study................................... 77
Table 5.45 AHP results for cargo case study 77
Table 5.46 Measurements for eShopOnContainers case study................................ 78
Table 5.47 Normalized measurements for eShopOnContainers case study 78
Table 5.48 AHP results for eShop case study 78

LIST OF FIGURES

FIGURES

Figure 1.1 Monolith vs. Microservices 5
Figure 2.1 Four different types of data movements in COSMIC function point 14
Figure 3.1 Process steps and outcomes 18
Figure 3.2 Research types.. 22
Figure 3.3 Count of papers per service type 22
Figure 3.4 Service types versus research types 23
Figure 3.5 Operational areas... 24
Figure 3.6 Empirical results in research 24
Figure 3.7 Operational area vs service type 25
Figure 3.8 Implementation of solutions... 25
Figure 3.9 Emerging standards 26
Figure 3.10 Common tools used for implementing microservices 26
Figure 3.11 Number of microservices papers over time (2017) 27
Figure 3.12 Number of microservices papers over time (2019) 27
Figure 4.1 The UML class diagram for the Cargo application 34
Figure 4.2 eShop application screenshot... 35
Figure 4.3 eShopOnContainers type dependency diagram 36
Figure 4.4 Basket entities 36
Figure 4.5 Catalog entities 37
Figure 4.6 Identity entities 37
Figure 4.7 Location entities... 37
Figure 4.8 Marketing entities 38
Figure 4.9 Ordering aggregates.. 38
Figure 4.10 The UML class diagram for the MGM Cargo example 39
Figure 4.11 The UML class diagram for the LGM Cargo example.................... 40
Figure 4.12 Four main services in eShopOnContainers.................................. 41
Figure 4.13 Five different services in eShopSmall 42

Figure 4.14 Three different services in eShopBig 43
Figure 4.15 The process followed for sampling and measurements........................ 45
Figure 5.1 CFP results for cargo case study 58
Figure 5.2 CFP results for eShop case study 73
Figure 5.3 Coupling values of cargo case study 74
Figure 5.4 Coupling values of eShopOnContainers case study............................... 74
Figure 5.5 Cohesion values of cargo case study... 75
Figure 5.6 Cohesion values of eShopOnContainers case study............................... 76

LIST OF ABBREVIATIONS

AHP	Analytic Hierarchy Process
AI	Artificial Intelligence
API	Application Programming Interface
App.	Application
BC	Bounded Context
CFP	COSMIC Function Points
COCOMO	Constructive Cost Model
COSMIC	Common Software Measurement International Consortium
CPM	Counting Practices Manual
CPU	Central Processing Unit
DB	Database
DDD	Domain-Driven Design
Desc.	Description
DM	Data Movement
FFP	Full Function Points
FiSMA	Finnish Software Measurement Association
FSM	Functional Size Measurement
FUR	Functional User Requirements
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission
IFPUG	International Function Point User Group
ISO	International Standards Organization
JTC	Joint Technical Committee
LGM	Less Granular Microservice application
LLOC	Logical Line of Code
LOC	Line of Code
MA	Microservices Architecture
MCDM	Multi-objective Decision Making

MGM	More Granular Microservice application
MIT	Massachusetts Institute of Technology
REST	Representational State Transfer
RQ	Research Question
SLA	Service Level Agreement
SMRQ	Systematic Mapping Research Question
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service-Oriented computing
SPR	Software Productivity Research

CHAPTER 1

INTRODUCTION

Cloud computing is an emerging paradigm [1]. It changes the way we do business as well as engineering. In a world where being fast and delivering new features to the end users everyday are much more important than what it has been a decade ago [2]. The services are expected to have at least 99.99 uptime [3]. Such Service Level Agreements (SLA) require the way we do software engineering to evolve. Monitoring and alerting upfront became critical in any service offering. The developer teams are expected to be autonomous so that they can make their own decisions and move quicker without waiting for integrated service updates. This decoupled fast pace change also led to a new architectural style called Microservices Architecture [4].

Microservices are assumed to do one thing and do it well [5]. The main purpose of the microservices is to divide the business behavior into small services which can run independent of each other [1]. The microservices are developed, deployed and maintained separately. This allows the teams to be autonomous where they can decide on the technology to use, which best addresses the current needs of the business behavior. The language and the database might be different from one microservice to another. They do not share data between each other, instead they use Representational State Transfer (REST) protocol to communicate to each other. Some took it to the extent of "Share Nothing Principle" even the common utility libraries [6].

So what are the reasons for moving from Software Oriented Architecture (SOA) to microservices? The most important benefits of using microservices are agility, autonomy, scalability, resilience and easy continuous deployment [7]. The autonomous and less sharing (de-coupled) services can deliver updates to the end user faster. Which means, time to market is reduced. The resilience is a must where a service is promising a certain SLA [8]. One outage in a less important service should not affect the critical behavior of the overall system. As a result, having decoupled services bring flexibility.

1.1. Statement of the Problem

Microservices are becoming the new architectural style for the cloud applications due to their advantages such as being easier to maintain, more adaptable, complying with the needs of continuous integration where the system down time is almost zero [9]. On the other hand, microservices also come with extra cost such as distributed communication. Microservices are self-contained processes talking over the network. Distributed communication introduces extra complexity and possible performance issues.

When the size of the software grows, the interdependency and coupling generally increases as well [10]. On the contrary, when a certain service is divided into multiple microservices, serialization and deserialization costs, remote network calls and the possibility of possible failures increase.

In order to gain the most benefit out of a microservice, it is important to find the optimal size. Architecting different sizes of microservices would affect to following factors:

- Network traffic: When the size of the microservices are bigger, the communication between different services will decrease. As a result, the network traffic will be less. However, when the size of the services is smaller, some of the communication which was carried out within the service will be carried out across services. As a result, the chattiness of the services will increase, which leads to more network traffic. In order to reduce the network traffic, the services are expected to be less granular.

- Performance: When the communication is carried out within the service, the Application Programming Interface (API) calls are within the service – mostly in memory [11]. When the microservice is divided into smaller pieces, the communication will be over HyperText Transfer Protocol (HTTP) mostly using REST [12]. This adds the performance cost of serialization or de-serialization of data. On top of serialization costs, there is also network latency for each API call. In order to improve the performance, less granular microservices should be selected.

- Complexity: By splitting one large thing into small bits you end up having more bits [13]. Each service is a separate deployment unit, which has to be released, tested and monitored [11]. The more services we have, the more configurations we will have to maintain. Given that services are communicating to each other over the network, each service has its own controls to handle possible failure scenarios (circuit breaker pattern, retry logic etc.). When the services are more granular, the overall complexity of the system increases. On the other hand, the logic inside the service becomes much simpler.
- Testing: Through microservices, end-to-end testing is usually more challenging as testing all microservices together requires a more sophisticated setup and testing approach as there are more moving parts, data sources and communication involved [11]. When the services are more granular, end to end testing becomes harder. On the other hand, unit testing becomes easier as the logic within a service becomes simpler.
- Coupling: The services do not necessarily share the hardware resources. In most cases the services are deployed as Docker instances, which have specific resources allocated to them. Services are not sharing the database/schema. As a result, one schema change for a service would not affect another. The database can be different for each service as well. The message routing between the services is carried out through REST calls. Unlike Simple Object Access Protocol (SOAP), the contract for the REST calls can be consumed by an online documentation (e.g. Swagger). The versions of the contracts can also be negotiated between the services. In order to reduce the coupling, some may decide to have more granular services.
- Resiliency: When the application loads its responsibility on a few services, one small failure on such a critical service may result in down time. It is easier to isolate a failure within a service [11]. As a result, having more granular services lead to fault isolation and increased fault tolerance [14]. In order to reduce the cost of failure, more granular services should be selected [13].
- Scalability: When the application has multiple modules packed into a single service, the scalability of just one module is out of question. The whole service

has to be replicated as a new instance [15]. As a result, when the services are less granular, it is easier to scale only the module which gets more load.

- Transactions: When a transaction has to be managed across two or more services, it is called distributed transaction. The distributed transactions make the system more complex to manage [11]. When a business action cannot be carried out through eventual consistency of data across multiple services, it could be more beneficial to combine those services into one. As a result, in order to avoid the complexity of distributed transactions, less granular services can be selected.
- Technology or Vendor Lock-in: In microservices architecture, different services can use different programming languages, different databases, and different libraries. As a result, the technology or vendor lock-in can be scoped to less critical services. Such vendor locked-in services will be isolated in a sense.
- Autonomy: When the services are smaller, the possible impact of a regression is smaller as well. This leads to faster development cycles [11]. The more granular services can be managed by smaller teams. When the smaller teams are more autonomous, new features can be delivered faster.

Due to the reasons mentioned above, there are some aspects which are better when the services are less granular. On the other hand, there are some aspects, which would benefit from more granular services. Finding the optimum size for a service would mean finding the sweet spot that balances all different aspects to consider.

There is a continuous debate around what the actual size of a microservice should be. Newman and Fowler suggested “The largest sizes reported follow Amazon's notion of the Two Pizza Team (i.e. the whole team can be fed by two pizzas), meaning no more than a dozen people.” [4]. There have been more size definitions; such as no more than 300 lines of code, being able to rewrite the microservice in 6 weeks [16], between 10 to 100 lines of code. It takes one week to write [17], as small as possible, etc. Some suggest that the size cannot be an indicator of a microservice [18]. It should be the context and boundary of responsibilities. For identifying the right context of a microservice, domain-driven design was proposed as well [19].

There has been research around extracting microservices from monolith legacy applications [5] [20] [21]. Considerable amount of such research focuses on measuring static code analysis metrics such as coupling and cohesion [5] [21]. The optimum result was targeted through low coupling and high cohesion values. In order to measure the coupling values, afferent and efferent coupling can be used. The cohesion can be measured through relational cohesion.

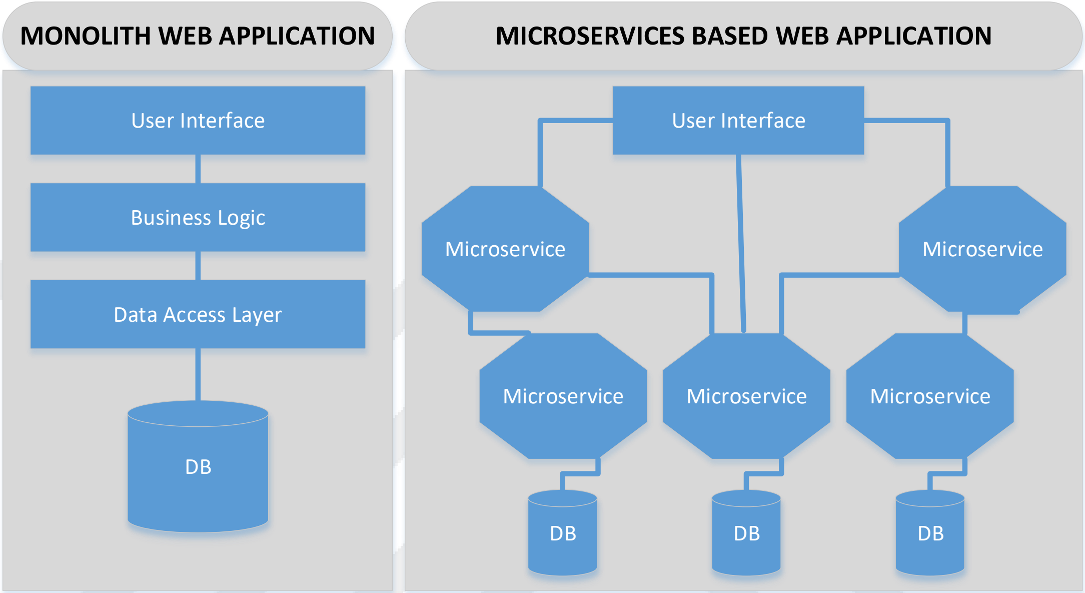

Figure 1.1 Monolith vs. Microservices Based Web Application

In order to measure the size of an application, there are many different methods. Out of those methods, one of the most commonly used one is the Common Software Measurement International Consortium (COSMIC) function points [22]. In the COSMIC function points, the layering is quite important. The studies around the validity of COSMIC function points have been carried out for web projects (3 tier architectures). The microservices are the new architectural style for cloud based web applications. The layering of the microservices is different from typical 3 tier web applications as seen in Figure 1.1. Currently, there is limited study around whether the COSMIC function points would still be applicable to microservices. Web of Science topic search for “COSMIC microservice” keywords returned only one result [23]. That particular result is a conference paper generated during the early phases of this thesis study. The same search on Google Scholar (excluding citations) returned a total of 6 results. Out of those 6 results, only 2 of them measured the size of a microservice using COSMIC function points [23] [24].

In order to measure the size of an application, COSMIC method can be used. COSMIC method is more suitable for measuring the size of layered design such as Service Oriented Architecture (SOA), because the layers are taken into account rather than the user facing process [25]. The problem around finding the right size for microservices still exist. There has not been enough research around measuring the size of microservice using COSMIC function points.

Maybe the real problem is not around finding the right size but mostly around finding the right granularity of the service by applying certain concepts. Does the size of the microservice really matter, or is the context more important than the size? Can domain-driven design lead to optimal granularity of microservices?

1.2. Scope and Outline of the Thesis

As part of this thesis, we try to get an answer for the granularity problem of microservices. The optimal granularity would depend on many factors such as performance, autonomous teams, resilience, etc. Depending on the needs, different measurements can be targeted. There has been previous research focusing on identifying the granularity of microservices based on static code analysis measurements such as coupling and cohesion. The current study scopes its measurements to coupling, cohesion and COSMIC function points, and aggregates them to find an optimal spot that balances these measurement results.

The problem around finding the optimal result will be addressed through applying Analytic Hierarchy Process (AHP). While applying AHP, different weights could be given to different factors measured. Let’s say someone values delivering features at a face pace over maintainability. In order to deliver a feature in a short period of time, that person might look for lower COSMIC function points. Because the higher the COSMIC function points, higher the time of development. When coupling values are low, maintenance is expected to be easier. So in this particular example, given that the person values fast feature delivery over maintainability, that person would give higher weight to COSMIC function points when compared to coupling values. In our case, we do not have any preference. As a result, while applying AHP, all the criteria are calculated as equal.

In the current study, we aim to answer the following research questions (RQs):

- RQ1: Is Microservices Architecture a trending topic?
- RQ2: Is there a relation between the granularity and the coupling of microservices?
- RQ3: Is there a relation between the granularity and the cohesion of microservices?
- RQ4: Can optimum granularity of a microservice be calculated by minimum coupling value and the largest cohesion value?
- RQ5: Is COSMIC function a good fit measuring the size of microservices?
- RQ6: Does domain driven design help identifying the optimum microservice?
- RQ7: Does microservice size measurement help finding the optimum size?

In order to answer the research questions, two different examples were studied on. The first example is a UML class diagram for a cargo application which was generated by applying domain-driven design concepts [26]. The second example is an open source online shopping application constructed by using domain-driven design [27]. Each example was originally generated by applying domain-driven design concepts. As part of this thesis, both examples were modified to generate more granular and less granular services. In the end, the original example, the more granular and the less granular examples are all measured for the following metrics:

- Afferent coupling
- Efferent coupling
- Relational cohesion
- COSMIC function points

All four metrics are collected for the resulting 6 different applications. The resulting measurements are studied with the aim of answering the research questions. The current study contributes to the literature through getting an answer for the ongoing debate around the size of a microservice.

The rest of the thesis is organized as follows: Chapter 2 gives background information on the terminology used in this thesis, Chapter 3 covers the literature survey related to microservices and boundaries, Chapter 4 explains the methodology used in this research, then the measurement results and discussions are shared in Chapter 5, finally we conclude with Chapter 6.

CHAPTER 2

BACKGROUND INFORMATION

In this thesis, we are focusing on finding the right size/context of microservices. First, we would like to give some background information on key terminology. We explain the microservice architecture style and compare it to service-oriented architecture. We also cover the history of microservices. Then, we explain different size estimation methods and include extra details around how COSMIC function calculation works. Lastly, we cover domain-driven design and its principles.

2.1. Microservices Architecture

Service-oriented architecture (SOA) has emerged as a means of developing distributed systems where the components are stand-alone services [28]. Services are basic units which are developed independently and made accessible over the Internet. Standard internet protocols are used for service communication among different computers. SOA provides many advantages to develop easy and economic distributed software systems and, therefore, it is the leading technology for interoperability on today's internet world. Service-oriented software engineering defines evolution of existing software engineering approaches to develop dependable and reusable services considering the requirements and characteristics of this technology [28]. Service-oriented computing (SOC) is the paradigm that utilizes services as the fundamental elements for developing applications. Therefore, service-oriented software engineering aims at designing and developing service-based applications consonant with SOC paradigm and SOA principles using software engineering methodologies.

After the popularity of cloud computing in recent years, new trends in software engineering have emerged, such as going to the market with minimal viable product and making small development teams autonomous. The architectural styles have also evolved based on the cloud environment needs. One of those new architectural styles is microservices. The aim of the microservices is to divide the business behavior into small services which can run independent of each other. As mentioned by Fowler, “While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data” [4]. Another definition for microservices is “Microservices are small, autonomous services that work together” [29].

The characteristics of the microservices are listed as follows [4]:

- Smart end points rather than smart pipes: Unlike SOA, use simple queue structures.
- Componentization around services: A service can be updated without affecting other services.
- Organized around business capabilities: During design time, the modules are identified based on business capabilities.
- Products not projects: The development team who owns the responsibility of the service, also owns the responsibility of running the service in production as well (including deployment responsibility).
- Decentralized governance: Instead of having a master decide what to do, the services talk to each other to carry out the operations.
- Decentralized data management: One service should not use the data of another service directly with database (DB) calls. Instead, it should call the responsible service’s API to get to data.
- Infrastructure automation: Automating all development and operation cycles.
- Design for failure: Think about failure cases in advance and design based on that.
- Evolutionary design: The design of the service is an ever-evolving process.

The microservices are developed, deployed and maintained separately. This allows the teams to be autonomous where they can decide on the technology to use which best addresses the current needs of the business behavior. The language and the database might be different from one microservice to another. They do not share data between each other, instead they use Representational State Transfer (REST) protocol to communicate with each other. The most important benefits of using microservices are agility, autonomy, scalability, resilience and easy continuous deployment.

2.2. Size Estimation Methods

There are many different techniques to measure the size of a software where some are more error prone and some follow certain standards. One of the most known methods for sizing a software is measuring the Line of Code (LOC) [30]. Even though it is one of the easiest methods, it varies a lot depending on the number of lines with comments only or style of the code. As a result, measuring the Logical Line of Code (LLOC) was proposed [31]. Even though it is more accurate than the LOC, it still causes ambiguity given that experienced developers could be developing the same functionality with far less code. The dead code would still be counted even in LLOC. Auto generated code cannot be counted for neither in LOC nor LLOC.

In mid 1970s, rather than measuring the lines of code, the idea of measuring the business functionality emerged. Alan J. Albert set the first version of function point measurements [32]. Since then it evolved and International Function Point User Group (IFPUG) was established. IFPUG published an International Standards Organization (ISO) compliant version of the function points.

ISO/IEC/JTC1/SC7 Standard 14143-1(2004) sets out the following definitions [25]:

- “Functional Size: A size of the software derived by quantifying the Functional User Requirements.”
- “Functional Size Measurement (FSM): The process of measuring Functional Size.”
- “FSM Method: A specific implementation of FSM defined by a set of rules, which conforms to the mandatory features of this part of ISO/IEC 14143.”

Function Points are seen as a more reliable way of measuring the size of the code when compared to LOC [33]. There have been variations of function points as well, such as Mark II, 3D Function Points, Full Function Points, DeMarco function points, Object points, Software Productivity Research (SPR) function points [34]. Out of many different size estimation models, only a few comply with the latest ISO standards which also happen to be the most common ones as well: Mark II, Nesma, Finnish Software Measurement Association(FiSMA) 1.1, IFPUG and COSMIC.

IFPUG Counting Practices Manual (CPM) is compliant with ISO/IEC 20926: 2009. When measuring function points according to IFPUG guidelines, the programming language does not affect the measurement. It can measure activities related to software development such as documentation, defects in requirements and design [34]. However, the measurement requires certified professionals and it is time consuming. Even though there are tools to automatically measure it, the reliability of the tools is not yet proved. The non-functional attributes such as quality or Central Processing Unit (CPU) heavy operations (algorithms, calculations etc.) are not meant to be measured by IFPUG CPM.

Nesma is compliant with ISO/IEC 24570:2005. It measures the data and the data flow unlike Mark II and COSMIC methods. Nesma puts a limit after a certain threshold just like IFPUG. It originated due to the fact that IFPUG did not clear certain areas on its guideline. However, after the publication of IFPUG CPM 4.2, IFPUG and Nesma converged to the same standard [35].

FiSMA is compliant with ISO/IEC 29881:2008. It measures the data and the data flow unlike Mark II and COSMIC methods. FiSMA puts a limit after a certain threshold. It is mainly service oriented where it requires all the services in the software to be identified. Besides publishing the generic measurement method, the FiSMA website offers specialized guides for special cases such as Multi-Layer Systems and Graphical User Interface. Finnish Software Measurement Association still updates the guidelines.

Mark II is compliant with ISO/IEC 20968:2002. Mark II is more granular when compared to IFPUG. Instead of measuring the groups of elements, all the entities and data elements are measured. IFPUG limits the size of a component after a certain size is reached. As in Mark II, there is no threshold. It is an accepted standard in UK. Mark II is good for DB heavy programs which use relational databases.

COSMIC Full Function Points (FFP) is compliant with ISO/IEC 19761:2009. Rather than measuring the data itself, it measures the movement of the data. Just like IFPUG CPM, it fails to measure the nonfunctional requirements such as quality and CPU heavy operations. Unlike IFPUG CPM, the COSMIC method gives specific guidelines on multi layered systems. Even though IFPUG CPM is based on measuring from the external users' point of view, the COSMIC method bases its measurements on functional users' point of view. COSMIC method does not put a threshold on the components measured. COSMIC method is more suitable for Service Oriented Architecture (SOA), because the layers are taken into account rather than the user facing process [25].

The Object Points are used by Constructive Cost Model (COCOMO). Unlike the name suggests, the method does not refer to the objects in object oriented programming. The objects in this model are screens, reports and modules. Several other not widely used methods are De Marco bang metrics, 3D functional points [36] and Feature Points [37].

As mentioned above almost all of the function point based metrics are lacking ways to measure algorithmic complexity (CPU intensive operations). In order to fill that gap, weighted micro function point method was introduced. This method is more detailed and more accurate when compared to traditional function point analysis. However, it requires a tool to measure all the necessary elements. The biggest downside of the current method is that it can only be done on the existing software. In order to measure the nonfunctional requirements, IFPUG also offered a method called SNAP. It is seen as complementary to IFPUG Function Point Analysis.

2.3. COSMIC Function Points

Common Software Measurement International Consortium (COSMIC) decided to form an accurate and compliant measurement method for software sizing – COSMIC function points.

The COSMIC method focuses on data movements between different layers. One of the benefits of COSMIC method is that it can give a result in the planning phase based on the functional user requirements (FUR). The four main data group types are as shown in Figure 2.1 [38]:

- Entry (E): An event triggers the beginning of a function process. When a data group is crossing a layer as a result of an event, then it is marked as an entry.

- Exit (X): When the result of an entry is returned, it is marked as an Exit. This shall also include the error cases if a message or notification is shown to the user. Unhandled/exceptional cases of errors do not count as an exit.
- Read (R): When data is read from a storage, it is marked as a read. The storage can be a database or cache. However, access to items kept in memory in that layer are not counted as a read.
- Write (W): When data is written to a storage, it is marked as a write. The storage can be a database or cache. However, storing an item in memory within the same layer does not count as a write.

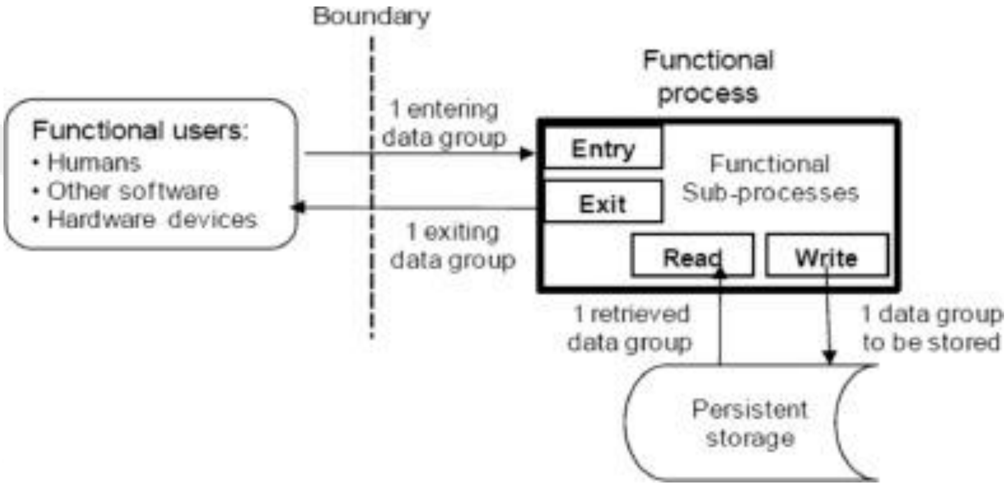

Figure 2.1 Four different types of data movements in COSMIC function point analysis [39]

2.4. Domain-driven Design

Domain-driven design (DDD) is an approach to software development for complex needs by connecting the implementation to an evolving model [40]. According to Evans [41], the domain-driven design evolved after experiencing considerable amount of discrepancies between the domain experts, the analysts, the designers and the developers. It became obvious that there was a need for constructing a common language. This is called “Ubiquitous Language” in domain-driven design.

The main aspect in domain-driven design is putting the domain knowledge in the center of the design. There are 4 layers in domain-driven design:

- User Interface: The presentation layer of the design.

- Application Layer: This layer's main goal is to provide the communication between the user layer and the domain layer. There is no logic in this level.
- Domain Layer: The heart of the design. All the domain related functionality resides in this layer.
- Infrastructure Layer: This is the layer for serving all the other layers with infrastructure needs, the common functionality which is not part of the domain knowledge.

The building blocks for the domain-driven design are [42]:

- Entity: Objects which have an identity are called entities.
- Value Objects: When the object doesn't have an identity and it is useful only for carrying certain attributes or carrying out a calculation, then it is called a value object. The value objects are immutable.
- Domain Event: Unlike the other building blocks, the domain events were introduced after the original domain-driven design book [41] was published. Domain event represents an event which the domain expert (end user) cares about.
- Aggregate: When certain objects are related to each other in a sense that one cannot exist without the root, then those are combined into one identity called aggregate. The main purpose of the aggregates is insulating the domain knowledge into one object so that any external method does not change it in a wrong way.
- Service: When an operation does not belong to only one entity or aggregate, then it is best to serve that functionality as a service.
- Repository: The infrastructure is responsible for carrying out the persistency operations (e.g. read from a database). The repository pattern shields the details of persistence in such a way that the aggregates can be properly encapsulated.
- Factory: There are cases when a construction of an object is so complicated that the caller may get tangled in that coupled code. In order to avoid such cases, the factories should be leveraged. Using the factory pattern frees the caller from dealing with the details of the object to be used.

The domain-driven design is best suited for complex systems with many different domain knowledge areas. It adds extra cost for the sake of introducing well defined domain boundaries [41]. These different domain components are called Bounded Context.

The main advantages of domain-driven design are maintainability, the autonomy of the team, ease of continuous integration and the use of different persistence tools. However, it comes with a price of complex design and costlier initial development.

In the cloud business where the changes are on a continuous basis, the initial development time becomes less of a concern when compared to maintainability. As a result, certain experts advise the adoption of domain-driven design along with microservices in cloud systems [43].

CHAPTER 3

LITERATURE SURVEY

The industry moved from monolith applications to Service-oriented architecture (SOA) more than a decade ago [44]. SOA has emerged as a means of developing distributed systems where the components are stand-alone services [28]. In order to achieve even more modular and independently scalable components, the Microservices Architecture (MSA) was proposed [4]. Each microservice is supposed to carry out one business capability [5]. Each microservice is loosely coupled, highly cohesive component which is run separately.

Even though microservices were first mentioned at [45] in 2010, the definition of the microservice given in that study does not totally map to the current microservice definition in literature. The study carried out in 2010 [45] defines microservices as light services using REST. It does not mention most of the characteristics listed in [4].

The research papers about microservices only dates back to 2014. There has been a systematic mapping carried out on microservices in 2016 [46]. In that study, the research questions are around the architectural diagrams used for microservices' representation, the quality attributes and the challenges. However, the emerging standards and de facto tools are not mentioned.

At the early stages of the current thesis study, a systematic literature mapping was carried out on microservices. In section 3.1, the results of that mapping study are covered. Recently, a literature review was held on coupling and cohesion of microservices. The results can be found in Section 3.2.

3.1. Systematic Literature Mapping on Microservices

3.1.1. Mapping Method

The mapping study is conducted a systematic mapping as defined in [47] with one slight modification (see Figure 3.1). The modification is that, we carry out the keywording according to the whole paper contents instead of just the abstracts. The reason for the modification to the original process is to enhance the classification criteria through adding new areas.

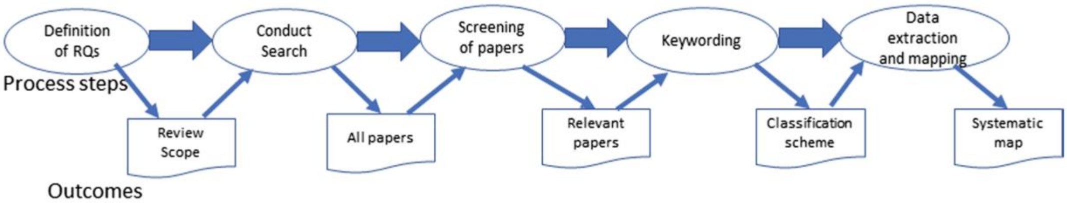

Figure 3.1 Process steps and outcomes

3.1.1.1. Search Sampling

The search is conducted using Web of Science (Thomson Reuters Web of Knowledge) [48], which includes the following online databases:

- ACM (Association for Computing Machinery) Digital Library
- CiteSeer
- Computer Source
- ebrary
- Human-Computer Interaction Bibliography
- IEEE Xplore
- INSPEC
- INSPEC Archive
- Nature
- Science
- Science & Technology Collection
- SciTech Connect

- Springer LINK

3.1.1.2. Search Iteration

The search is carried on with the following criteria:

- Keywords: microservice OR micro-service
- Research Area: Computer Science

The search iteration has returned 39 results [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [46] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [45].

3.1.2. Screening the Papers

The papers are evaluated according to the inclusion and exclusion criteria. The ones which do not meet the criteria are excluded. The inclusion criteria:

- All papers returned from the search criteria

Exclusion criteria:

- If the microservices is just mentioned in the research but the focus of the research is not directly on microservices.

Out of 39, 2 papers were excluded based on the exclusion criteria [66], [67]. As a result, 37 papers are included into the mapping process.

3.1.3. Keywording

As part of keywording four different categorization schemes are identified:

- Service models in cloud computing
- Operational areas
- Research types
- Emerging standards and tools.

3.1.3.1. Service Models in Cloud Computing

The service models in cloud computing are classified in three different types:

- Infrastructure as a Service (IaaS): the infrastructure is supplied as a service (e.g. virtual machine, hard disk, load balancer etc.).
- Platform as a Service (PaaS): The platform is supplied as a service (e.g. Azure SQL, Tomcat etc.).
- Software as a Service (SaaS): The software itself is supplied as a service (e.g. Office 365, Gmail etc.).

Even though the microservice architecture style is shaped considering cloud needs, the research papers returned as part of the search criteria do not necessarily use cloud. As a result, on premise installations (OnPrem) are also included in the service models.

3.1.3.2. Operational Areas

In [86] several different operational areas are called out for cloud:

- Accounting and billing
- SLA management (Service Level Agreement)
- Service/resource provisioning
- Capacity planning
- Configuration management
- Security and privacy assurance
- Fault management

Some of the research papers included in the current study are focusing on cloud whereas some are not. As a result, the operational areas were modified to fit the needs as follows:

- Cost comparison
- Availability/Resiliency
- Performance
- Security
- Test technique

- Functionality/Design
- Analytics/Monitoring
- Scalability
- Deployment

3.1.3.3. Research Types

In [87], 6 different research types are called out (See Table 3.1). The types are identified based on what is being proposed and how the research is being carried out.

Table 3.1. Research types

Class	Description
Validation research	Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the lab
Evaluation research	Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes identification of problems in industry
Solution proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation
Philosophical papers	These papers sketch a new way of looking at existing things by structuring the field inform of a taxonomy or conceptual framework
Opinion papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should have been done. They do not rely on related work and research methodologies
Experience papers	Experience papers explain what and how something has been done in practice. It has to be the personal experience of the author

3.1.3.4. Emerging Standards and Tools

The papers included in this systematic mapping study can be seen as a representation of the common tools used for microservices. Given that microservices is a new concept, the standards are not yet well formed. The systematic study aimed to give an answer on emerging standards for microservices.

3.1.4. Data Extraction and Mapping

The results obtained from mapping are converted into different graphs and they are given below. The papers are mapped to the research types as seen in Figure 3.2. The most widely used research type is Solution Proposal which is followed by Validation Research and Evaluation Research.

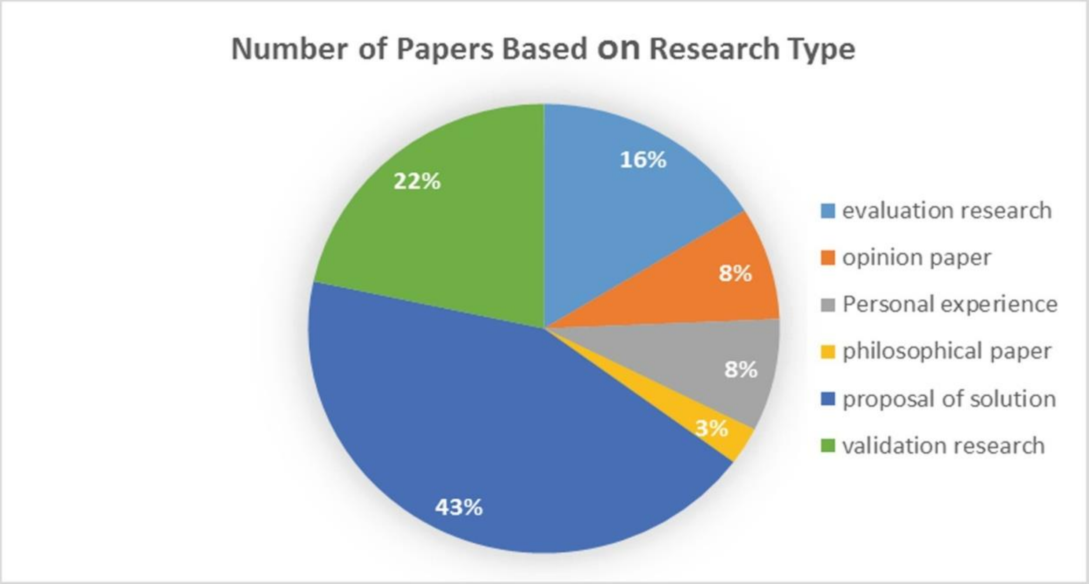

Figure 3.2 Research types

The papers are classified according to the service types as seen in Figure 3.3. Almost half of the papers did not explicitly mention the service type they were targeting (represented as NA in the figure). SaaS by far the most common service type being investigated. Also, some papers refer to more than one service type.

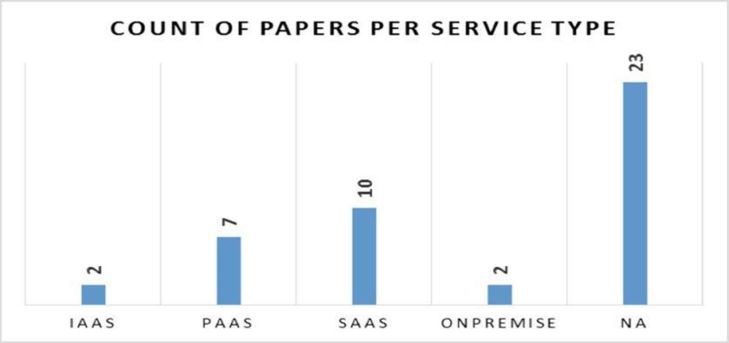

Figure 3.3 Count of papers per service type

The bubble chart in Figure 3.4 illustrates an analysis based on research types versus service types. The figure shows that there are only two studies on IaaS investigation regarding microservices. This is an expected outcome given that the microservices is a high level architectural style. On the other hand, there is only one official philosophical research papers on microservices. Most probably the reason is that the philosophical statement of microservices was laid out by Lewis and Fowler [4] on 2014. Mostly, the research is around Solution Proposal which do not explicitly call out the possible service types applicable for that solution.

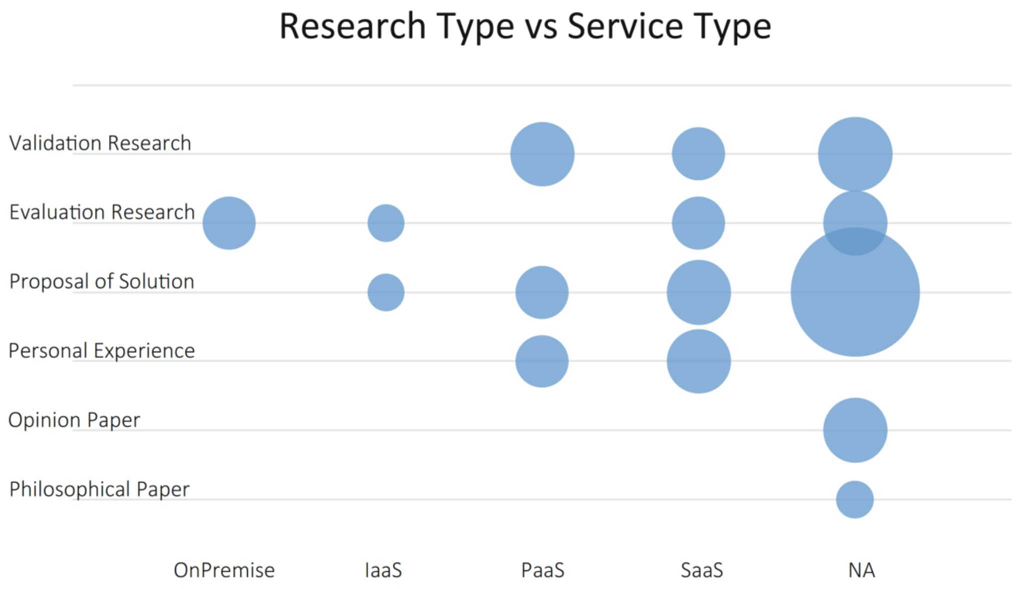

Figure 3.4 Service types versus research types

The papers were mapped to the operational areas and obtained results are shown in Figure 3.5. The main motives were around functionality followed by performance and test techniques. Given that the microservices paradigm was first mentioned around 2014 and official research papers started to show up in 2015, it is natural to expect the functionality be main concerns of research.

Keep in mind that the original systematic mapping was carried out at the early stages of this thesis. If the same systematic mapping was carried out in 2020, the results may differ.

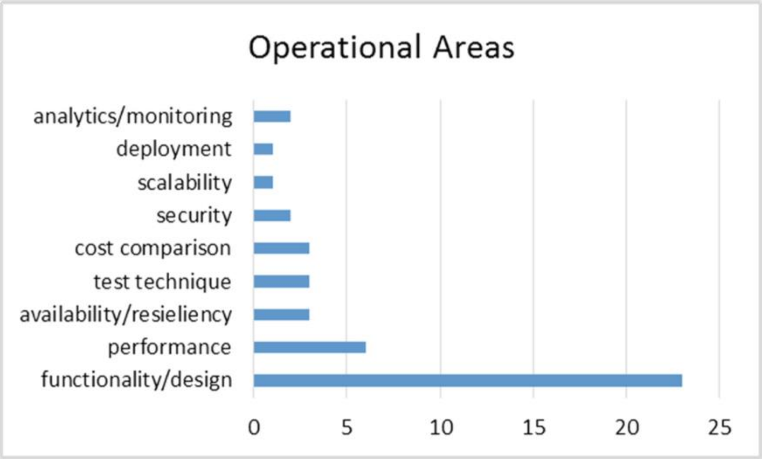

Figure 3.5 Operational areas

Figure 3.6 aims to answer if the study has empirical results or not. Our analysis showed that the empirical studies were smaller in amount. As this new architecture style gets more mature overtime, it is natural to expect more empirical studies compared to just proposals.

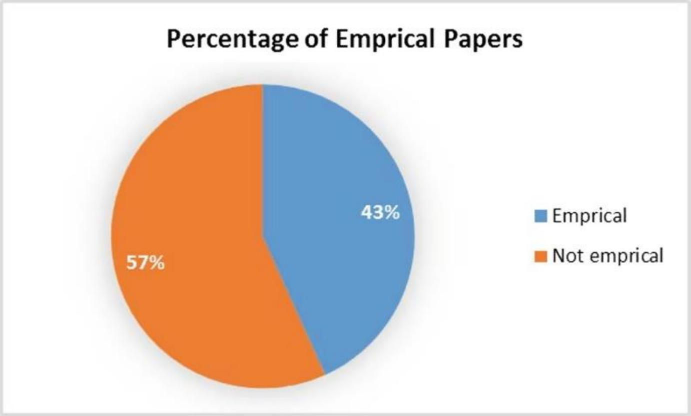

Figure 3.6 Empirical results in research

Figure 3.7 illustrates an analysis based on operational areas versus service types. The most remarkable point is that most of the studies focus on the functionality/design issues. The next topic is around performance. The scalability and deployment are not really mentioned much. Most probably the reason is that the Microservices Architecture is expected to solve those problems inherently.

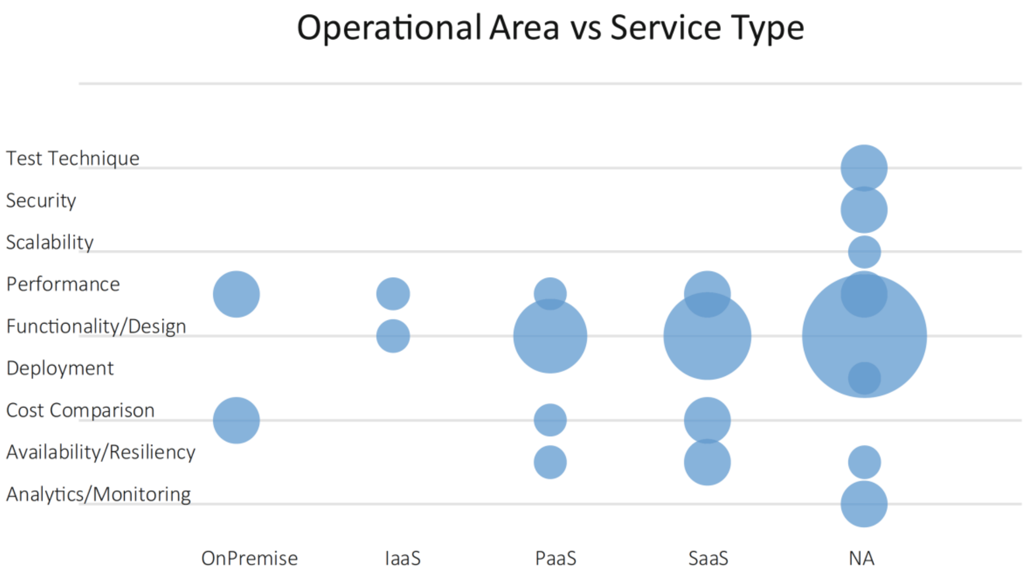

Figure 3.7 Operational area vs service type

Figure 3.8 shows if there is a new solution proposed and/or implemented. As seen in the figure, most of the research propose new solutions. Another noticeable point is that the implementation ratio of new solutions is higher than the implementation ratio of existing solutions.

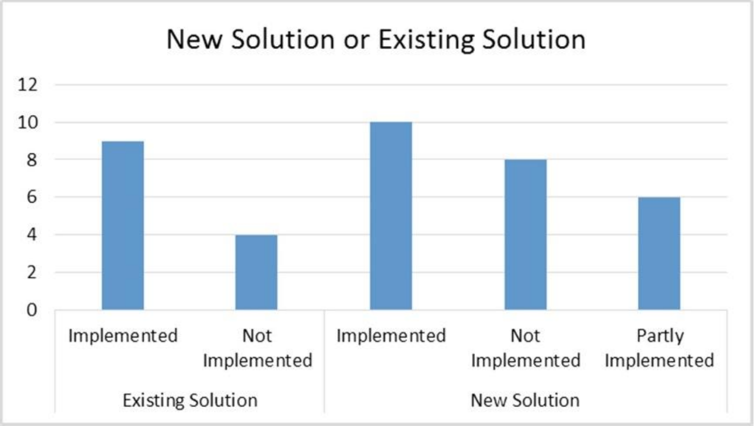

Figure 3.8 Implementation of solutions

The occurrence of standards proposed or implemented in the research papers included into the systematic mapping can be seen in Figure 3.9. The figure includes all the standards either implemented or proposed in systematic mapping papers. As clearly seen in the figure, REST can be called out as the standard for Microservices, even though there is one outlier paper which used non-REST protocol in their study [60].

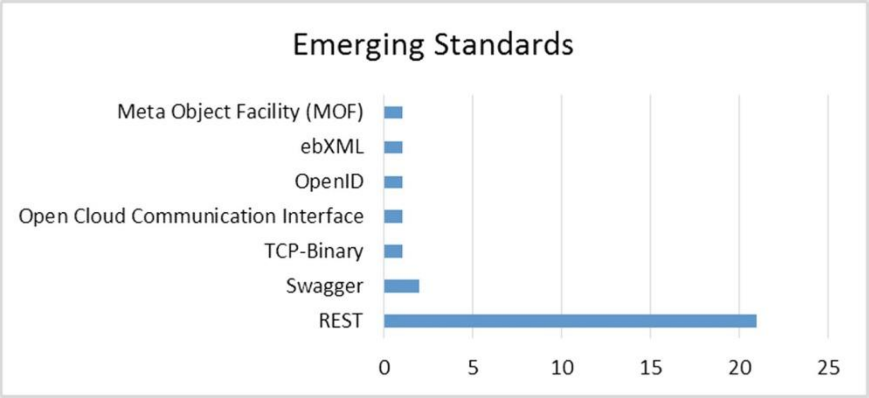

Figure 3.9 Emerging standards

Only Swagger is used for microservice markup language. It is interesting to see that WADL or API Blueprint is not mentioned.

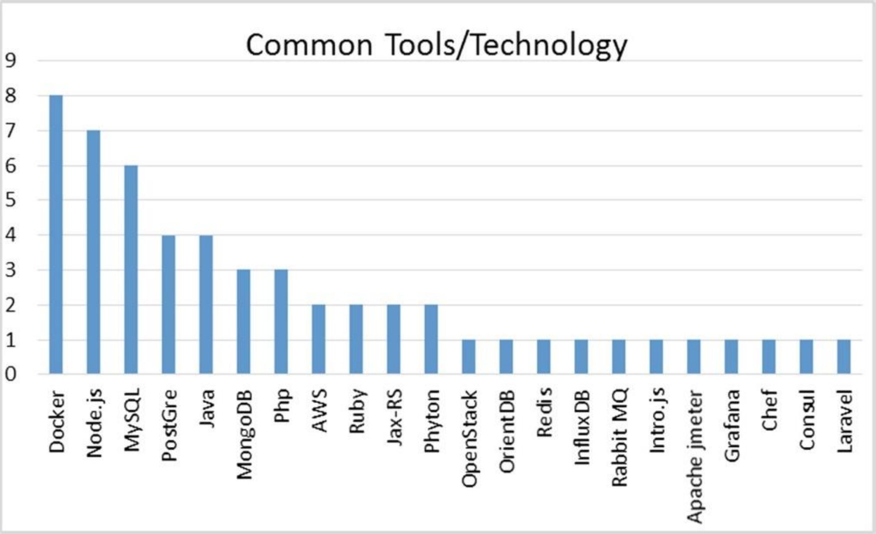

Figure 3.10 Common tools used for implementing microservices

The occurrence of tools used in proposed or implemented solutions can be found in Figure 3.10. Docker is seen as the most frequently used tool in studies.

The microservices topic is new and the official research started to show up in research papers in 2015. As a result, it is expected for the number of research on microservices to increase over time. Figure 3.11 shows publication numbers over time. The last search was carried out on the Web of Science in January 2017. On the figure, the line shows the trend. It is seen that the amount of papers increases radically and the trend line is going up.

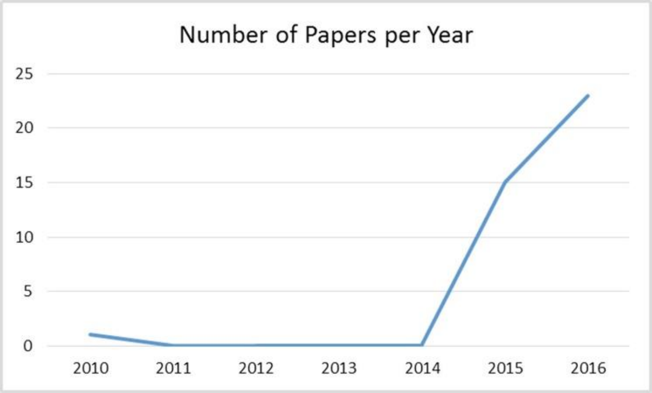

Figure 3.11 Number of microservices papers over time without applying exclusion criteria (searched on January 20, 2017)

Considering the mapping results, we can conclude that microservices was a trending topic in 2017. The same search for Figure 3.11 was repeated at the end of 2019. The resulting trend line shows similar results. Note that some of the papers published in 2019 did not make it into the web of science index yet.

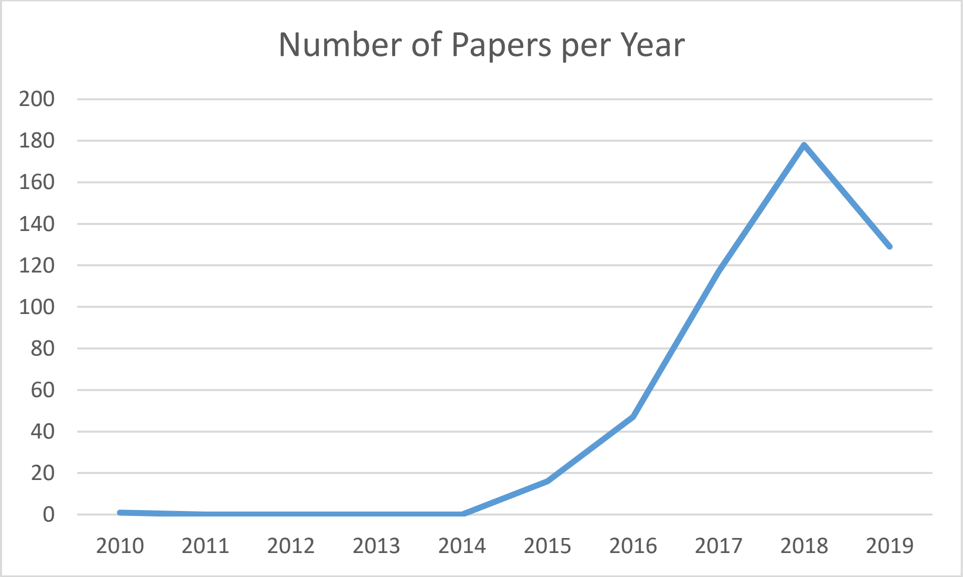

Figure 3.12 Number of microservices papers over time without applying exclusion criteria (searched on January, 2020)

3.2. Studies on Coupling and Cohesion Calculations of Microservices

The literature survey was carried out for identifying the research which focus on the coupling and cohesion of microservices. Two separate searches were carried out on

Web of Science [48]. One of the search was for identifying the research which focuses on the coupling aspect of microservices – Microservice Coupling Search. The other search was for identifying the research that focuses on the coherency aspect of a microservice – Microservice Coherency Search.

3.2.1. Microservice Coupling Search

The Microservice Coupling Search was carried out with the following keywords on 3rd of August, 2019: ((microservice OR "Micro service") AND ((coupling OR coupled) OR couple)). The search resulted in forty-two research papers. Out of these papers thirty of them neither focus on the boundary identification (granularity), nor the effect of coupling on finding the right size. The rest of the related papers are listed below.

There have been several research which applied Model Driven Design (MDD) to microservices architecture. In [88], MSA MDD and SOA MDD are conceptually compared. Another research which incorporates MDD into microservices is [89]. They propose using DDD for identifying the microservices. Then, Model Driven Design is used for transforming the domain models into more specific models where the developers could interpret better while implementing. The proposed example only has two microservices and there is no ambiguity about the bounded context. They later added aspect-oriented modelling [90] to their previous research [89]. This time their example was more elaborate. However, how they came up with the service boundaries is not covered in their research.

One real-life example on generating microservices is [91]. It explains the best practices adopted for microservices usage in one of Europe's biggest e-commerce sites. In order to improve resiliency, they favor decoupling as share nothing principle. They started with four loosely coupled applications and moved into 12 verticals. They define verticals as a part of the platform that is responsible for a single bounded context. According to their definitions, verticals are coarser grained then microservices. They have not clearly explained what the difference between a microservice and a vertical is and why they have used "vertical" instead of microservice.

Given that the microservice is a fairly new approach [92] [93], there have been multiple studies around dividing the existing monolith applications into microservices. One such research [94], suggests the existing services are decoupled by converting them into separately deployable components. Their focus was not around revisiting the service boundaries. In [95], Promise Theory was used for forming the microservices. It is claimed that the formed microservices are loosely coupled, but no coupling measurement is included. One other research on moving from monolith to microservices [20], mentions that there are many factors to divide a monolithic application into microservice such as functional dependency, function popularity and security. In their work, they divided the monolithic application in such a way that each function is a different service. Their focus was around comparing the monolith application's performance to the microservices. They neither investigated the other possibilities of divisions, nor measured the coupling values of the resulting microservices.

Another research [21] focusing on extraction of microservices from monolith application, leverages static and dynamic analysis. First, they generate a call graph using the static analysis. Then by tracing data and function calls in real time, they generate the coupling degree of functions and classes from the interaction frequency. Using k-means clustering algorithm, they decide on the partition (granularity) of microservices. In their research, they mention that the dynamic analysis method has a drawback of being limited to automated test coverage. In another study [5], the authors are proposing an automated method for identifying possible microservices out of a monolith application. They measure static coupling based on abstract syntax trees. The evolutionary coupling is measured by detecting changes between consecutive commits. Then, a graph clustering algorithm is used for identifying the microservice candidates. The resulting candidates were examined with the developers for cohesion values. There is no detail on how the cohesion values were measured.

As seen above, there has been detailed research on extraction of microservices from monolith applications. It is no surprise that there also has been a systematic mapping on migration of legacy systems into microservices. The study aimed to look into the challenges and the quality attributes of microservices as well as the motivation behind moving from a monolith system into a microservices architecture.

Besides MDD, DDD was also applied for microservices architecture. One such example describes the architecture of a community healthcare [96]. DDD was mentioned in the study. However, their main focus was not around how the division was established. One notable study around applying DDD microservices is [97]. They tried three different ways to extract microservices from a monolith application. They first tried logical coupling strategy as an extraction method. The coupling is measured based on the revision history of classes that are changed together. The second extraction method was based on contributor coupling which is calculated by the classes changed by the same contributors mostly. The third one was semantic coupling, which was based on DDD. Out of these methods, DDD gave the best results.

3.2.2. Microservice Cohesion Search

Up to now Microservice Coupling Search results were discussed. Now we will look into the Microservice Coherency Search. The search was carried out with the following keywords on 7th of August 2019: ((Microservice OR "micro service") AND (coher OR coherency OR coherent OR cohesion OR cohesive)). The search resulted in 12 research papers. 4 of them were already evaluated as part of the previous Microservice Coupling Search. Out of the remaining 8 research papers, 3 of them neither focus on the boundary identification (granularity), nor the effect of coherency on finding the right size. The rest of the 5 papers are listed below.

Similar to research mentioned as part of Microservice Coupling Search, the theme around converting legacy applications to microservices extends into Microservices Coherency Search results as well. One such study [98] performs SArF clustering algorithm to identify the possible microservices from 2 different monolith applications. They were expecting their results in the first application to be similar to the original microservices created for that application. They were not as close as they have expected. In order to improve the results, they are planning to make some modifications. Besides clustering algorithms, applying Promise Theory for microservice identification was also used [99]. That particular research does not include information about validating their result by measuring coherency.

Machine learning algorithms were also applied for microservices identification. In [100], the genetic algorithm was applied for identifying and improving the microservices. This resulted in a 20% of improvement. Their evaluation criteria were neither coupling, nor cohesion. The research paper of [101], proposes a method for identifying the microservices by evaluating the Open API specification of code by using DISCO (a database for identifying the semantic similarity of terms). They mentioned that when the results were too coarse grained, they had to rerun the algorithm in order to improve the results.

[102] proposes a variability based, pattern driven architecture migration. Depending on the needs of the company and the goals, certain patterns for migrating to cloud is advised. The research mainly focuses on the migration process rather than focusing on how to divide a certain service into microservices.

3.2.3. Search Results

Up to now, we looked into 12 coupling related and 5 cohesion related papers mostly focused on migration of existing legacy code into microservices architecture. Only 4 papers were related to writing a microservice from scratch or improving an already existing microservice. There were 2 papers mostly talking about concepts but not working on a real-life example with empirical results.

As seen in Table 3.2, the hands-on research about starting with microservices while creating a new application, has not yet been studied in detail. The percentage of hand-on research focusing on the migration of legacy applications is 75% (excluding the current study). In order to shed light into that area, the current research paper is aiming to get an answer at design time. As seen on [97], DDD is a good means for identifying the microservices at design time. However, the problem around how to define the bounded context is not clear and open to research. That is why this research proposes measuring coupling and coherency values for identification of the best possible bounded context. The last row in Table 3.2 shows that the current study is one of the few examples of having an empirical research focusing on creating microservices from scratch or improving an existing one.

Table 3.2 Mapping results of Microservice Coupling Search & Microservice Cohesion Search.

Referenced Papers	**Conceptual / Implementation**	**Migrating Legacy**	**Creating/Improving Microservices**
[89]	Conceptual,	N/A	N/A
[95]	Conceptual	N/A	N/A
[90]	Implemented	Yes	No
[5]	Implemented	Yes	No
[21]	Implemented	Yes	No
[94]	Implemented	Yes	No
[103]	Implemented	Yes	No
[20]	Implemented	Yes	No
[97]	Implemented	Yes	No
[96]	Implemented	Yes	No
[99]	Implemented	Yes	No
[98]	Implemented	Yes	No
[102]	Implemented	Yes	No
[91]	Implemented	No	Yes
[88]	Implemented	No	Yes
[100]	Implemented	No	Yes
[101]	Implemented	No	Yes
Current Thesis	Implemented	No	Yes

CHAPTER 4

METHOD

In the current chapter, we first explain the two sample applications chosen as the basis for this thesis. Then we explain how we came up with the extra designs/applications we generated. We then explain the measurement methods used. Finally, we finish the chapter by explaining the overall method.

4.1. Chosen Example Applications

As part of this thesis study, two different example applications were chosen both of which were constructed by applying domain-driven design. The first application was published in IEEE [26]. The second application is a well-known .Net Core application for demonstrating microservices example application [27].

4.1.1. Original Cargo Application

The first example application is a UML class diagram for a cargo application. The example was generated as part of [104] and later reused in [26]. As shown in Figure 4.1, the cargo application has three different bounded contexts. The main bounded context is around the cargo entities. The cargo entity holds different customers, some of which can be shippers and the others are receivers. They are differentiated based on the role. The cargoes’ handling events are kept in the delivery history. Some of those handling events may involve a movement from a source to a location. The cargo has a goal to reach the destination based on delivery specification.

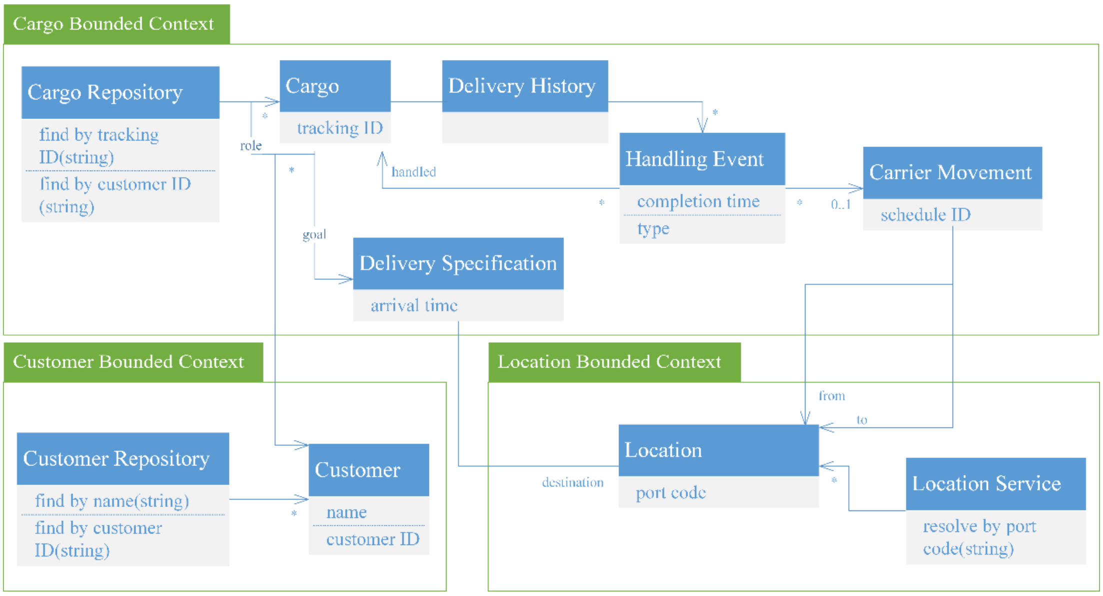

Figure 4.1 The UML class diagram for the Cargo application

The cargo bounded context consists of the following items:

- The cargo repository: It is a collection of different cargo entities. It is responsible for carrying out the persistency operations for the cargo aggregate. It allows the search of the cargo either by a tracking ID or a customer ID.
- The cargo aggregate: It consists of multiple different items, such as the cargo item itself as well as a delivery history, delivery specification and handling event.
- The delivery specification value object: It is a value object as it doesn't have to have its own identity but rather relies on the existence of the cargo item. If specified, the delivery specification presents the preferred arrival time.
- The handling event entity: It is an entity because each event has its own identity. It cannot exist without a cargo. As a result, the handling event entity is part of the cargo aggregate.
- The carrier movement entity: Some handling actions may result in a carrier movement. The carrier movement is a result of another entity. However, it has a purpose which is more than just giving a certain attribute. As a result, the carrier movement is another entity as well.
- The delivery history entity: It stores the history of handling events. Each history item has its own identity. As a result, the delivery history is an entity.

The customer bounded context consists of the following constructs:

- The customer entity: It is an entity representing the person who is sending or receiving the cargo.
- The customer repository: It is responsible for shielding the details of persistence of customer.

The location bounded context has the following items:

- The location service: It does not belong to any aggregate or entity.
- The location entity: It is served an independent item with its own ID.

4.1.2. Original eShopOnContainers Application

The second chosen sample application is a cloud based e-commerce application. The end user browses the items on the website, adds the items to a cart and checkouts [27].

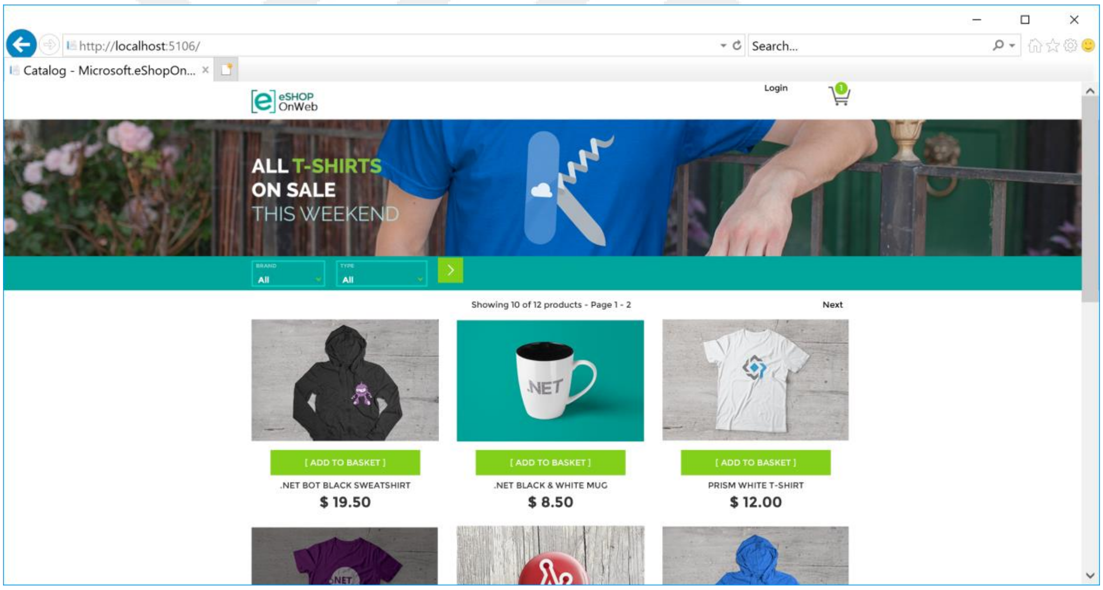

Figure 4.2 eShop application screenshot

The eShopOnContainers example has 43 entities and 7 services. In order to give an idea about the size and complexity of the application, the type dependency diagram is shown Figure 4.3. There are more than 400 types in eShopOnContainers application.

The eShopOnContainers application is based on new technology stack, such as .Net core, Docker, Mediatr, Integration Events between different microservices, etc. In order to run the system, 16 different Docker containers are composed up.

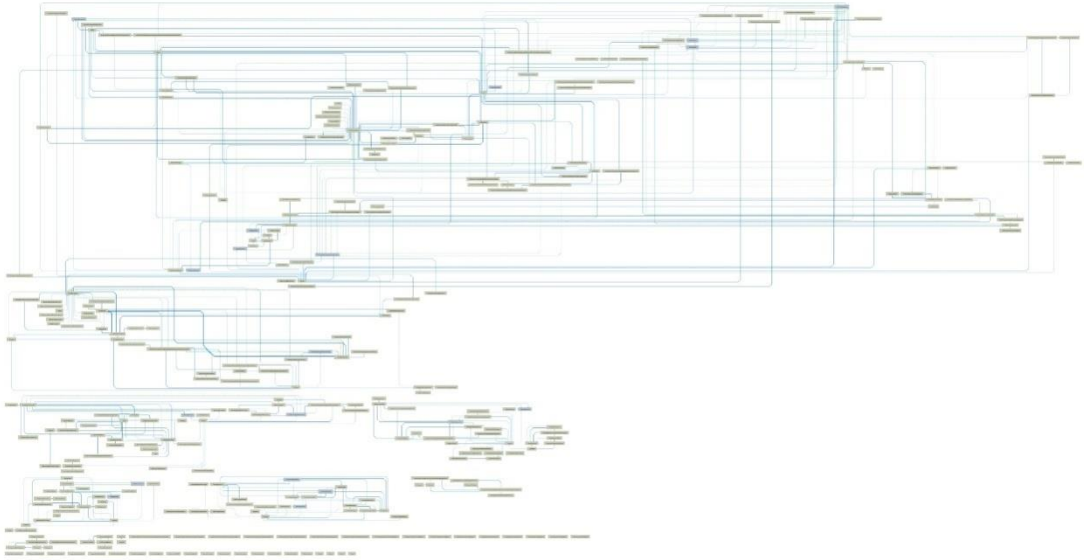

Figure 4.3 eShopOnContainers type dependency diagram

The eShopOnContainers application is a complete solution for a simple online shopping site. Given that it has around 400 types, there are many different entities for the seven services in the application. The entities residing in each service are detailed in Figure 4.4 to Figure 4.9.

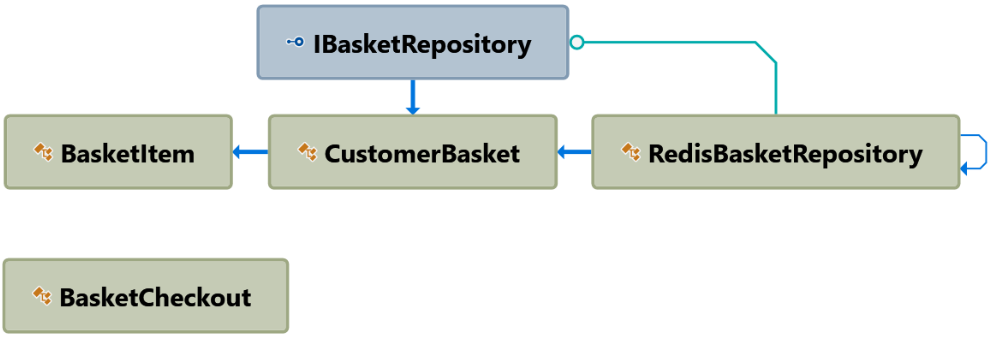

Figure 4.4 Basket entities

The basket repository is responsible for shielding NoSQL DB operations for the basket items. The basket item is an entity as it has its own identity (Figure 4.4). The customer basket consists of the basket items added to the customer basket. The basket information is kept in Redis cache.

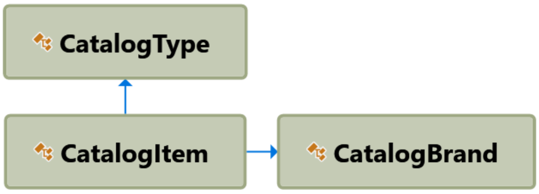

Figure 4.5 Catalog entities

The catalog item is an entity which can be added to a basket. When it is added, the added instance becomes a basket item. The catalog item has catalog type and catalog brand as value objects (Figure 4.5).

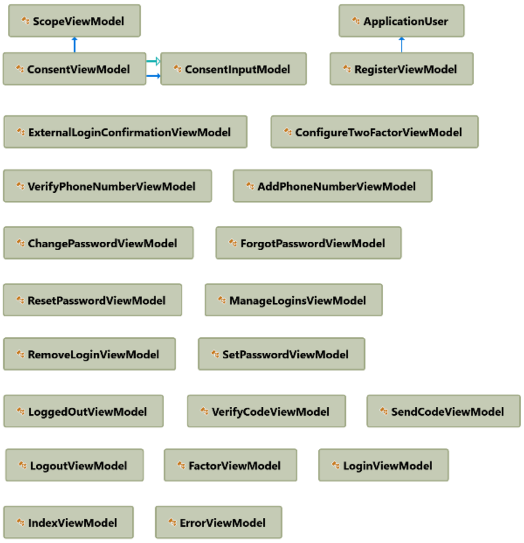

Figure 4.6 Identity entities

The identity entity has many value objects associated to it. As seen in Figure 4.6, the main entity is the application user entity.

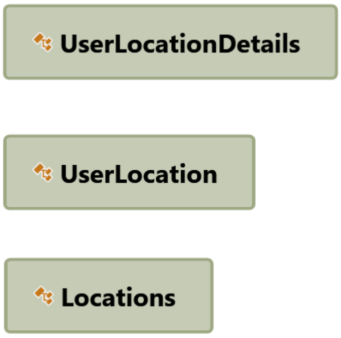

Figure 4.7 Location entities

The locations entity is a repository for keeping the locations of the users. The user location is an entity for each user (Figure 4.7).

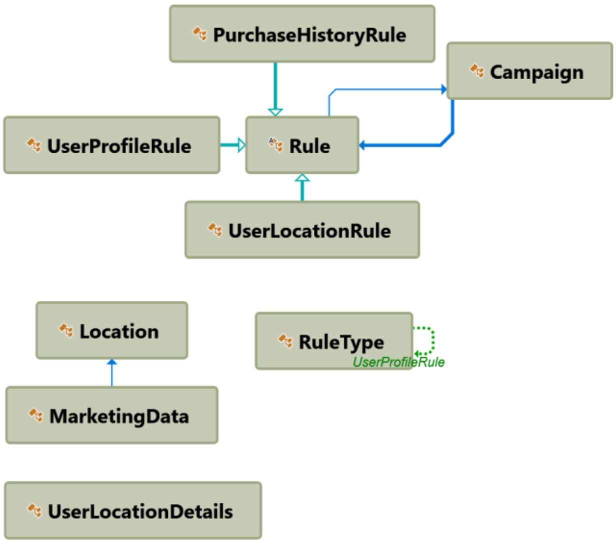

Figure 4.8 Marketing entities

The marketing bounded context has marketing data entity and the campaign entity as the main entities as seen in Figure 4.8. It also has several supporting value objects, such as rule, etc.

The only service which has aggregates is the Ordering service. The ordering service aggregates can be seen in Figure 4.9.

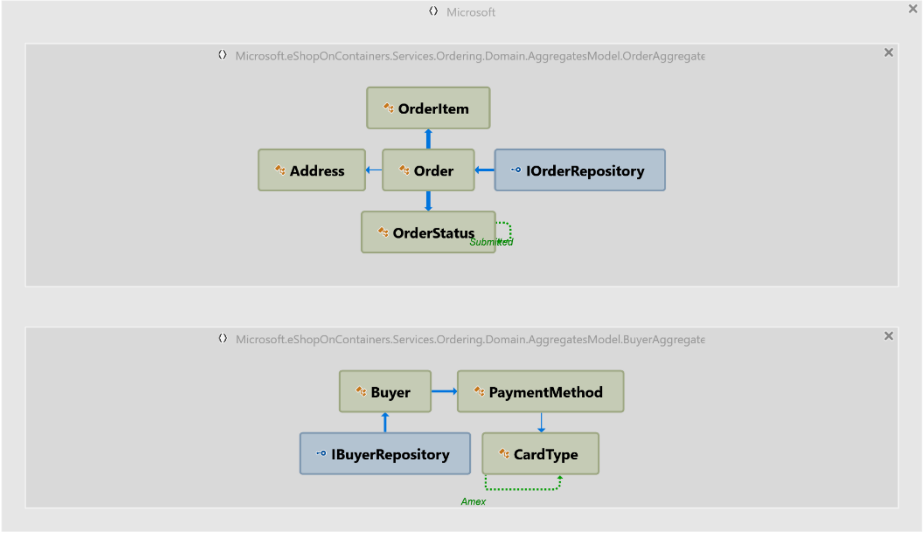
Microsoft
Microsoft.eShopOnContainers.Services.Ordering.Domain.AggregatesModel.OrderAggregate
OrderItem
Address
Order
IOrderRepository
OrderStatus
Submitted
Microsoft.eShopOnContainers.Services.Ordering.Domain.AggregatesModel.BuyerAggregate
Buyer
PaymentMethod
IBuyerRepository
CardType
Amex

Figure 4.9 Ordering aggregates

4.2. Generated Applications

As part of this thesis, each example application (Cargo application and eShopOnContainers application) was updated to generate more granular and less granular applications. The main goal is to collect different measurements on those results and compare the results to see the best performing application.

The more granular applications were generated by dividing an existing bounded context into two different bounded contexts. The less granular applications were generated by combining two different bounded contexts into one.

4.2.1. Generated Cargo Applications

4.2.1.1. More Granular Cargo Application

The original example is divided into smaller constructs as seen in Figure 4.10. The resulting application is called More Granular Microservices (MGM). The MGM cargo example has four different bounded contexts. The handling bounded context is the newly added one.

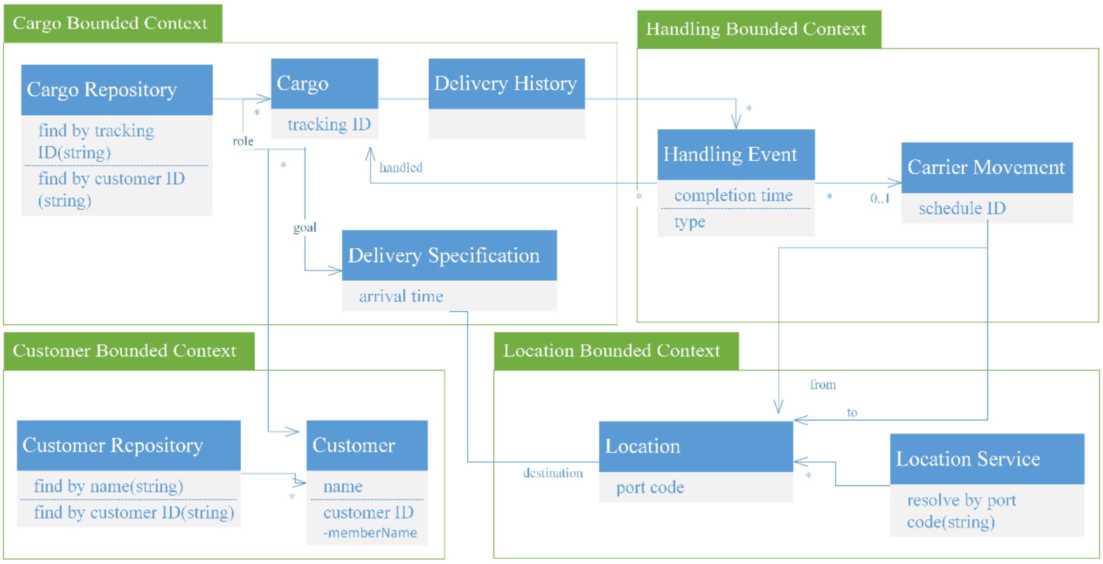

Figure 4.10 The UML class diagram for the MGM Cargo example

4.2.1.2. Less Granular Cargo Application

The original example is combined into bigger constructs as seen in Figure 4.11. The resulting application is called Less Granular Microservices (LGM). The LGM cargo case study was created by combining customer bounded context into the cargo bounded context. It has a total of two bounded contexts.

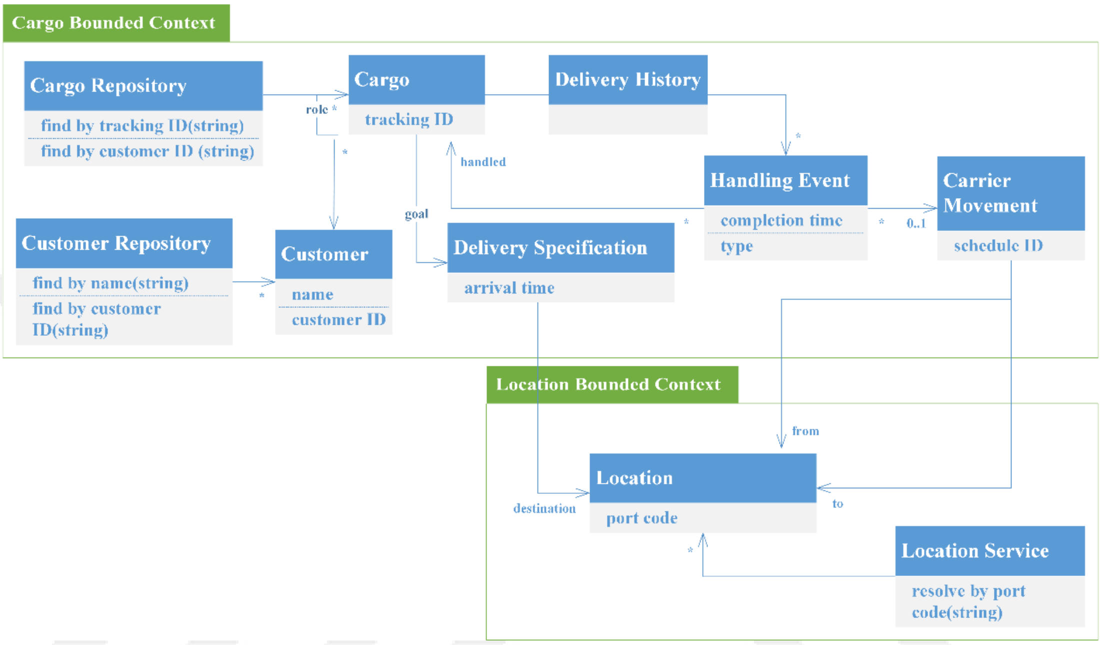

Figure 4.11 The UML class diagram for the LGM Cargo example

4.2.2. Generated eShopOnContainers Applications

The original application code is located at GitHub [105]. The original application code has Massachusetts Institute of Technology (MIT) License. We also published the generated code applications as part of this thesis under MIT License as well [106], [107].

The original eShopOnContainers application (called eShopOrginal) has seven microservices. In order to construct More Granular Microservices (called eShopSmall) and Less Granular Microservices (called eShopBig), out of those seven microservices only four of them were modified. That is why the focus is on the four microservices as seen in Figure 4.12.

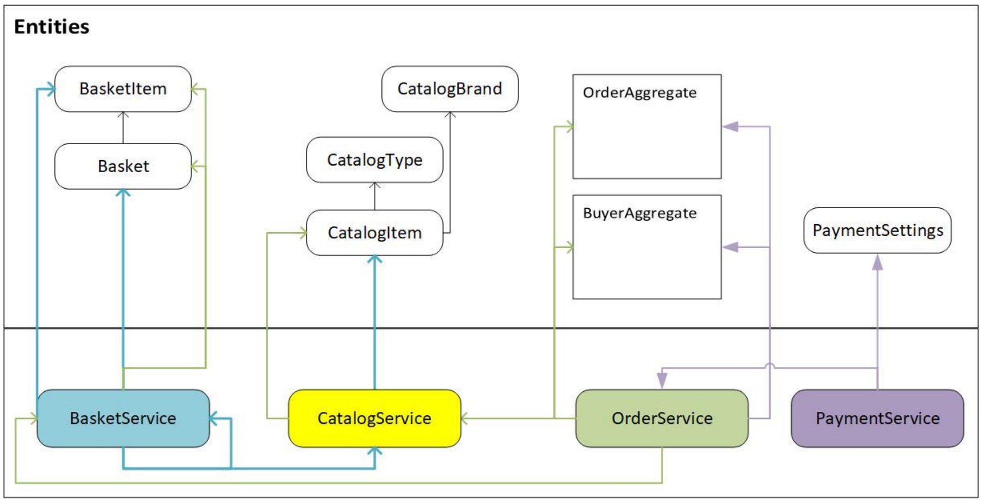

Figure 4.12 Four main services in eShopOnContainers

4.2.2.1. More Granular eShopOnContainers Application

The most complex microservice in eShopOrginal example was the ordering service. It had two aggregates in it: Order aggregate and buyer aggregate. That is why ordering service was the best candidate to divide into two different microservices: Ordering microservice and buying microservice.

As seen in Figure 4.12, the ordering service has two concepts in it: Order and buyer. As part of creating smaller microservices, the ordering service was divided into ordering and buying services. The divided eShopOnContainers is called as eShopSmall and the original is called as eShopOriginal.

The buyer aggregate is moved to buyer service and the order aggregate stayed in the ordering service. In the original application during checkout, the order was updated along with the payment method information within the same database. In the new design, the buyer aggregate resides in a different database. While the order related actions are taken, integration events are published for each microservice. The microservices that are registered to that integration event will get the notification and take the related action. This pattern along with extensive dependency injection helps with the decoupling of the overall system.

While dividing the microservice, there was a design decision on either applying eventual consistency or distributed transaction. The former is chosen in the current case. The domain-driven design model of eShopSmall can be seen in Figure 4.13.

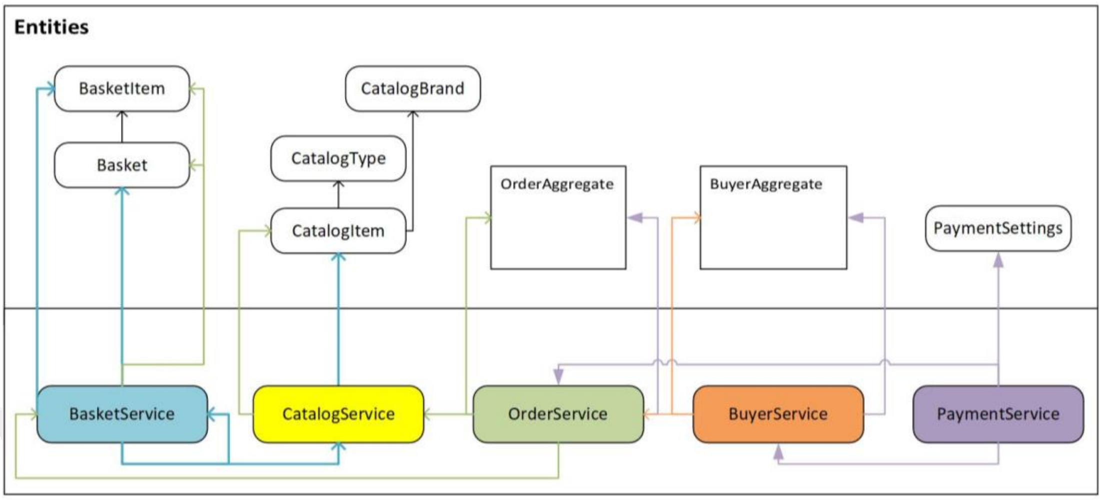

Figure 4.13 Five different services in eShopSmall

4.2.2.2. Less Granular eShopOnContainers Application

In order to get less granular application, two different microservices were combined into one microservice. As seen in figure 4.12, the payment service is only interacting with the order service. In order to generate the eShopBig example, the order service and the payment service are combined into order service as seen in Figure 4.14.

The PaymentSettings entity is moved to order service and the payment service was disabled. In the original application during checkout, the payment service was triggering an integration event. In the new application, that turned into a domain event rather than an integration event. The database for payment service and the order service were combined as well.

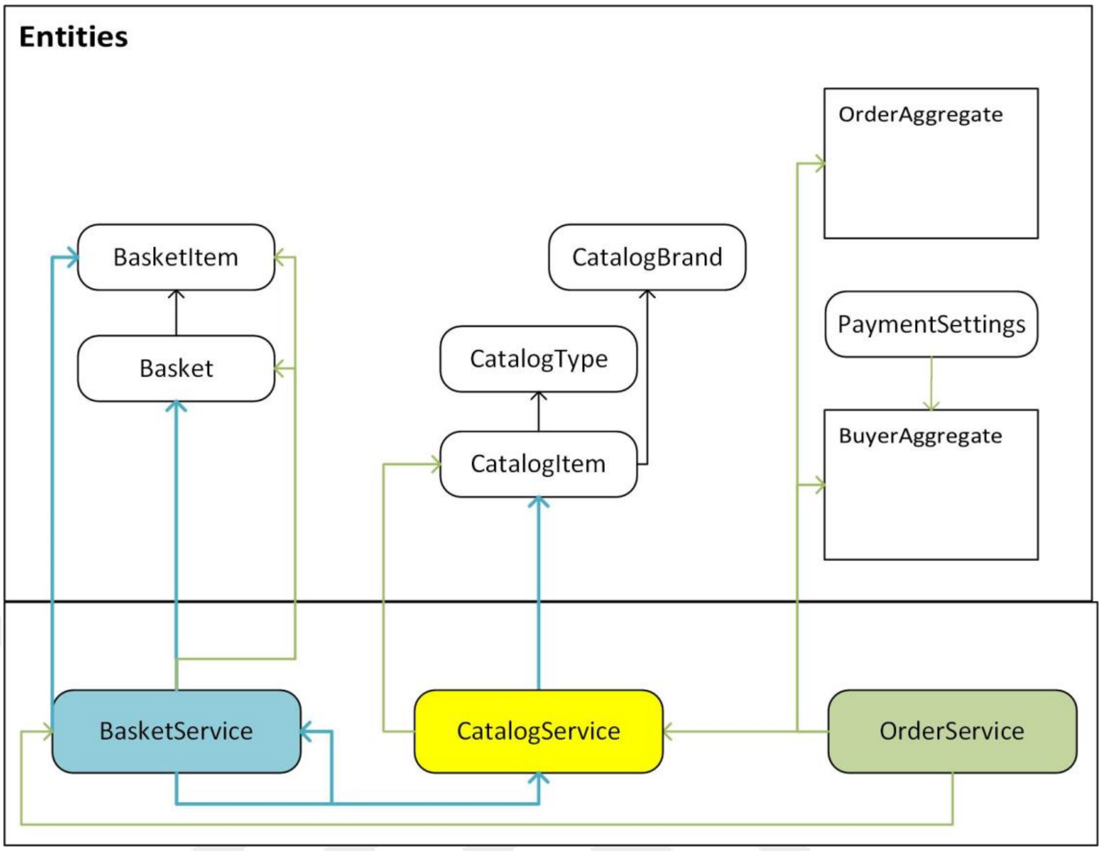

Figure 4.14 Three different services in eShopBig

4.3. Measurement Methods

The coupling is measured as afferent coupling and efferent coupling as mentioned in [108]. The afferent coupling in this study is measured at the assembly level (Bounded context level for the UML class diagrams). It calculates the number of types outside the assembly which depend on the types within the current assembly. High afferent coupling indicates that there are many other assemblies relying on the current one. The desirable behavior is to have low afferent coupling. The average afferent coupling values are calculated as seen in Equation (1).

$$Avg(C_a) = \frac{\sum_{i=1}^{n}(C_a)_i}{n} \quad (1)$$

Where, $(C_a)_i$ denotes the afferent coupling of an assembly *i* and n is the number of assemblies.

The efferent coupling in this study is measured by calculating the number of types outside the current assembly (Bounded context level for the UML class diagrams), which the types within the current assembly use. If the efferent coupling of an assembly is too high, it indicates that the current assembly relies on many other assemblies. In an ideal world, efferent coupling is expected to be low. The average efferent coupling values are calculated as seen in Equation (2).

$$Avg(C_e) = \frac{\sum_{i=1}^{n}(C_e)_i}{n} \tag{2}$$

Where, $(C_e)_i$ denotes the efferent coupling of an assembly i and n is the number of assemblies.

In this case study, the cohesion is measured as a relational cohesion within assemblies (bounded context) as mentioned in [108]. The desirable behavior is having high relational cohesion. The relational cohesion (H) of an assembly is calculated as seen in Equation (3).

$$H = \frac{R+1}{N} \tag{3}$$

Where, R denotes the number of relations of types within the assembly and N is the number of types within the assembly. The average relational cohesion values are calculated as seen in Equation (4).

$$Avg(H) = \frac{\sum_{i=1}^{n} H_i}{n} \tag{4}$$

Where, H_i denotes the relational cohesion of an assembly i and n is the number of assemblies. For the cargo application, the calculations were carried out manually. Given that the eShopOnContainer application has around 400 types, an automated tool [109] called NDepend was used for collecting the relational cohesion, afferent coupling, and efferent coupling values. The tool calculates static code metrics by analyzing the code, such as afferent coupling, efferent coupling and relational cohesion at the assembly level. The average values were calculated based on the results received for each assembly.

4.4. The Overall Method for Carrying out the Case Studies

In order to answer the aforementioned research questions in Chapter 1, we have looked into applications with different sizes. As seen in Figure 4.15, we first selected two different applications. Our research questions are around domain-driven design. That is why both applications were specifically chosen to be originally developed by applying domain-driven design.

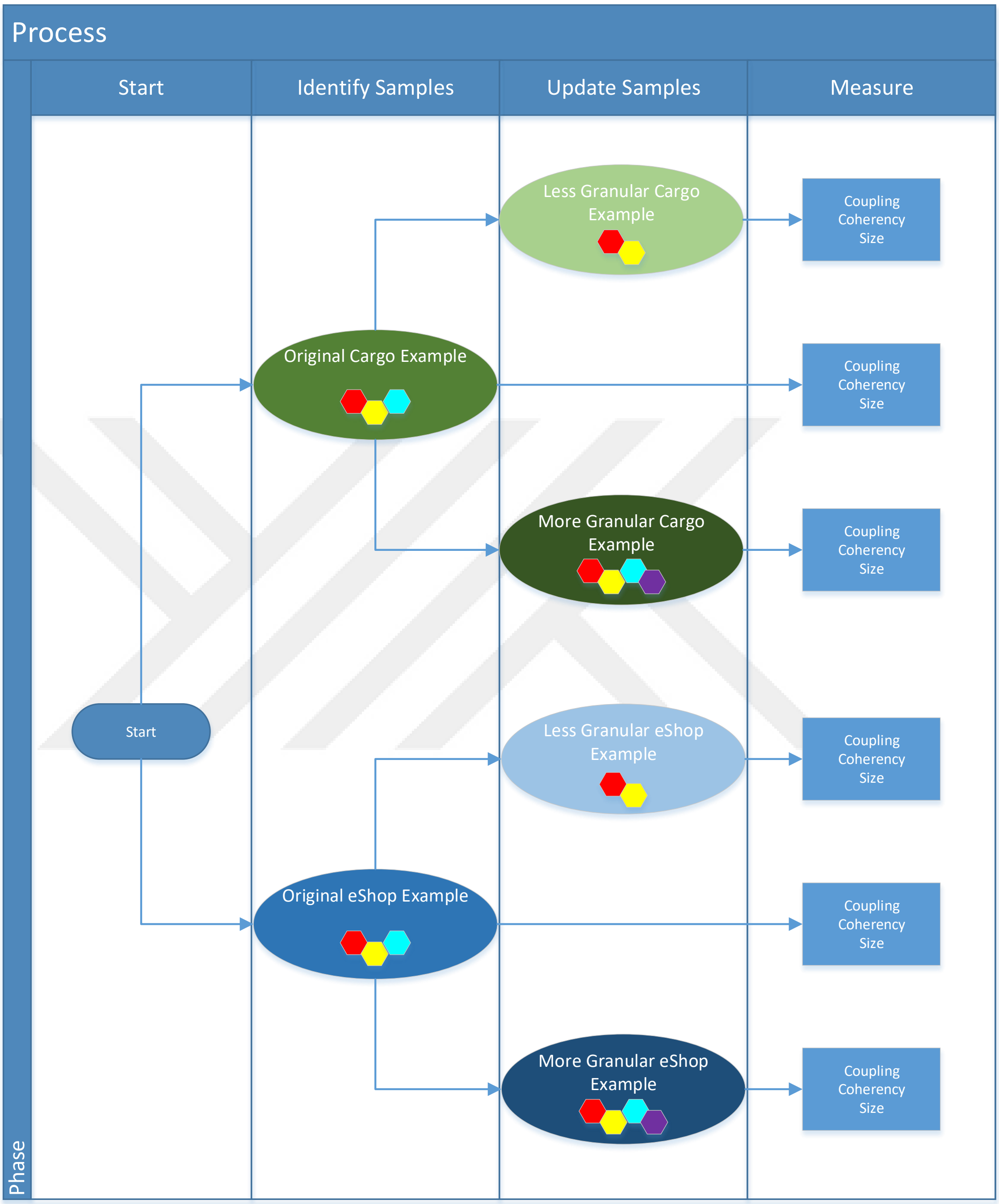

Figure 4.15 The process followed for sampling and measurements

The first application is based on UML class diagrams generated by previous research [104]. The second application is based on an existing microservices application – eShopOnContainers [27]. As shown in Figure 4.15, the next step is to generate more granular and less granular applications for each original application. After this step, we had six different applications:

- the original cargo application
- the less granular cargo application (bigger bounded context)
- the more granular cargo application (smaller bounded context)
- the original eShopOnContainers application
- the less granular eShopOnContainers applications (bigger microservices and less number of microservices in total)
- the more granular eShopOnContainers applications (smaller microservices and more number of microservices in total)

As seen in Figure 4.15, the third step is to collect the measurements explained in section 4.3. The resulting measurements were analyzed to answer the research questions laid out in Section 1.2. The final problem is a multi-objective decision making (MCDM). One of the most used MCDM method was applied – Analytic Hierarchy Process [110].

CHAPTER 5

MICROSERVICES MEASUREMENTS

As part of the current thesis study, the aim is to understand the factors that may play a role in finding the optimum size for a microservice. In this section we first collect the measurements for two different case studies. At the end of the chapter we analyze and discuss the results.

5.1. COSMIC Function Point Measurements

The more microservices would mean higher cost to develop but smaller cost to maintain (depending on the cohesion and coupling values). The COSMIC function point (CFP) analysis would give a good idea on the cost of development. We first calculate CFP values for the original cargo application and the generated cargo applications. Then we calculate the CFP values for the original eShopOnContainers application and the generated eShopOnContainers applications.

5.1.1. CFP Measurements for the Cargo Case Study

In order to collect the COSMIC function points for the original cargo application, the less granular microservices (LGM) cargo application, and the more granular microservices (MGM) cargo application, the Functional User Requirements (FUR) are identified. Given that all three applications are carrying out the same functionality, the FUR for all three applications are the same as well. The list of FUR can be seen below:

1. New cargo (1): A new cargo is created without any delivery specification.
2. New cargo (2): A new cargo is created with a specific delivery instruction.
3. Handle cargo (1): A new cargo handling action is taken which do not involve movement of the cargo (e.g. getting clearance from customs).

4. Handle cargo (2): A new cargo handling is carried out which involves moving the cargo from one location to another. The cargo is delivered to an intermediate location – not at the final destination yet.
5. Handle cargo (3): A new cargo handling is carried out which involves moving the cargo to the final destination.
6. Track cargo (1): List the cargo delivery history for a given tracking ID.
7. Track cargo (2): List the cargo delivery history for a given user name.

In the next section, the COSMIC function point calculation of each application is explained. For each FUR, all the data movements (DM) are calculated. Each integration event (crossing a service boundary) is an entry (E) and exit (X) action. Every time a read action is carried out, it is represented as (R) and the write action as (W). The total number of data movements gives the COSMIC Function Points.

5.1.1.1. CFP Measurements for the Original Cargo Application

In the original cargo example application there are three different bounded contexts (BC): Cargo BC, customer BC, and location BC. When user requirement triggers a switch from one bounded context to another, it is seen as a layer change and calculated as an entry and exit operation. When a user wants to send a cargo, it triggers an action in cargo bounded context. Cargo bounded context requests the user information from the customer bounded context. The CFP for this user requirement is 8 (Table 5.1).

Table 5.1 New cargo (1) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	1	New cargo(1)	E	Cargo info and user name		Cargo BC	1
			X	Get user ID by name		Cargo BC	1
			E		User name	Customer BC	1
			R		Get user ID by name	Customer BC	1
			X		User ID	Customer BC	1
			E	User ID		Cargo BC	1
			W	Cargo		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for New Cargo(1)							8

Some customers might want to add a specific delivery instruction (e.g. delivery time) while sending the cargo. In that case, besides the cargo aggregate the delivery specification value object should also be persisted in DB. As a result, the CFP for this user requirement is 9 (Table 5.2).

Table 5.2 New cargo (2) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	2	New cargo(2)	E	Cargo info and user name		Cargo BC	1
			X	Get user ID by name		Cargo BC	1
			E		User name	Customer BC	1
			R		Get user ID by name	Customer BC	1
			X		User ID	Customer BC	1
			E	User ID		Cargo BC	1
			W	Cargo		Cargo BC	1
			W	Delivery Specification		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for New Cargo(2)							9

When a cargo is handled with no movement operations (such as clearance from customs), the execution operations are simple. The handling event and type are read from DB. The new handling operation information is written back to the DB. The details around the handling event are written to the delivery history as well. Depending on the status of the delivery operation, either a confirmation message or an error message is returned. The COSMIC function point for simple handling operation is 6 as seen in Table 5.3.

Table 5.3 Handle cargo (1) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	3	Handle cargo(1)	E	Handle		Cargo BC	1
			R	Handling event		Cargo BC	1
			R	Type		Cargo BC	1
			W	Handling event		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(1)							6

When the handling operation involves a carrier movement, both cargo and the location bounded contexts get involved in the execution. The cargo bounded context passes the port no to the location service and gets the location information back. The handling event is updated. In addition, the details about the handling event are written into delivery history. The COSMIC function point for this user requirement is 12 as shown in Table 5.4.

Table 5.4 Handle cargo (2) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	4	Handle cargo(2)	E	Handle		Cargo BC	1
			R	Handling event		Cargo BC	1
			R	Type		Cargo BC	1
			W	Carrier movement		Cargo BC	1
			X	In/out port code		Cargo BC	1
			E		In/out port code	Location BC	1
			R		Get location from port	Location BC	1
			X		Location	Location BC	1
			E	Location		Cargo BC	1
			W	Handling event		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(2)							12

The user requirement seen in Table 5.5 is responsible for delivering the cargo to the right customer with obliging to the delivery instructions. When the handle action is received at the cargo bounded context, it checks the cargo type and writes the carrier movement. At the time of delivery, the customer information should be validated as well. In order to get the required customer information, the customer bounded context is called. Then the location information is retrieved from the location service based on the port code parameter passed in. After the delivery is completed, the information regarding the handling is written into the delivery history. The COSMIC function point value for this requirement is 16.

Table 5.5 Handle cargo (3) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	5	Handle cargo(3)	E	Handle		Cargo BC	1
			R	Handling event		Cargo BC	1
			R	Type		Cargo BC	1
			W	Carrier movement		Cargo BC	1
			X	Customer ID		Cargo BC	1
			E		Customer ID	Customer BC	1
			R		Get customer name by ID	Customer BC	1
			X		Customer Name	Customer BC	1
			E		In/out port code	Location BC	1
			R		Get location from port	Location BC	1
			X		Location	Location BC	1
			E	Location		Cargo BC	1
			R	Delivery specification		Cargo BC	1
			W	Handling event		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(3)							16

The cargo can be tracked by the tracking ID. The details for this functional user requirement is laid out in Table 5.6. When the tracking is carried out by tracking ID, the execution is simple. The tracking ID is used for reading the delivery history information for the supplied tracking ID. The COSMIC function point value for tracking a cargo with tracking ID is 3.

Table 5.6 Track cargo (1) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	6	Track cargo(1)	E	Tracking ID		Cargo BC	1
			R	Delivery history		Cargo BC	1
			X	List delivery history		Cargo BC	1
TOTAL CFP for Track Cargo (1)							3

If the cargo needs to be tracked by customer name, more operations are involved as seen in Table 5.7. The customer ID has to be pulled first based on the given customer name. Then the customer ID is used for pulling the cargo information. The delivery history has details around the related cargo, such as all the handling events and movements. As a result the CFP for this requirement is 9.

Table 5.7 Track cargo (2) CFP calculation for original cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	7	Track cargo(2)	E	Customer name		Cargo BC	1
			X	Customer name		Cargo BC	1
			E		Customer name	Customer BC	1
			R		Customer ID	Customer BC	1
			X		Customer ID	Customer BC	1
			E	Customer ID		Cargo BC	1
			R	Find cargo by Customer ID		Cargo BC	1
			R	List delivery history		Cargo BC	1
			X	List delivery history		Cargo BC	1
TOTAL CFP for Track Cargo(2)							9

The CFP values for all the functional user requirements are added up to find the size and complexity of the original cargo application. As seen in Table 5.8, the resulting CFP value for original cargo application is 63.

Table 5.8 Total CFP of original cargo application

FP Name	**CFP**
New cargo (1)	8
New cargo (2)	9
Handle cargo (1)	6
Handle cargo (2)	12
Handle cargo (3)	16
Track cargo (1)	3
Track cargo (2)	9
TOTAL	**63**

5.1.1.2. CFP Measurements for the MGM Cargo Application

The generated more granular cargo application has one more bounded context - Handling BC. The seven functional user requirements are examined for the MGM cargo application. Out of seven FURs, only three of them were different for MGM cargo application. All three of those FURs are related to handling the cargo.

The cargo handling can either be triggering a carrier movement or not. As seen in Table 5.9, when the handling event doesn’t involve carrier movement, the action starts at the cargo bounded context and moves on to the handling bounded context. When compared to the original cargo application, MGM cargo application has entry and exit actions across different layers. As a result the CFP for MGM cargo application is bigger than the original cargo application. The CFP value for this functional user requirement is 10.

Table 5.9 Handle cargo (1) CFP calculation for MGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	3	Handle cargo(1)	E	Handle		Cargo BC	1
			X	Handle		Cargo BC	1
			E		Handle	Handling BC	1
			R		Handling event	Handling BC	1
			R		Type	Handling BC	1
			W		Handling event	Handling BC	1
			X	Handling event		Handling BC	1
			E	Handling event		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(1)							10

When the cargo handling event involves carrier movement, three different bounded contexts need to orchestrate the execution steps between each other. The CFP for this user requirement is 14 (Table 5.10). This FUR was already complex in original cargo application. The MGM cargo application made it only a little more complex (CFP 14) than what it originally was (CFP 12).

Table 5.10 Handle cargo (2) CFP calculation for MGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	4	Handle cargo(2)	E	Handle		Cargo BC	1
			X	Handle		Cargo BC	1
			E		Handle	Handling BC	1
			R		Handling event	Handling BC	1
			R		Type	Handling BC	1
			W		Handling event	Handling BC	1
			W		Carrier movement	Handling BC	1
			X	In/out port code		Handling BC	1
			E		In/out port code	Location BC	1
			R		Get location from port	Location BC	1
			X		Location	Location BC	1
			E	Location		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(2)							14

Table 5.11 Handle cargo (3) CFP calculation for MGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	5	Handle cargo(3)	E	Handle		Cargo BC	1
			X	Handle		Cargo BC	1
			E		Handle	Handling BC	1
			R		Handling event	Handling BC	1
			R		Type	Handling BC	1
			W		Handling event	Handling BC	1
			W		Carrier movement	Handling BC	1
			X	Customer ID		Handling BC	1
			E		Customer ID	Customer BC	1
			R		Get customer name by ID	Customer BC	1
			X		Customer Name	Customer BC	1
			E		In/out port code	Location BC	1
			R		Get location from port	Location BC	1
			X		Location	Location BC	1
			E	Location		Cargo BC	1
			R	Delivery specification		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo (3)							18

As seen in Table 5.11, the most complex FUR for any of the cargo applications is the delivery of the cargo handling event. It involves four different bounded contexts. The resulting CFP is 18. The total CFP for MGM cargo application is more from the original application (Table 5.8).

Table 5.12 Total CFP of MGM cargo application

FP Name	CFP
New cargo (1)	8
New cargo (2)	9
Handle cargo (1)	10
Handle cargo (2)	14
Handle cargo (3)	18
Track cargo (1)	3
Track cargo (2)	9
TOTAL	**71**

5.1.1.3. CFP Measurements for the LGM Cargo Application

The LGM cargo application combined customer bounded context into the cargo bounded context. Out of the original seven FURs, four of them are affected by this change. The affected FUR are:

- New cargo (1)
- New cargo (2)
- Handle cargo (3)
- Track cargo (2)

The original cargo application made a call to customer bounded context to receive the customer ID. As customer and cargo bounded contexts are combined in LGM cargo application, the operations became simpler. There is no need for entry and exit operations between different layers. As a result the CFP for this functional requirement dropped from 8 to 4 (Table 5.13).

Table 5.13 New cargo (1) CFP calculation for LGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	1	New cargo(1)	E	Cargo info and user name		Cargo BC	1
			R	Get user ID by name		Cargo BC	1
			W	Cargo		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for New Cargo(1)							4

The customer can specify certain delivery instructions at the time of dropping the cargo to the delivery corporation (initiating a new cargo action). The specified instructions are written as delivery specification value object. Given that the customer and the cargo bounded contexts are combined, the operation for getting the user ID from name occurs at the same layer. Which means that there isn’t an entry or exit operation from one bounded context to the other. As a result the COSMIC function point for this functional user requirement has dropped from 9 to 5 as shown in Table 5.14.

Table 5.14 New cargo (2) CFP calculation for LGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	2	New cargo(2)	E	Cargo info and user name		Cargo BC	1
			R	Get user ID by name		Cargo BC	1
			W	Cargo		Cargo BC	1
			W	Delivery Specification		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for New Cargo (2)							5

The handling cargo operation for delivering the cargo to the final destination becomes simpler as well as seen in Table 5.15. Again one of the extra layer is removed and as a result, the related entry and exit operations are eliminated as well. The COSMIC function point for this functional user requirement drops from 18 to 14 CFP.

Table 5.15 Handle cargo (3) CFP calculation for LGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	5	Handle cargo(3)	E	Handle		Cargo BC	1
			R	Handling event		Cargo BC	1
			R	Type		Cargo BC	1
			W	Carrier movement		Cargo BC	1
			R	Get customer name by ID	Get customer name by ID	Cargo BC	1
			X	In/out port code		Cargo BC	1
			E		In/out port code	Location BC	1
			R		Get location from port	Location BC	1
			X		Location	Location BC	1
			E	Location		Cargo BC	1
			R	Delivery specification		Cargo BC	1
			W	Handling event		Cargo BC	1
			W	Delivery history		Cargo BC	1
			X	Error/Confirmation		Cargo BC	1
TOTAL CFP for Handle Cargo(3)							14

The functional user requirement for tracking the cargo by user name also becomes simpler in LGM cargo application. Because the user information and the cargo information are at the same bounded context (layer). As shown in Table 5.16, the CFP value for this requirement has dropped from 9 to 5.

Table 5.16 Track cargo (2) CFP calculation for LGM cargo application

FUR	FP	FP Name	DM Type	Data Group Desc. In Top Layer	Data Group Desc. In Lower Layers	Layer	CFP
1	7	Track cargo(2)	E	Customer name		Cargo BC	1
			R	Customer ID		Cargo BC	1
			R	Find cargo by Customer ID		Cargo BC	1
			R	List delivery history		Cargo BC	1
			X	List delivery history		Cargo BC	1
TOTAL CFP for Track Cargo(2)							5

The total CFP for LGM cargo application is 49 as seen in Table 5.17. The complexity of the application is reduced from 63 to 49 when compared to the original cargo application.

Table 5.17 Total CFP of LGM cargo application

FP Name	CFP
New cargo (1)	4
New cargo (2)	5
Handle cargo (1)	6
Handle cargo (2)	12
Handle cargo (3)	14
Track cargo (1)	3
Track cargo (2)	5
TOTAL	**49**

5.1.1.4. Summary of CFP Measurements for the Cargo Case Study

As seen in Figure 5.1, the more granular cargo application has the highest COSMIC function point value. The next application in cargo case study with the average COSMIC function point value is the original application. The least COSMIC function value belongs to the less granular cargo application.

These results indicate that the most complex cargo application is the more granular one. There are more number of bounded contexts and data is passed through different layers. When the application has less number of bounded contexts or layers, then the complexity and relatively the cost of implementing the application is reduced. So by looking at Figure 5.1, we can tell that the least costly application to develop is LGM cargo application.

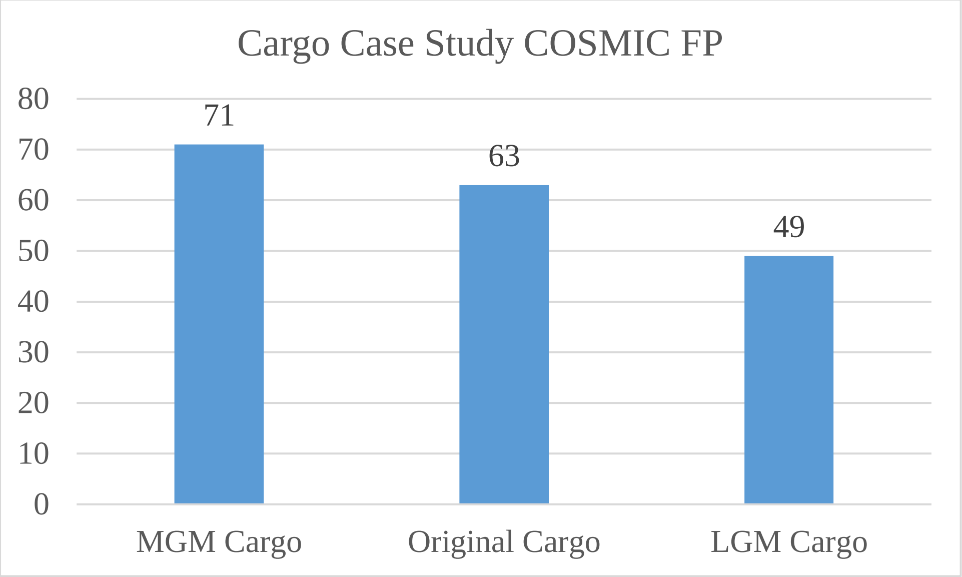

Figure 5.1 CFP results for cargo case study

5.1.2. CFP Measurements for the eShop Case Study

In order to collect the COSMIC function points for eShopSmall, eShopOriginal and eShopBig, the Functional User Requirements (FUR) should be identified. Given that all three applications (including original example and generated applications) are carrying out the same functionality, the FUR for all three applications are the same as well. The list of FUR can be seen below:

1. Login
2. List Items for Sale
3. Filter Based on Brand
4. Filter Based on Type
5. Add Item to a Basket
6. List Items in the Basket
7. Update Item in the Basket
8. Delete Item from the Basket
9. Checkout
10. Get Buyer Profile
11. Update Buyer Profile
12. Update Buyer Password
13. List Orders

14. Get Order Item Detail
15. Update Marketing
16. Logout

In the next section, the COSMIC function point calculation of each application (6 different applications were laid out in Chapter 4) are presented. For each FUR, all the data movements (DM) are calculated. Each integration event (crossing a service boundary) is an entry (E) and exit (X) action. Every time a read action is carried out, it is represented as (R) and the write action as (W).

5.1.2.1. CFP Measurements for eShopOriginal

As seen in Table 2.1 the entry (E) and exit (X) actions for the application layer and the services are calculated. In addition, the read (R) and write (W) operations are calculated as well.

Table 5.18 eShopOriginal "Login" CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	1	Login	E	Buyer info		Application Layer	1
			X	Is valid username and password		Application Layer	1
			E		Is valid username and password	Order Service	1
			R		Buyer	Order Service	1
			X		Authorization info	Order Service	1
			X	Authorization info		Application Layer	1
TOTAL CFP for Login							6

The login action starts at the application layer. The buyer information (username and password) is entered at the application layer. Then the application layer passes that information to the order service layer. In the order service, the entered information is compared against the value in the database (a DB read operation is carried out). Then the resulting authorization info is returned back to the application layer. There are a total of 6 data movements in login operation as seen on Table 5.18. So the COSMIC function points for login operation is 6.

Table 5.19 eShopOriginal “List items for sale” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	2	List Items for Sale	E	List catalog items request		Application Layer	1
			X	Retrieve catalog items		Application Layer	1
			E		Retrieve catalog items	Catalog Service	1
			R		Catalog item	Catalog Service	1
			R		Catalog brand	Catalog Service	1
			R		Catalog type	Catalog Service	1
			X		List of catalog items	Catalog Service	1
			X	List of catalog items		Application Layer	1
TOTAL CFP for List Items for Sale							8

When the user enters the application, the list of items up for sale are listed. At the start of this functional user requirement, the action for listing the catalog items is triggered at the application layer. The application layer passes this request to the catalog service layer. The catalog service gets the item, brand, and type information from DB. The resulting list is returned to the application layer. The total number of data movements is 8 as seen in Table 5.19, so the COSMIC function point for this functional user requirement is 8.

Table 5.20 eShopOriginal “Filter on brand” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	3	Filter Based on Brand	E	Filter catalog items based on brand request		Application Layer	1
			X	Retrieve catalog items		Application Layer	1
			E		Retrieve catalog items	Catalog Service	1
			R		Catalog item	Catalog Service	1
			R		Catalog brand	Catalog Service	1
			R		Catalog type	Catalog Service	1
			X		List of catalog items	Catalog Service	1
			X	List of catalog items		Application Layer	1
TOTAL CFP for Filter Based on Brand							8

The user can filter the items based on the brand. The request starts at the application layer and passed on to the catalog service layer. The catalog service retrieves the item, brand and type information from database and returns the list of results back to the application layer. As seen in Table 5.20, in order to execute the requested functional user requirement, 8 data movements are carried out (entry, exit, read and write). As a result, the COSMIC function point value for the current functional user requirement is 8 as well.

The user can filter the items in the catalog based on type. When such an action is received at the application layer, it is passed to the catalog service. Catalog service issues a query in the DB to retrieve the matching items. The resulting list is returned back to the application layer. As seen in Table 5.21, the CFP for this functional user requirement is 8.

Table 5.21 eShopOriginal "Filter on type" CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	4	Filter Based on Type	E	Filter catalog items based on type request		Application Layer	1
			X	Retrieve catalog items		Application Layer	1
			E		Retrieve catalog items	Catalog Service	1
			R		Catalog item	Catalog Service	1
			R		Catalog brand	Catalog Service	1
			R		Catalog type	Catalog Service	1
			X		List of catalog items	Catalog Service	1
			X	List of catalog items		Application Layer	1
TOTAL CFP for Filter Based on Type							8

When a user decides to add an item to the basket, this action is received by the application layer and directly passed to the basket service. The basket service reads the information about the current basket items first and then adds the new item to the list or updates the existing count if the item already exists in the list. If the action is successfully completed, a confirmation information is returned to the application layer. If the action fails, the error message is returned back. As shown in Table 5.22, the CFP for these operations is 7.

Table 5.22 eShopOriginal “Add item to a basket” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	5	Add Item to a Basket	E	Add item to a basket request		Application Layer	1
			X	Add basket item		Application Layer	1
			E		Add basket item	Basket Service	1
			R		Basket	Basket Service	1
			W		Basket item	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Add Item to a Basket							7

As seen in Table 5.23, when the user wants to see the items in the basket, the initial trigger happens at the application layer. Then the request is passed to the basket service. The basket service reads the information about the basket and all the items inside the basket and returns the result back to application layer. The CFP for these operations are 7.

Table 5.23 eShopOriginal “List items in the basket” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	6	List Items in the Basket	E	List items in the basket request		Application Layer	1
			X	Retrieve basket items		Application Layer	1
			E		Retrieve basket items	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			X		List of basket items	Basket Service	1
			X	List of basket items		Application Layer	1
TOTAL CFP for List Items in the Basket							7

In Table 5.24, the functional user requirement of updating an item in the basket is laid out. The request is received at the application layer and directly passed to the basket service. Basket service retrieves the basket and related basket item information and updates the basket item. The result is either returned as a confirmation or an error message to the application layer. The CFP for this functional requirement is 8.

Table 5.24 eShopOriginal “Update item in basket” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	7	Update Item in the Basket	E	Update item in the basket request		Application Layer	1
			X	Update basket item		Application Layer	1
			E		Update basket item	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			W		Basket item	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Item in the Basket							8

If the user decides to delete an item from the basket, the request is handled by application and basket service. The details of the operation are listed in Table 5.25. The CFP for this user requirement is 8.

Table 5.25 eShopOriginal “Delete item from basket” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	8	Delete Item from the Basket	E	Delete item from the basket request		Application Layer	1
			X	Delete basket item		Application Layer	1
			E		Delete basket item	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			W		Basket	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Delete Item From the Basket							8

When the user decides to checkout, the request is received at the application layer. The requested is sent to the order service. The order service creates a new order after receiving the buyer and order information. Once the order is created, the basket service is notified to delete the items in it. Then the payment service is triggered to realize the payment. This is the most complex functional requirement when compared to the rest of the functional requirements in eShopOnContainers application. As seen in Table 5.26, the CFP for “checkout” functional requirement is 17.

Table 5.26 eShopOriginal “Checkout” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	9	Checkout	E	Checkout request		Application Layer	1
			X	Create order		Application Layer	1
			E		Create order	Order Service	1
			R		Buyer	Order Service	1
			R		Order	Order Service	1
			W		Order	Order Service	1
			X		Error/Confirmation	Order Service	1
			E		Delete basket item	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			W		Basket	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			E		Initiate Payment	Payment Service	1
			R		Payment Info	Payment Service	1
			W		Pay	Payment Service	1
			X		Error/Confirmation	Payment Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Checkout							17

In “Get buyer profile” user requirement, application layer receives the information from order service and display it back to the user. The CFP is 6 (Table 5.27).

Table 5.27 eShopOriginal “Get buyer profile” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	10	Get Buyer Profile	E	Get buyer profile request		Application Layer	1
			X	Get buyer info		Application Layer	1
			E		Get buyer info	Order Service	1
			R		Buyer	Order Service	1
			X		Error/Confirmation	Order Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Get Buyer Profile							6

The buyer profile update starts in the application layer and ends in the order service (Table 5.28). As we have seen in Chapter 4, for the eShopOriginal application, the buyer information is kept in the order service. Which means that there are no extra data tossing between the layers. The total CFP for the current functional user requirement is 7.

Table 5.28 eShopOriginal “Update buyer profile” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	11	Update Buyer Profile	E	Update buyer profile request		Application Layer	1
			X	Update buyer info		Application Layer	1
			E		Update buyer info	Order Service	1
			R		Buyer	Order Service	1
			W		Buyer	Order Service	1
			X		Error/Confirmation	Order Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Buyer Profile							7

If the buyer decides to update his/her password, application layer receives the request and passes it on to the order service (Table 5.29). The order service reads user information and then updates the buyer password. If the operation is carried out successfully, a confirmation message is returned. If the operation fails, the error message is returned back to the application layer. The total CFP for the functional user requirement is 7.

Table 5.29 eShopOriginal “Update buyer password” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	12	Update Buyer Password	E	Update buyer password		Application Layer	1
			X	Update buyer info		Application Layer	1
			E		Update buyer info	Order Service	1
			R		Buyer	Order Service	1
			W		Buyer	Order Service	1
			X		Error/Confirmation	Order Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Buyer Password							7

If the buyer wants to check the list of orders he/she gave, “List orders” functional user requirement is triggered as shown in Table 5.30. This requirement is received at the application layer and processed at the order service layer. The order service reads the list of orders from database. The list of orders is returned from the order service back to the application layer. The total CFP for this functional user requirement is 6.

Table 5.30 eShopOriginal “List orders” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	13	List Orders	E	List orders request		Application Layer	1
			X	Retrieve orders		Application Layer	1
			E		Retrieve orders	Order Service	1
			R		Order	Order Service	1
			X		List of orders	Order Service	1
			X	List of orders		Application Layer	1
TOTAL CFP for List Orders							6

If the user wants to see the details of an item in the order, the user requirement specified in Table 5.31 is triggered. The action is served with the help of application layer and the order service layer. The resulting CFP is 6.

Table 5.31 eShopOriginal “Get order item detail” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	14	Get Order Item Detail	E	Get order item detail request		Application Layer	1
			X	Retrieve order		Application Layer	1
			E		Retrieve order	Order Service	1
			R		Order	Order Service	1
			X		Order detail	Order Service	1
			X	Order detail		Application Layer	1
TOTAL CFP for Get Order Item Detail							6

The logout action is directly carried out at the application layer. The request enters the application layer and exists. The resulting COSMIC function point value for logout functional user requirement is 2 as shown in Table 5.32.

Table 5.32 eShopOriginal “Logout” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	15	Logout	E	Logout request		Application Layer	1
			X	Log buyer out		Application Layer	1
TOTAL CFP for Logout							2

The requirement for updating marketing information starts at the application layer. The request is passed to the marketing service. The update is carried out in the service then the result is returned back to the application layer (Table 5.33). The CFP for marketing update is 7.

Table 5.33 eShopOriginal “Update marketing” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	16	Update Marketing	E	Update marketing		Application Layer	1
			X	Update marketing info		Application Layer	1
			E		Update marketing info	Marketing Service	1
			R		Marketing	Marketing Service	1
			W		Marketing	Marketing Service	1
			X		Error/Confirmation	Marketing Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Marketing							7

The total COSMIC function points (CFP) for eShopOriginal is 118 CFP as shown in Table 5.34.

Table 5.34 Total CFP of eShopOriginal

FP Name	CFP
Login	6
List Items for Sale	8
Filter Based on Brand	8
Filter Based on Type	8
Add Item to a Basket	7
List Items in the Basket	7
Update Item in the Basket	8
Delete Item from the Basket	8
Checkout	17
Get Buyer Profile	6
Update Buyer Profile	7
Update Buyer Password	7
List Orders	6
Get Order Item Detail	6
Logout	2
Update Marketing	7
TOTAL	**118**

5.1.2.2. CFP Measurements for eShopSmall (MGM)

The eShopSmall application has an extra service “Buyer service” when compared to the eShopOriginal application (Figure 4.12 and Figure 4.13). Out of 16 functional user requirements in eShopOriginal application, only 5 of them are different for eShopSmall. Those 5 FUR were related to just the order service before, now the functionality is divided between the order service and buyer service. The list FUR which are different for eShopOriginal and eShopSmall are listed below:

- Login
- Checkout
- Get buyer profile
- Update buyer profile
- Update buyer password

As seen in Table 5.35, the execution of login user requirement starts at the application layer and carried out at the buyer service and the authorization info is returned back to the application layer. The CFP for the login user requirement is 6.

Table 5.35 eShopSmall "Login" CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	1	Login	E	Buyer info		Application Layer	1
			X	Is valid username and password		Application Layer	1
			E		Is valid username and password	Buyer Service	1
			R		Buyer	Buyer Service	1
			X		Authorization info	Buyer Service	1
			X	Authorization info		Application Layer	1
TOTAL CFP for Login							6

The most complex functional user requirement in eShopSmall application is the checkout user requirement (Table 5.36). Besides the application layer, four different services are involved: Order service, basket service, payment service, buyer service. The request is received at the application layer and passed to the order service. The order service received the request and creates the order. Then it sends an integration event to notify the related services. The buyer service returns the buyer information; the basket service deletes the basket items. The payment service initiates the payment. The CFP for the checkout user requirement is 19. When compared to the eShopOriginal application, the CFP value for checkout operation has increased from 17 to 19.

The action for getting the user profile is passed from the application layer to the buyer service (Table 5.37). The buyer service gets the information from DB and returns it back to the application layer. The CFP for the user requirement is 6.

Table 5.36 eShopSmall “Checkout” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	9	Checkout	E	Checkout request		Application Layer	1
			X	Create order		Application Layer	1
			E		Create order	Order Service	1
			R		Order	Order Service	1
			W		Order	Order Service	1
			X		Error/Confirmation	Order Service	1
			E		Get Buyer Info	Buyer Service	1
			R		Buyer	Buyer Service	1
			X		Buyer Info	Buyer Service	1
			E		Delete basket item	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			W		Basket	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			E		Initiate Payment	Payment Service	1
			R		Payment Info	Payment Service	1
			W		Pay	Payment Service	1
			X		Error/Confirmation	Payment Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Checkout							19

If buyer decides to update his/her profile, the request is passed from the application layer to the buyer service layer as seen in Table 5.38. The buyer service updates the buyer profile at the database. If the operation was successful, it returns a confirmation message. If the operation fails, the buyer service returns an error message back to the application layer. The CFP for the current functional user requirement is 7.

Table 5.37 eShopSmall “Get buyer profile” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	10	Get Buyer Profile	E	Get buyer profile request		Application Layer	1
			X	Get buyer info		Application Layer	1
			E		Get buyer info	Buyer Service	1
			R		Buyer	Buyer Service	1
			X		Error/Confirmation	Buyer Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Get Buyer Profile							6

Table 5.38 eShopSmall “Update buyer profile” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	11	Update Buyer Profile	E	Update buyer profile request		Application Layer	1
			X	Update buyer info		Application Layer	1
			E		Update buyer info	Buyer Service	1
			R		Buyer	Buyer Service	1
			W		Buyer	Buyer Service	1
			X		Error/Confirmation	Buyer Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Buyer Profile							7

When the buyer wants to update his/her password, the main action is carried out at the buyer service layer. The result of the operation is returned back to the application layer either as a confirmation message or an error message. When compared to eShopOriginal, the only change is that, the database write operation is carried out by the buyer service rather than the order service. The CFP for the current functional user requirement is 7 (Table 5.39).

Table 5.39 eShopSmall “Update buyer password” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	12	Update Buyer Password	E	Update buyer password		Application Layer	1
			X	Update buyer info		Application Layer	1
			E		Update buyer info	Buyer Service	1
			R		Buyer	Buyer Service	1
			W		Buyer	Buyer Service	1
			X		Error/Confirmation	Buyer Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Update Buyer Password							7

The COSMIC function point calculation for eShopSmall application can be found in Table 5.40. The total CFP for eShopSmall application is 120. The difference of 2 CFP between the original eShopOnContainers application and the MGM eShopOnContainers application was in “checkout” functional user requirement. Given that the total number of COSMIC function points is bigger in eShopSmall when compared to the eShopOriginal application, it is costlier to write eShopSmall when compared to eShopOriginal.

Table 5.40 Total CFP of eShopSmall

FP Name	**CFP**
Login	6
List Items for Sale	8
Filter Based on Brand	8
Filter Based on Type	8
Add Item to a Basket	7
List Items in the Basket	7
Update Item in the Basket	8
Delete Item from the Basket	8
Checkout	19
Get Buyer Profile	6
Update Buyer Profile	7
Update Buyer Password	7
List Orders	6
Get Order Item Detail	6
Logout	2
Update Marketing	7
TOTAL	**120**

5.1.2.3. CFP Measurements for eShopBig (LGM)

The eShopBig application has one less service. The “Payment service” was combined with the “Order service” (Figure 4.12 and Figure 4.14). As a result, the resulting application is less granular when compare to the original eShopOnContainers application.

Out of 16 functional user requirements in eShopOriginal application, only one of them is different for eShopSmall, which is “checkout” functional user requirement. Original example had extra entry and exit data movements into and out of the payment service.

As seen in Table 5.41, the checkout request is received at the application layer and passed to the order service layer. The order service receives the request and creates the order for the related user. Then the payment domain event is triggered and the payment operation are executed. Unlike the original example, the payment related execution operations are carried out in the order service. This leads to reduced number of entry and exit operations. As a result, the CFP for checkout in eShopBig is 15, whereas it was 17 for eShopOriginal.

Table 5.41 eShopBig “Checkout” CFP calculation

FUR	FP	FP Name	DM Type	Data Group Desc. In App. Layer	Data Group Desc. In Domain Layer	Service/Application	CFP
1	9	Checkout	E	Checkout request		Application Layer	1
			X	Create order		Application Layer	1
			E		Create order	Order Service	1
			R		Buyer	Order Service	1
			R		Order	Order Service	1
			W		Order	Order Service	1
			R		Payment Info	Order Service	1
			W		Pay	Order Service	1
			X		Error/Confirmation	Order Service	1
			E		Delete basket item	Basket Service	1
			R		Basket	Basket Service	1
			R		Basket Item	Basket Service	1
			W		Basket	Basket Service	1
			X		Error/Confirmation	Basket Service	1
			X	Error/Confirmation		Application Layer	1
TOTAL CFP for Checkout							15

The COSMIC function point calculation for eShopBig application can be found in Table 5.42. The total number of CFP is 116 for eShopBig application. When compared to the original example, it is less costly to write eShopBig application.

Table 5.42 Total CFP of eShopBig

FP Name	CFP
Login	6
List Items for Sale	8
Filter Based on Brand	8
Filter Based on Type	8
Add Item to a Basket	7
List Items in the Basket	7
Update Item in the Basket	8
Delete Item from the Basket	8
Checkout	15
Get Buyer Profile	6
Update Buyer Profile	7
Update Buyer Password	7
List Orders	6
Get Order Item Detail	6
Logout	2
Update Marketing	7
TOTAL	**116**

5.1.2.4. Summary of CFP Measurements for the eShop Case Study

As seen in Figure 5.2, the more granular eShopOnContainers application has the highest COSMIC function point value. The next application in eShop case study with the average COSMIC function point value is the original application. The least COSMIC function value belongs to the less granular eShop application.

These results indicate that the most complex eShop application is the more granular one. There are more number of microservices and data is passed through those different layers. When the application has less number of microservices or layers, then the complexity and relatively the cost of implementing the application is reduced.

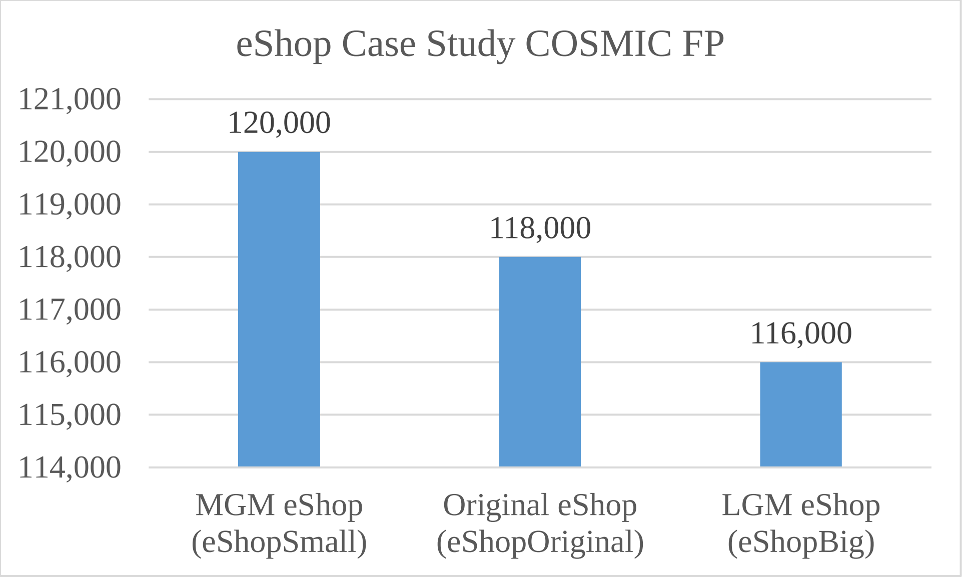

Figure 5.2 CFP results for eShop case study

5.2. Coupling Measurements

The coupling measurement for eShopOriginal, eShopSmall, and eShopBig applications were carried out using an automated tool called NDepend [109]. The coupling measurements for original cargo application, MGM cargo application, and LGM cargo application are carried out manually. In order to measure the afferent coupling the incoming associations to the related bounded context is measured. The non-directional association is considered as a two-way association. The efferent coupling is measured as the number of outgoing associations for the related bounded context.

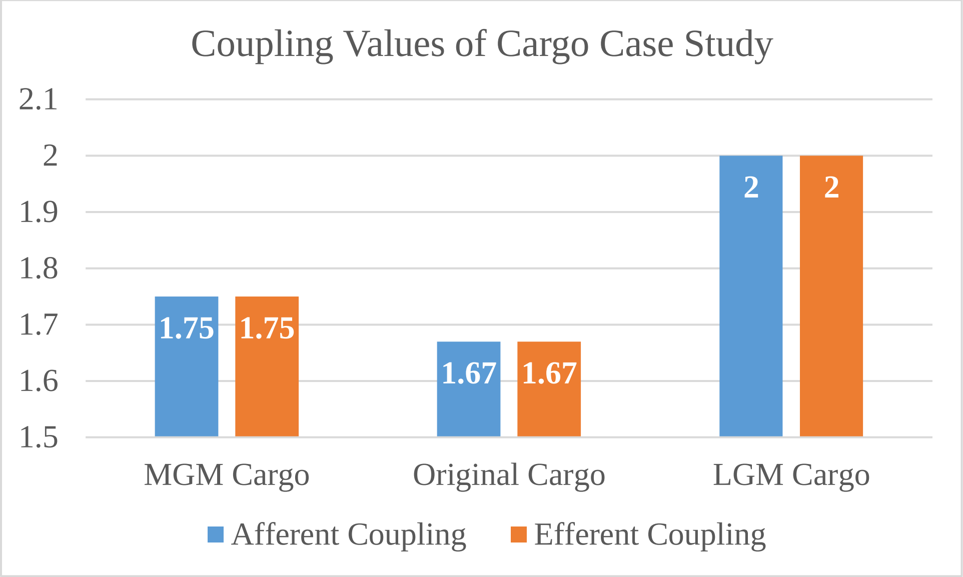

Figure 5.3 Coupling values of cargo case study

As seen in Figure 5.3, the afferent and efferent coupling values are the same for each application. The original cargo application has the best coupling values, followed by the more granular example. The least granular application had the worst coupling values.

The original eShopOnContainers application has the best coupling values (Figure 5.4). It is followed by MGM (eShopSmall) then LGM (eShopBig) application.

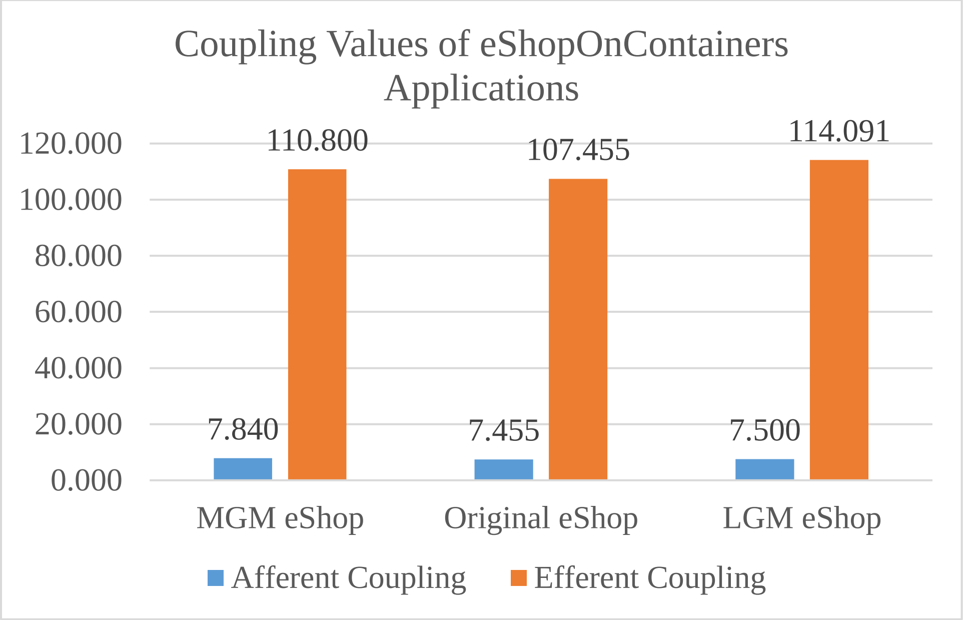

Figure 5.4 Coupling values of eShopOnContainers case study

5.3. Cohesion Measurements

The relational cohesion measurement for eShopOriginal, eShopSmall, and eShopBig applications were carried out using an automated tool called NDepend [109]. The relational cohesion measurement for original cargo application, MGM cargo application, and LGM cargo application are carried out manually.

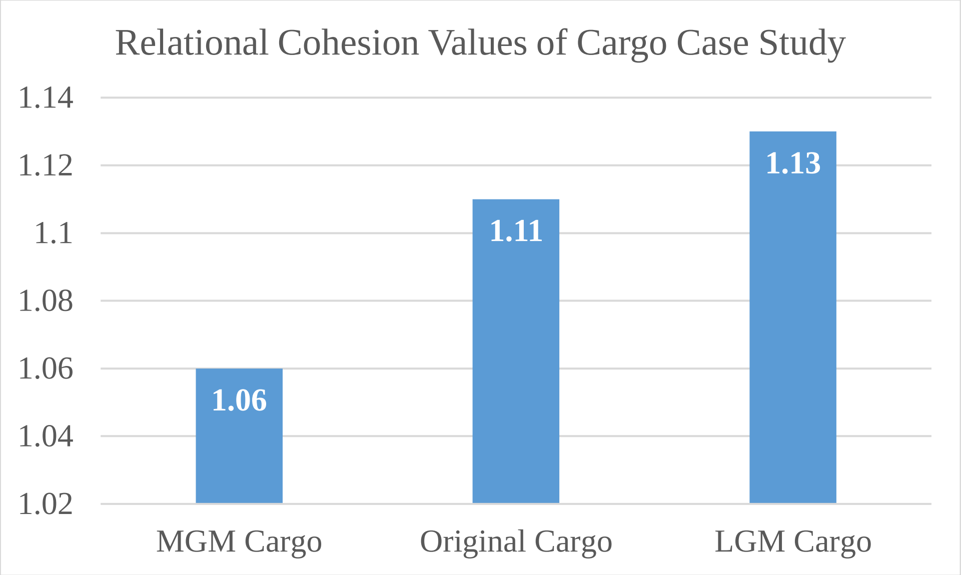

Figure 5.5 Cohesion values of cargo case study

As show in Figure 5.5, the best cohesion value is observed in LGM cargo application. It is followed by the original cargo application. The worst relational cohesion value is observed in MGM cargo application.

A similar pattern is observed for the eShopOnContainers applications as well – see Figure 5.6. The best relational cohesion value is seen in LGM eShopOnContainer application (eShopBig). It is closely followed by the original eShopOnContainers application. The worst relational cohesion numbers are seen in MGM eShopOnContainers application (eShopSmall).

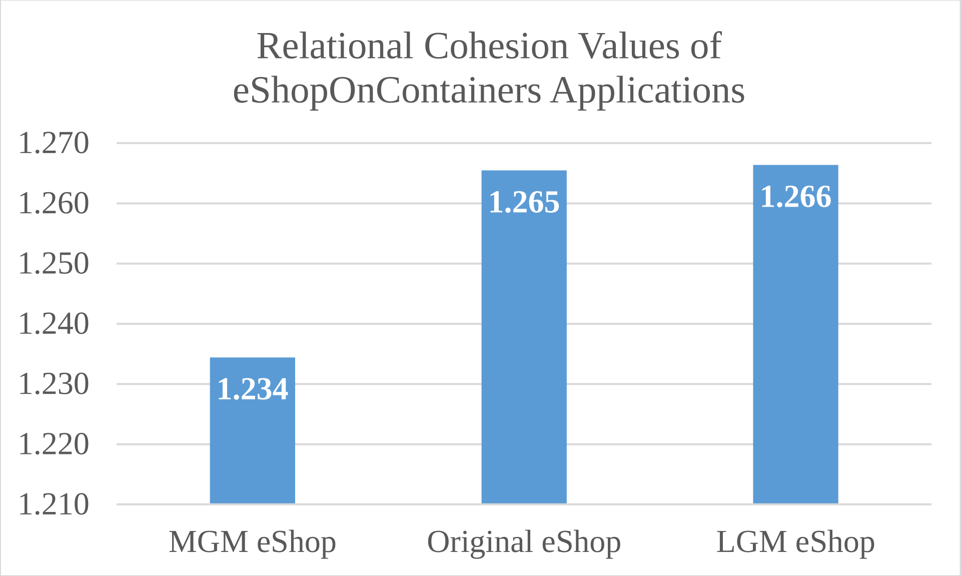

Figure 5.6 Cohesion values of eShopOnContainers case study

5.4. Analysis of Measurements

The collection of measurements for the cargo applications can be seen in Table 5.43. The original case study has the smallest coupling values. The LGM cargo case study has the largest cohesion value and the smallest COSMIC function point value.

Table 5.43 Measurements for cargo case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC FP**
MGM Cargo	1.75	1.75	1.06	71
Original Cargo	1.67	1.67	1.11	63
LGM Cargo	2	2	1.13	49

The values are normalized in such a way that all the measurements get a value between 0 and 1. The normalization technique used is called min-max scalar normalization [111] [112]. Given a set of matching scores $\{s_k\}$, $k = 1, 2, \ldots, n$, the normalized scores are given by Equation (5). Where, min denotes the minimum value in the dataset and max denotes the maximum value in the dataset.

$$s'_k = \frac{s_k - min}{max - min} \qquad (5)$$

The most desirable value for that column gets 1, the least desirable one gets 0. For example, relational cohesion is expected to be higher. As a result, LGM cargo application having the largest relational cohesion value gets 1. The MGM cargo application has the least relational cohesion, so it gets 0. The original cargo application has a value in between, closer to the desired side. After applying the normalization formula, based on direct proportionality, it gets a value of 0.71. The normalized numbers for each measurement is shown in Table 5.44.

Table 5.44 Normalized measurements for cargo case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC FP**
MGM Cargo	0.76	0.76	0	0
Original Cargo	1	1	0.71	0.50
LGM Cargo	0	0	1	1

In order to find the optimum result, we applied a multi attribute decision making process called Analytic Hierarchy Process (AHP) method [110]. As stated in [113], unless the importance of each measured aspect can be identified, they are treated equally. This principle is called the principle of equal importance of features. As a result, all the weights are given equally while applying AHP. In order to calculate AHP values, an online calculator was utilized [114].

Table 5.45 AHP results for cargo case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC FP**	**Weighted Sum**	**Rank**
Criteria Weight	0.25	0.25	0.25	0.25	**-**	-
MGM Cargo	0.76	0.76	0	0	0.380	3
Original Cargo	1	1	0.71	0.50	0.802	1
LGM Cargo	0	0	1	1	0.500	2

As seen in Table 5.45, the most desirable result is achieved by the original cargo case study (having rank 1). The next desirable solution is LGM cargo. The least optimal one is MGM cargo.

The plain numbers for eShopSmall, eShopOriginal, and eShopBig are listed in Table 5.46. These values were normalized to get a number between 0 and 1 based on the desirability. The same calculation methods used for the cargo applications are followed for eShopOnContainers applications as well.

Table 5.46 Measurements for eShopOnContainers case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC FP**
MGM eShop (eShopSmall)	7.840	110.800	1.234	120.000
Original eShop (eShopOriginal)	7.455	107.455	1.265	118.000
LGM eShop (eShopBig)	7.500	114.091	1.266	116.000

The normalized numbers for eShopSmall, eShopOriginal, and eShopBig can be seen in Table 5.47. These values were normalized to get a number between 0 and 1 based on the desirability.

Table 5.47 Normalized measurements for eShopOnContainers case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC FP**
MGM eShop (eShopSmall)	0	0.51	0	0
Original eShop (eShopOriginal)	1	1	0.97	0.5
LGM eShop (eShopBig)	0.88	0	1	1

As shown in Table 5.48, the optimum result is achieved by the original eShop cases study. It is closely followed by LGM eShop.

Table 5.48 AHP results for eShop case study

	Afferent Coupling	**Efferent Coupling**	**Relational Cohesion**	**COSMIC CFP**	**Weighted Sum**	**Rank**
Criteria Weight	0.25	0.25	0.25	0.25	**-**	-
MGM eShop	0	0.51	0	0	0.128	3
Original eShop	1	1	0.97	0.50	0.867	1
LGM eShop	0.88	0	1	1	0.720	2

As mentioned before, four different measurements were collected. Out of those measurements, the expected behavior for a definition of good design is explained below:

- COSMIC Function Points: The higher the COSMIC function points is, the more complex the application will be.
- Afferent Coupling: The afferent coupling values are expected to be lower.
- Efferent Coupling: The afferent coupling values are expected to be lower.
- Relational Cohesion: Cohesion is expected to be higher.

In Chapter 1, we laid out the research questions. Based on the measurement results, those research questions are answered below:

- RQ1: Is Microservices Architecture a trending topic?
  As seen in Figure 3.11, the microservices related research had an upward trend in 2016. When the same query operation was carried out in 2020, we see a decrease in the last year (Figure 3.12). This might be due to some research not being indexed by web of science database as well. However, the same search was carried out on the same month of 2017. As we do not observe a fall in the number of research papers within the last year in Figure 3.11, we doubt that the same search in 2020 might be affected due to delayed indexing in database.
- RQ2: Is there a relation between the granularity and the coupling of microservices?
  No. Both MGM and LGM had more coupling values when compared to the original. One of the biggest benefits of Microservices Architecture is reducing coupling. Moving from monolith to smaller services is expected to bring less coupling [20]. Some people call microservices as SOA done right [115]. However, as seen in Table 5.45 and Table 5.48, smaller microservices do not always mean less coupled code. According to these results, there is a sweet spot for finding the optimal coupling value for a service.
- RQ3: Is there a relation between the granularity and the cohesion of microservices?
  In the covered case studies, it looks like the cohesion increased by the increase in microservice size. However, just two case studies are not enough to have a definitive answer. If all the services were to be collected in one monolithic service, this could affect the cohesion in a negative way. More case studies should be carried out before having a definite conclusion.
- RQ4: Can optimum granularity of a microservice be calculated by minimum coupling value and the largest cohesion value?
  As seen in Table 5.45 and Table 5.48 the coupling values are the best in the original example. The cohesion value is the best in More Granular Microservice example. However, the increase in the cohesion is not very noticeable when compared to the small decrease in coupling in Table 5.48. So,

it is possible to assume that the optimal point for the granularity is the original example, which has a desired loose coupling and acceptable cohesion values. The optimum granularity point can be seen as the right place to identify the bounded context. Identifying the right bounded context has been a challenge when using DDD [116]. The usage of coupling and cohesion values can be used for identifying the right bounded context in DDD.

- RQ5: Is COSMIC function a good fit measuring the size of microservices?
  Before starting this thesis, we expected the less granular microservices to be less costly and more granular services to be more costly. The results we saw made it obvious that if the functional user requirement is already complex (passing multiple boundaries/layers) the effect of dividing a service into smaller pieces makes it complex but the difference is mostly around 2 CFP. However, when the functional user requirement is not complex (carried out within one bounded context/service), adding an extra service introduces more complexity – mostly around 4 CFP. Using COSMIC function points as means of measurement of microservices is a viable method.
- RQ6: Does domain driven design help identifying the optimum microservice?
  The case studies, we have carried out, had the best result in the original applications. The domain-driven design was used for both of the original examples. As a result, it is safe to say that domain driven design leads to the optimum level of granularity.
- RQ7: Does microservice size measurement help finding the optimum size?
  The main debate around “Does size matter?” should not be around the size. For the cargo example the most complex application has COSMIC CFP of 71. On the other hand, the least complex example of eShopOnContainers had COSMIC CFP of 116. Instead of focusing on the size, focusing on the context boundaries and applying domain-driven design gives us the optimal microservice size.

CHAPTER 6

CONCLUSION

The introduction of Microservice Architecture raised questions around what makes it different from Service Oriented Architecture. Does the word “micro” refer to having smaller sized services? This led to an ongoing debate around “Does the size of a microservice matter?” Some suggested that the important factor is not the size but rather the context of a microservice. The goal of this thesis is to research the factors on finding the optimum granularity of microservices.

This thesis study, laid out six different research questions around the factors for identifying the ones which affect the optimum granularity of microservices. In order to answer the research questions, two different examples were studied on. The first example is a UML class diagram for a cargo application which was generated by applying domain-driven design concepts [26]. The second example was an open source online shopping application constructed by using domain-driven design [27]. Each example was originally generated by applying domain-driven design concepts. As part of this thesis, both examples were modified to generate more granular and less granular services. In the end the original example, the more granular and the less granular examples are all measured for the following metrics:

- Afferent coupling
- Efferent coupling
- Relational cohesion
- COSMIC function points

Each measurement was normalized to get a number between 0 and 1 based on the desirability of the actual behavior. The top performing result received 1 and the low performing result got 0. The other measurements get the corresponding score based on their performance on desirability.

The measurement results showed that dividing an existing service into smaller services does not necessarily result in less coupling. In our measurements, we observed that the cohesion increases when the service gets bigger. However, we didn’t explore the limits of that theory. More research should be carried out before generalizing the result.

We concluded that finding the optimal coupling and cohesion values lead to the optimum service granularity. In both case studies studied as part of this thesis, the original example generated by domain-driven design lead to the optimum granularity of the microservices.

The COSMIC function points are good for measuring the microservices. They give a good idea about the complexity of each microservice as well as the overall system.

It turns out that depending on the functional user requirement, services end up having a wide variety of COSMIC function point sizes. So the important factor for identifying the optimum granularity cannot be based on the size, but it should rather be based on the domain context.

Although just two case studies are not enough, it is a good starting point to elaborate in the future. Improvements can be made to the proposed method by investigating whether measurements on asynchronous eventing behavior will be useful or not. For example, eShopOnContainers example have integration events (across microservices) and domain events (within a microservice). These events are not reflected in the static code analysis techniques. Because when the event is triggered, the Rabbit MQ is called. The other microservice registers to that particular event. So Rabbit MQ triggers the other microservice. There is no direct call from one microservice to another. As a result such eventing behavior cannot be identified in static code analysis tools. These events may account for measuring coupling and cohesion values from an asynchronous microservices perspective. For the future studies, the existing method is planned to be expanded by adding measurements for asynchronous eventing behavior.

In the upcoming studies, we plan to explore different approaches focusing on identifying the right bounded context at design time. In order to keep up with the ever-changing requirements of the customers and the environment, the code in the cloud needs to change at a fast pace. Having the right design from the beginning is not enough anymore. The granularity and efficiency of microservices need to change over time. A monitoring tool for measuring microservice granularity metrics over time and giving suggestions for code refactoring would be a beneficial for everyone who has adopted the microservices architecture. The tool can also incorporate machine learning algorithms targeting cluster identification.

REFERENCES

[1] H. Vural, M. Koyuncu, and S. Guney, “A Systematic Literature Review on Microservices,” Springer, Cham, 2017, pp. 203–217.

[2] “Beyond agile: Reorganizing IT for faster software delivery | McKinsey.” [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/beyond-agile-reorganizing-it-for-faster-software-delivery. [Accessed: 28-Dec-2019].

[3] “Amazon Compute Service Level Agreement.” [Online]. Available: https://aws.amazon.com/compute/sla/. [Accessed: 28-Dec-2019].

[4] J. Lewis and M. Fowler, “Microservices,” 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed: 10-Oct-2019].

[5] S. Eski and F. Buzluca, “An automatic extraction approach - Transition to microservices architecture from monolithic application,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1477, pp. 1–6, 2018, doi: 10.1145/3234152.3234195.

[6] “Don’t Share Libraries among Microservices.” [Online]. Available: https://phauer.com/2016/dont-share-libraries-among-microservices/. [Accessed: 29-Dec-2019].

[7] “5 Major Benefits of Microservice Architecture - Skelia :: Skelia.” [Online]. Available: https://skelia.com/articles/5-major-benefits-microservice-architecture/. [Accessed: 28-Dec-2019].

[8] “Exceeding the SLA–Its about resilience | Brent’s Notepad.” [Online]. Available: https://brentdacodemonkey.wordpress.com/2012/04/10/exceeding-the-slaits-about-resilience/. [Accessed: 29-Dec-2019].

[9] M. Wasson, “Building microservices on Azure | Microsoft Docs,” 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/microservices/. [Accessed: 30-Dec-2019].

[10] D. Lebrero, “The LOC is often used to measure complexity, and it is often right. Not for c... — DEV.” [Online]. Available: https://dev.to/bousquetn/comment/b4o. [Accessed: 29-Dec-2019].

[11] P. Hauer, “Microservices in a Nutshell. Pros and Cons.,” 2015. [Online]. Available: https://phauer.com/2015/microservices-nutshell-pros-cons/. [Accessed: 29-Dec-2019].

[12] H. Vural, M. Koyuncu, and S. Guney, “A Systematic Literature Review on Microservices,” Springer, Cham, 2017, pp. 203–217.

[13] S. Carey, “The pros and cons of microservices: Lessons learned as Comparethemarket.com moves away from a single, monolithic application | Computerworld,” 2016. [Online]. Available: https://www.computerworld.com/article/3427177/the-pros-and-cons-of-microservices--lessons-learned-as-comparethemarket-com-moves-away-from-a-single.html. [Accessed: 30-Dec-2019].

[14] P. Wittmer, “Microservices Disadvantages & Advantages - Tiempo Development [2019] - Pitfalls of Microservices Architecture,” 2019. [Online]. Available: https://www.tiempodev.com/blog/disadvantages-of-a-microservices-architecture/. [Accessed: 30-Dec-2019].

[15] J. Soldani, D. A. Tamburri, and W. J. Van Den Heuvel, “The pains and gains of microservices: A Systematic grey literature review,” *J. Syst. Softw.*, vol. 146, pp. 215–232, Dec. 2018, doi: 10.1016/j.jss.2018.09.082.

[16] W. Łabaj, “Goodbye microservices, hello right-sized services,” *Particular Software*, 2015. [Online]. Available: https://particular.net/blog/goodbye-microservices-hello-right-sized-services. [Accessed: 29-Dec-2019].

[17] D. Belcham, “Microservice Sizing,” *Western Devs*, 2015. [Online]. Available: https://westerndevs.com/microservices-sizing/. [Accessed: 29-Dec-2019].

[18] R. Li, “Microservices: It’s not the size that matters, but the scope | @datawireio,” 2016. [Online]. Available: https://www.datawire.io/microservices-not-size-matters-scope/. [Accessed: 30-Dec-2019].

[19] M. Plöd, “Microservices love Domain Driven Design, why and how?,” *innoQ*, 2017. [Online]. Available: https://speakerdeck.com/mploed/microservices-love-domain-driven-design-why-and-how. [Accessed: 29-Dec-2019].

[20] V. Singh and S. K. Peddoju, “Container-based microservice architecture for cloud applications,” *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2017*, vol. 2017-Janua, pp. 847–852, 2017, doi: 10.1109/CCAA.2017.8229914.

[21] Z. Ren *et al.*, “Migrating web applications from monolithic structure to microservices architecture,” *ACM Int. Conf. Proceeding Ser.*, 2018, doi: 10.1145/3275219.3275230.

[22] “Software sizing: Function Point methods to measure software sizes,” 2019. [Online]. Available: https://www.estimancy.com/en/2019/09/30/software-sizing-if-you-cant-measure-it-you-cant-improve-it/. [Accessed: 30-Dec-2019].

[23] H. Vural, M. Koyuncu, and S. Misra, *A case study on measuring the size of microservices*, vol. 10964 LNCS. 2018.

[24] P. Koss, “Evaluating and improving a scenario-based analysis method for service-based systems,” Universität Stuttgart, 2019.

[25] P. Morris, “COSMIC-FFP and IFPUG 4.1 Similarities and Differences,” 2009.

[26] F. Rademacher, J. Sorgalla, and S. Sachweh, “Challenges of domain-driven microservice design: A model-driven perspective,” *IEEE Softw.*, vol. 35, no. 3, pp. 36–43, May 2018, doi: 10.1109/MS.2018.2141028.

[27] C. de la Torre, B. Wagner, and M. Rousos, *.NET Microservices Architecture for Containerized NET Applications*, 2.2.1. Redmond: Microsoft Developer Division, 2019.

[28] I. Sommerville, *Software Engineering (10th Edition)*. Pearson Education Limited, 2015.

[29] S. Newman and S. Newman, *Building microservices : designing fine-grained systems*. O’Reilly Media, 2015.

[30] B. W. Boehm, *Software engineering economics*. Prentice-Hall, 1981.

[31] C. Jones, *Programming Productivity*. New York: McGraw-Hill, 1986.

[32] "Fundamentals & Introduction of Function Point Analysis." [Online]. Available: http://www.softwaremetrics.com/fpafund.htm. [Accessed: 13-Jun-2017].

[33] Z. Yinhuan, W. Beizhan, Z. Yilong, and S. Liang, "Estimation of software projects effort based on function point," in *Proceedings of 2009 4th International Conference on Computer Science and Education, ICCSE 2009*, 2009, pp. 941–943, doi: 10.1109/ICCSE.2009.5228317.

[34] C. Jones, "Strengths and weaknesses of software metrics." [Online]. Available: http://www.compaid.com/caiinternet/ezine/capers-StrngWk.pdf. [Accessed: 13-Jun-2017].

[35] "Counting Practices Manual EN v2.1 - Nesma." [Online]. Available: https://nesma.org/downloads/countingpracticesmanual-en-v21/?highlight=ifpug cpm 4.2. [Accessed: 29-Dec-2019].

[36] R. Dumke and A. Abran, *COSMIC function points : theory and advanced practices*. CRC Press, 2011.

[37] L. Lavazza and G. Robiolo, "Introducing the evaluation of complexity in functional size measurement: A UML-based approach," in *ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010, doi: 10.1145/1852786.1852820.

[38] C. R. Symons and A. Lesterhuis, "Measurement Manual 4.0.1 - COSMIC," 2017. [Online]. Available: https://cosmic-sizing.org/publications/measurement-manual-401/. [Accessed: 29-Dec-2019].

[39] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Web Effort Estimation: Function Point Analysis vs. COSMIC," *Inf. Softw. Technol.*, vol. 72, pp. 90–109, Apr. 2016, doi: 10.1016/j.infsof.2015.12.001.

[40] "WikiPedia - Domain-driven design." [Online]. Available: https://en.wikipedia.org/wiki/Domain-driven_design. [Accessed: 18-Dec-2019].

[41] E. Evans, *Domain-driven design : tackling complexity in the heart of software*. Addison-Wesley, 2004.

[42] E. Evans, *Domain-driven design reference : definitions and patterns summaries*. 2014.

[43] E. Evans, "DDD and Microservices: At Last, Some Boundaries!," 2016. [Online]. Available: https://www.infoq.com/presentations/ddd-microservices-2016/. [Accessed: 18-Dec-2019].

[44] S. Arshed, "Monolithic vs SOA vs Microservices — How to Choose Your Application Architecture," 2018. [Online]. Available: https://medium.com/@saad_66516/monolithic-vs-soa-vs-microservices-how-to-choose-your-application-architecture-1a33108d1469. [Accessed: 05-Jan-2020].

[45] J.´e I. Fern´andez-Villamor, C.´A. Iglesias, and Garijo, "Microservices - Lightweight Service Descriptions for REST Architectural Style," in *2nd International Conference on Agents and Artificial Intelligence (ICAART)*, 2010.

[46] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proceedings - 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA 2016*, 2016, pp. 44–51, doi: 10.1109/SOCA.2016.15.

[47] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *12th International Conference on Evaluation and Assessment in Software Engineering*, 2008, pp. v17, p1.

[48] Clarivate Analytics, “About us - Clarivate Analytics,” 2019. [Online]. Available: https://apps.webofknowledge.com. [Accessed: 10-Oct-2019].

[49] O. Lysne, K. J. Hole, C. Otterstad, O. Ytrehus, R. Aarseth, and J. Tellnes, “Vendor Malware: Detection Limits and Mitigation,” *Computer (Long. Beach. Calif).*, vol. 49, no. 8, pp. 62–69, Aug. 2016, doi: 10.1109/MC.2016.227.

[50] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, and V. Sekar, “Gremlin: Systematic Resilience Testing of Microservices,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2016-Augus, pp. 57–66, 2016, doi: 10.1109/ICDCS.2016.11.

[51] M. Villamizar *et al.*, “Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures,” in *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, 2016, pp. 179–182, doi: 10.1109/CCGrid.2016.37.

[52] M. Villamizar *et al.*, “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud,” in *2015 10th Colombian Computing Conference, 10CCC 2015*, 2015, pp. 583–590, doi: 10.1109/ColumbianCC.2015.7333476.

[53] Y. Sun, S. Nanda, and T. Jaeger, “Security-as-a-service for microservices-based cloud applications,” in *Proceedings - IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015*, 2016, pp. 50–57, doi: 10.1109/CloudCom.2015.93.

[54] M. Rahman and J. Gao, “A reusable automated acceptance testing architecture for microservices in behavior-driven development,” in *Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015*, 2015, vol. 30, pp. 321–325, doi: 10.1109/SOSE.2015.55.

[55] V. D. Le *et al.*, “Microservice-based architecture for the NRDC,” in *Proceeding - 2015 IEEE International Conference on Industrial Informatics, INDIN 2015*, 2015, pp. 1659–1664, doi: 10.1109/INDIN.2015.7281983.

[56] S. Alpers, C. Becker, A. Oberweis, and T. Schuster, “Microservice based tool support for business process modelling,” in *Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOCW 2015*, 2015, pp. 71–78, doi: 10.1109/EDOCW.2015.32.

[57] P. Bak, R. Melamed, D. Moshkovich, Y. Nardi, H. Ship, and A. Yaeli, “Location and Context-Based Microservices for Mobile and Internet of Things Workloads,” in *Proceedings - 2015 IEEE 3rd International Conference on Mobile Services, MS 2015*, 2015, pp. 1–8, doi: 10.1109/MobServ.2015.11.

[58] D. Malavalli and S. Sathappan, “Scalable microservice based architecture for enabling DMTF profiles,” in *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*, 2015, pp. 428–432, doi: 10.1109/CNSM.2015.7367395.

[59] A. Krylovskiy, M. Jahn, and E. Patti, “Designing a Smart City Internet of Things Platform with Microservice Architecture,” in *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015*

International Conference on Open and Big Data, OBD 2015, 2015, pp. 25–30, doi: 10.1109/FiCloud.2015.55.

[60] A. Ciuffoletti, "Automated Deployment of a Microservice-based Monitoring Infrastructure," in *Procedia Computer Science*, 2015, vol. 68, pp. 163–172, doi: 10.1016/j.procs.2015.09.232.

[61] K. Meinke and P. Nycander, "Learning-based testing of distributed microservice architectures: Correctness and fault injection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9509, pp. 3–10, doi: 10.1007/978-3-662-49224-6_1.

[62] C. Pahl and P. Jamshidi, "Software architecture for the cloud – A roadmap towards control-theoretic, model-based cloud architecture," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9278, pp. 212–220, doi: 10.1007/978-3-319-23727-5_17.

[63] P. Nicolaescu and R. Klamma, "A methodology and tool support for widget-based web application development," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9114, pp. 515–532, doi: 10.1007/978-3-319-19890-3_33.

[64] I. Koren, P. Nicolaescu, and R. Klamma, "Collaborative drawing annotations on Web videos," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9114, pp. 671–674, doi: 10.1007/978-3-319-19890-3_54.

[65] E. Braun, C. Düpmeier, D. Kimmig, W. Schillinger, and K. Weissenbach, "Generic Web Framework for Environmental Data Visualization," 2017, pp. 289–299.

[66] D. S. Linthicum, "Practical Use of Microservices in Moving Workloads to the Cloud," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 6–9, 2016, doi: 10.1109/MCC.2016.114.

[67] T. Inagaki, Y. Ueda, and M. Ohara, "Container management as emerging workload for operating systems," *Proc. 2016 IEEE Int. Symp. Workload Charact. IISWC 2016*, pp. 65–74, 2016, doi: 10.1109/IISWC.2016.7581267.

[68] T. Ueda, T. Nakaike, and M. Ohara, "Workload characterization for microservices," in *Proceedings of the 2016 IEEE International Symposium on Workload Characterization, IISWC 2016*, 2016, pp. 85–94, doi: 10.1109/IISWC.2016.7581269.

[69] L. Florio and E. Di Nitto, "Gru: An approach to introduce decentralized autonomic behavior in microservices architectures," in *Proceedings - 2016 IEEE International Conference on Autonomic Computing, ICAC 2016*, 2016, pp. 357–362, doi: 10.1109/ICAC.2016.25.

[70] C. Gadea, M. Trifan, D. Ionescu, and B. Ionescu, "A reference architecture for real-time microservice API consumption," in *3rd Workshop on CrossCloud Infrastructures and Platforms, CrossCloud 2016 - Colocated with EuroSys 2016*, 2016, doi: 10.1145/2904111.2904115.

[71] J. Renz, D. Hoffmann, T. Staubitz, and C. Meinel, "Using A/B testing in MOOC environments," in *ACM International Conference Proceeding Series*, 2016, vol. 25-29-April-2016, pp. 304–313, doi: 10.1145/2883851.2883876.

[72] W. Hasselbring, "Microservices for scalability: [Keynote talk abstract]," *ICPE 2016 -*

Proc. 7th ACM/SPEC Int. Conf. Perform. Eng., pp. 133–134, 2016, doi: 10.1145/2851553.2858659.

[73] W. Scarborough, C. Arnold, and M. Dahan, “Case study: Microservice evolution and software lifecycle of the XSEDE user portal API,” in *ACM International Conference Proceeding Series*, 2016, vol. 17-21-July-2016, doi: 10.1145/2949550.2949655.

[74] G. Kecskemeti, A. C. Marosi, and A. Kertesz, “The ENTICE approach to decompose monolithic services into microservices,” in *2016 International Conference on High Performance Computing and Simulation, HPCS 2016*, 2016, pp. 591–596, doi: 10.1109/HPCSim.2016.7568389.

[75] O. Barais, J. Bourcier, Y. D. Bromberg, and C. Dion, “Towards microservices architecture to transcode videos in the large at low costs,” in *2016 International Conference on Telecommunications and Multimedia, TEMU 2016*, 2016, pp. 69–74, doi: 10.1109/TEMU.2016.7551918.

[76] H. Kang, M. Le, and S. Tao, “Container and microservice driven design for cloud infrastructure DevOps,” in *Proceedings - 2016 IEEE International Conference on Cloud Engineering, IC2E 2016: Co-located with the 1st IEEE International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, 2016, pp. 202–211, doi: 10.1109/IC2E.2016.26.

[77] A. Messina, R. Rizzo, P. Storniolo, M. Tripiciano, and A. Urso, “The database-is-the-service pattern for microservice architectures,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9832 LNCS, pp. 223–233, doi: 10.1007/978-3-319-43949-5_18.

[78] S. Hassan and R. Bahsoon, “Microservices and their design trade-offs: A self-adaptive roadmap,” in *Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016*, 2016, pp. 813–818, doi: 10.1109/SCC.2016.113.

[79] J. Bogner and A. Zimmermann, “Towards Integrating Microservices with Adaptable Enterprise Architecture,” in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*, 2016, vol. 2016-September, pp. 158–163, doi: 10.1109/EDOCW.2016.7584392.

[80] N. Kratzke and R. Peinl, “ClouNS - A Cloud-native Application Reference Model for Enterprise Architects,” Sep. 2017, doi: 10.1109/EDOCW.2016.7584353.

[81] T. Thiele, T. Sommer, S. Stiehm, S. Jeschke, and A. Richert, “Exploring research networks with data science: A data-driven microservice architecture for synergy detection,” in *Proceedings - 2016 4th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud 2016*, 2016, pp. 246–251, doi: 10.1109/W-FiCloud.2016.58.

[82] S. Qanbari *et al.*, “IoT design patterns: Computational constructs to design, build and engineer edge applications,” in *Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, 2016, pp. 277–282, doi: 10.1109/IoTDI.2015.18.

[83] D. Guo, W. Wang, G. Zeng, and Z. Wei, “Microservices architecture based cloudware deployment platform for service computing,” in *Proceedings - 2016 IEEE Symposium on Service-Oriented System Engineering, SOSE 2016*, 2016, pp. 358–364, doi: 10.1109/SOSE.2016.22.

[84] L. Safina, M. Mazzara, F. Montesi, and V. Rivera, “Data-driven workflows for

microservices: Genericity in jolie," in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 2016, vol. 2016-May, pp. 430–437, doi: 10.1109/AINA.2016.95.

[85] N. Kratzke, "About Microservices, Containers and their Underestimated Impact on Network Performance," Sep. 2017.

[86] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives," *J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 2918–2933, 2014, doi: 10.1016/j.jpdc.2014.06.007.

[87] "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," doi: 10.1007/s00766-005-0021-6.

[88] F. Rademacher, S. Sachweh, and A. Zundorf, "Differences between model-driven development of service-oriented and microservice architecture," *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 38–45, 2017, doi: 10.1109/ICSAW.2017.32.

[89] F. Rademacher, J. Sorgalla, P. N. Wizenty, S. Sachweh, and A. Zündorf, "Microservice architecture and model-driven development: Yet singles, Soon Married (?)," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1477, no. i, 2018, doi: 10.1145/3234152.3234193.

[90] F. Rademacher, S. Sachweh, and A. Zundorf, "Aspect-oriented modeling of technology heterogeneity in microservice architecture," *Proc. - 2019 IEEE Int. Conf. Softw. Archit. ICSA 2019*, pp. 21–30, 2019, doi: 10.1109/ICSA.2019.00011.

[91] W. Hasselbring and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 243–246, 2017, doi: 10.1109/ICSAW.2017.11.

[92] A. Carrasco, B. Van Bladel, and S. Demeyer, "Migrating towards microservices: Migration and architecture smells," in *IWoR 2018 - Proceedings of the 2nd International Workshop on Refactoring, co-located with ASE 2018*, 2018, pp. 1–6, doi: 10.1145/3242163.3242164.

[93] H. Vural, M. Koyuncu, and S. Guney, *A systematic literature review on microservices*, vol. 10409 LNCS. 2017.

[94] S. Sarkar, G. Vashi, and P. P. Abdulla, "Towards Transforming an Industrial Automation System from Monolithic to Microservices," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2018-Septe, pp. 1256–1259, 2018, doi: 10.1109/ETFA.2018.8502567.

[95] A. Tarmizi and M. Shanudin, "A Method for Analyzing and Designing Microservice Holistically," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 12, pp. 281–287, 2017, doi: 10.14569/ijacsa.2017.081236.

[96] R. Hill, D. Shadija, and M. Rezai, "Enabling community health care with microservices," *Proc. - 15th IEEE Int. Symp. Parallel Distrib. Process. with Appl. 16th IEEE Int. Conf. Ubiquitous Comput. Commun. ISPA/IUCC 2017*, pp. 1444–1450, 2018, doi: 10.1109/ISPA/IUCC.2017.00220.

[97] G. Mazlami, J. Cito, and P. Leitner, "Extraction of Microservices from Monolithic Software Architectures," *Proc. - 2017 IEEE 24th Int. Conf. Web Serv. ICWS 2017*, pp. 524–531, 2017, doi: 10.1109/ICWS.2017.61.

[98] M. Kamimura, K. Yano, T. Hatano, and A. Matsuo, “Extracting Candidates of Microservices from Monolithic Application Code,” *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2018-Decem, pp. 571–580, 2019, doi: 10.1109/APSEC.2018.00072.

[99] A. T. A. Ghani and M. S. Zakaria, “Method for designing scalable microservice-based application systematically: A case study,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 125–135, 2018, doi: 10.14569/IJACSA.2018.090817.

[100] S. Klock, J. M. E. M. Van Der Werf, J. P. Guelen, and S. Jansen, “Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures,” *Proc. - 2017 IEEE Int. Conf. Softw. Archit. ICSA 2017*, pp. 11–20, 2017, doi: 10.1109/ICSA.2017.38.

[101] L. Baresi, M. Garriga, and A. De Renzis, “Microservices Identification Through Interface Analysis,” Springer, Cham, 2017, pp. 19–33.

[102] P. Jamshidi, C. Pahl, and N. C. Mendonça, “Pattern-based multi-cloud architecture migration,” *Softw. - Pract. Exp.*, vol. 47, no. 9, pp. 1159–1184, 2017, doi: 10.1002/spe.2442.

[103] S. A. K. Ghayyur, A. Razzaq, S. Ullah, and S. Ahmed, “Matrix clustering based migration of system application to microservices architecture,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 1, pp. 284–296, 2018, doi: 10.14569/IJACSA.2018.090139.

[104] F. Rademacher, S. Sachweh, and A. Zündorf, “Towards a UML profile for domain-driven design of microservice architectures,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10729 LNCS, pp. 230–245, doi: 10.1007/978-3-319-74781-1_17.

[105] F. Melo, “eShopOnContainers.” [Online]. Available: https://github.com/dotnet-architecture/eShopOnContainers. [Accessed: 13-Oct-2019].

[106] H. Vural, “hulyav/eShopContainersSmall,” 2019. [Online]. Available: https://github.com/hulyav/eShopContainersSmall. [Accessed: 13-Oct-2019].

[107] H. Vural, “hulyav/eShopContainersBig,” 2019. [Online]. Available: https://github.com/hulyav/eShopContainersBig. [Accessed: 13-Oct-2019].

[108] C. S. Atole and K. V. Kale, “Assessment of package cohesion and coupling principles for predicting the quality of object oriented design,” in *2006 1st International Conference on Digital Information Management, ICDIM*, 2006, doi: 10.1109/ICDIM.2007.369321.

[109] “NDepend Product Features.” [Online]. Available: https://www.ndepend.com/features/. [Accessed: 10-Oct-2019].

[110] “Analytic hierarchy process - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Analytic_hierarchy_process. [Accessed: 17-Jan-2020].

[111] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems): Jiawei Han, Micheline Kamber, Jian Pei: 9781558609013: Amazon.com: Books*, Second. Massachusetts: Morgan Kaufmann, 2006.

[112] P. Josh and G. Adam, *Deep learning a practitioners approach.*, vol. 26, no. 7553. O’Reilly, 2017.

[113] B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach (Chapman & Hall/CRC Computer Science & Data Analysis Book 3) 1, Boris Mirkin - Amazon.com*. 2005.

[114] “Multi Criteria Decision Making Calculator.” [Online]. Available: https://people.revoledu.com/kardi/tutorial/AHP/MCDM_Calculator.html. [Accessed: 17-Jan-2020].

[115] O. Zimmermann, “Microservices tenets,” *Comput. Sci. - Res. Dev.*, vol. 32, no. 3–4, pp. 301–310, Jul. 2017, doi: 10.1007/s00450-016-0337-0.

[116] M. I. Josélyne, D. Tuheirwe-Mukasa, B. Kanagwa, and J. Balikuddembe, “Partitioning microservices: A domain engineering approach,” in *Proceedings - International Conference on Software Engineering*, 2018, pp. 43–49, doi: 10.1145/3195528.3195535.