INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Solução para o Cálculo de Indicadores-Chave de Desempenho nas Redes Móveis de Comunicação

SOFIA MOREIRA DA SILVA
Junho de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Framework for Calculating Key Performance Indicators in Mobile Communication Networks

Sofia Moreira da Silva

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Informatics, Specialisation Area of Software Engineering

Academic Supervisor: Isabel de Fátima Silva Azevedo
Enterprise Supervisor: Diego Asth Schuenck Leal

Porto, June 29, 2021

Dedicatory

To all those who crossed my path, directly or indirectly and contributed positively to what I am today.

Abstract

The work described in this report was developed in the context of the masters in Software Engineering and was carried out at Celfinet company.

Telecommunications are a fundamental foundation of our society. A fast, robust and accessible telecommunications network for everyone represents a strategic advantage for companies and countries. Their importance has rapidly grown in recent years.

However, as they become increasingly complex, communication networks need to be prepared for intensive and efficient use. Therefore, their monitoring is essential to detect problems and allow corrective measures or avoid their occurrence. Thus, how to rapidly calculate Key Performance Indicators (KPIs) deserved special attention in the initial study of telecommunications concepts.

The requirements were gathered as generic to abstract the functionalities and allow the customization to several enterprises. A generic solution was designed to be easily adapted to different operators. The developed application can import and process raw data to calculate KPIs, based on the Microservices architecture style. In summary, it is extensible and flexible and can be included in different ecosystems without complexity.

Keywords: Radio Access Network, Key Performance Indicator, Performance Management, Microservices

Resumo

O trabalho descrito neste relatório foi desenvolvido no âmbito do mestrado em Engenharia de Software e foi realizado na empresa Celfinet.

As telecomunicações são a base fundamental da nossa sociedade. Uma rede de telecomunicações rápida, robusta e acessível para todos, representa uma vantagem estratégica para empresas e países. A sua importância cresceu rapidamente nos últimos anos.

No entanto, à medida que se tornam cada vez mais complexas, as redes de comunicação necessitam de ser preparadas para o uso intensivo e eficiente. Portanto, o seu monitoramento é fundamental para detetar problemas e permitir medidas corretivas ou evitar a ocorrência de erros. Assim, como calcular rapidamente indicadores chave de desempenho mereceu atenção especial no estudo inicial de conceitos de telecomunicações.

Os requisitos foram abstraídos de forma a permitir a customização para várias empresas. Uma solução genérica foi projetada para ser facilmente adaptada a diferentes operadores. A aplicação desenvolvida pode importar e processar dados brutos de medidas de desempenho para calcular os indicadores chaves, baseado no estilo de arquitetura de microserviços. Em resumo, é extensível e flexível e pode ser incluído em diferentes ecossistemas sem complexidade.

Palavras-chave: Radio Access Network, Key Performance Indicator, Performance Management, Microservices

Acknowledgement

First of all, I thank my family and my boyfriend, especially my mother and father who have always helped and supported me in everything they could.

Many thanks to Celfinet and all my colleagues, particularly my supervisor, Diego Leal and the product team. I also thank the entire solutions team for lending me to the product team to carry out this project.

Special thanks to my supervisor, Isabel Azevedo, for being always available to clarify any doubts and for all the critiques and suggestions that helped in the development of this document.

A special thanks to everyone I met at this institution and who I am pleased to call a friend. I would also like to thank everyone who followed my academic path throughout the degree and masters, namely colleagues and professors.

Finally, I thank everyone who crossed my path, the friendships I made over the years and who were by my side regardless of distance and time without communicating.

Contents

List of Figures **xv**

List of Tables **xix**

List of Source Code **xxi**

List of Acronyms **xxiii**

1 Introduction **1**
1.1 Context . . . 1
1.2 Problem . . . 2
1.3 Objectives . . . 2
1.4 Research Methodology . . . 3
1.5 Document Structure . . . 4

2 The State of the Art **5**
2.1 Performance Management . . . 6
2.1.1 Performance Data Aggregation . . . 9
2.2 Key Performance Indicators . . . 10
2.3 Existing Applications . . . 13
2.3.1 Helix Performance Management . . . 13
2.3.2 VantagePM . . . 14
2.3.3 Comarch Performance Management . . . 14
2.3.4 NetPM . . . 15
2.3.5 Network Insight . . . 15
2.3.6 Summary of Applications . . . 16

3 Value Analysis **17**
3.1 Innovation Process . . . 17
3.2 Value . . . 19
3.3 Value Proposition . . . 20
3.4 Quality Function Deployment . . . 22

4 Requirements and Domain Analysis **25**
4.1 Requirements . . . 25
4.1.1 Use Cases . . . 26
4.1.2 Quality Attributes . . . 27
4.1.3 Constraints . . . 27
4.1.4 Architectural Concerns . . . 28
4.2 Domain Model . . . 28
4.3 Approaches to Aggregate Performance Data . . . 31
4.3.1 Approach 1 - Aggregations on-demand . . . 31

4.3.2 Approach 2 - Pre-defined Aggregations . . . 32
4.3.3 Approach 3 - Pre-defined and on-demand Aggregations . . . 33
4.3.4 Approach 4 - Pre-defined Aggregations and KPI Calculations . . . 34
4.3.5 Approaches comparison . . . 34
4.3.6 Alternative Approaches . . . 35

5 Architectural Design **39**
5.1 Attribute Driven Design . . . 39
5.2 ADD Step 1: Review Inputs . . . 40
5.3 Iteration 1: Establishing an Overall System Structure . . . 42
5.4 Iteration 2: Support creation of KPIs and formulas . . . 46
5.5 Iteration 3: Support the calculation of KPIs . . . 52

6 Solution **61**
6.1 Development Approach . . . 61
6.2 Implementation . . . 61
6.2.1 Inventory Microservice . . . 62
6.2.2 Aggregations Microservice . . . 66
6.2.3 Calculator Microservice . . . 69
6.2.4 Particular Situations . . . 71
6.3 Code Validation and Analysis . . . 73
6.3.1 Unit, Integration and Acceptance Tests . . . 73
6.3.2 Code Coverage . . . 75
6.3.3 Code Analysis . . . 76

7 Evaluation **79**
7.1 Introduction . . . 79
7.2 Evaluation Methodology . . . 79
7.3 Accuracy . . . 80
7.4 Processing time . . . 81
7.5 Scaling . . . 86
7.6 Retention times . . . 88
7.7 Overall Evaluation . . . 89

8 Conclusions **91**
8.1 Work Summary . . . 91
8.2 Project Contributions . . . 92
8.3 Limitations and Future Work . . . 92
8.4 Concluding Remarks . . . 93

Bibliography **95**

A AHP Method **99**
A.1 Phase 3 . . . 99
A.2 Phase 4 . . . 99
A.3 Phase 5 . . . 100

B Attribute Driven Design **103**
B.1 Iteration 1: Establishing an Overall System Structure . . . 103
B.2 Iteration 2: Support creation of KPIs and formulas . . . 110

B.3 Iteration 3: Support the calculation of KPIs 111

C Experimentation and Evaluation **113**
C.1 Processing times 113
C.1.1 UtranCell-Hour 117
C.1.2 NodeFunction-Hour 117
C.1.3 SubNetwork-Hour 118
C.2 Quantitative Evaluation Framework 118

List of Figures

2.1 Network domains . . . 6
2.2 PM nodal aggregations . . . 9
2.3 Sequence of aggregations to calculate the value of the counter attImmediateAssignProcs for the SubNetwork . . . 10
2.4 KPI optimisation strategy . . . 11
2.5 TEOCO Helix Performance Management (PM) . . . 13
2.6 Comarch Performance Management . . . 15

3.1 Evolution of technology connections . . . 18
3.2 Growth of global mobile data . . . 19
3.3 Value proposition . . . 21
3.4 Quality Function Deployment example . . . 22
3.5 Quality Function Deployment . . . 23

4.1 Use case diagram . . . 26
4.2 Domain model . . . 29
4.3 Domain and subdomains . . . 30
4.4 Activity diagram of the KPI calculation . . . 31
4.5 Approach one to calculate KPIs . . . 32
4.6 Approach two to calculate KPIs . . . 32
4.7 Approach three to calculate KPIs . . . 33
4.8 Approach four to calculate KPIs . . . 34
4.9 Hierarchical approach decision tree . . . 36
4.10 Hierarchical approach tree resulting from calculations . . . 37

5.1 Context diagram for KPI calculation system . . . 42
5.2 Logical view of the system . . . 45
5.3 Deployment diagram of the system . . . 46
5.4 Logic view of KPIFormulaInventory . . . 48
5.5 Sequence diagram for adding a KPI . . . 48
5.6 Sequence diagram for adding a tag . . . 49
5.7 Sequence diagram for tag a KPI . . . 49
5.8 Sequence diagram for get all KPIs . . . 50
5.9 Sequence diagram for adding a formula . . . 50
5.10 Sequence diagram for getting a KPI's formulas . . . 51
5.11 Data model of the KPIFormulaInventory . . . 52
5.12 Example of aggregations to execute . . . 53
5.13 Logic view of KPICalculator and PerformanceDataAggregator . . . 55
5.14 Sequence diagram for aggregate values . . . 55
5.15 Sequence diagram for calculating a KPI . . . 57
5.16 Sequence diagram for calculating a KPI . . . 57
5.17 Sequence diagram for calculating a KPI . . . 58

5.18 Sequence diagram to visualize the calculated KPIs . . . 59
5.19 Data model for PerformanceDataAggregator . . . 59
5.20 Data model for KPICalculator . . . 60

6.1 Agile methodology . . . 61
6.2 Package diagram of KPIFormulaInventory . . . 63
6.3 Class diagram of KPIFormulaInventory . . . 63
6.4 Usage of DTO . . . 65
6.5 Package diagram of PerformanceDataAggregator . . . 67
6.6 Example of aggregation fall-back . . . 67
6.7 Package diagram of KPICalculator . . . 69
6.8 Example of dependency injection . . . 70
6.9 Swagger UI of the KPICalculator component . . . 70
6.10 Manage topics view . . . 71
6.11 Manage calculateKPIs view . . . 72
6.12 View of the repositories in Azure Repos . . . 72
6.13 Build pipeline of the KPICalculator service . . . 73
6.14 Code coverage of KPIFormulaInventory service . . . 75
6.15 Code coverage of PerformanceDataAggregator service . . . 75
6.16 Code coverage of KPICalculator service . . . 76
6.17 Checkstyle report from KPICalculator component . . . 76
6.18 SpotBugs report from PerformanceDataAggregator component . . . 77

7.1 JMeter test plan . . . 82
7.2 Box plot of UtranCell-Hour calculations . . . 83
7.3 Wilcox test for UtranCell-Hour . . . 83
7.4 Box plot of NodeFunction-Hour calculations . . . 84
7.5 Wilcox test for NodeFunction-Hour . . . 85
7.6 Box plot of SubNetwork-Hour calculations . . . 85
7.7 Wilcox test for SubNetwork-Hour . . . 86
7.8 Configuration of the Horizontal Pod Autoscaler . . . 87
7.9 Management view of the workload KPICalculator in Rancher . . . 87
7.10 Example of aggregation fall-back . . . 88

A.1 IR values for square n matrices of order n . . . 99

B.1 Monolithic versus microservices . . . 104
B.2 Layered architecture pattern . . . 107
B.3 Microkernel architecture . . . 108
B.4 Sequence diagram for remove a KPI . . . 110
B.5 Sequence diagram for removing a tag . . . 110
B.6 Sequence diagram for getting a formula . . . 111

C.1 Lilliefors test for UtranCell-Hour . . . 117
C.2 Lilliefors test for NodeFunction-Hour . . . 118
C.3 Lilliefors test for SubNetwork-Hour . . . 118
C.4 Quantitative evaluation framework - overview . . . 119
C.5 Quantitative evaluation framework - dimensions . . . 120
C.6 Quantitative evaluation framework - functionality dimension overview . . . 121
C.7 Quantitative evaluation framework - quality dimension overview . . . 122

C.8 Quantitative evaluation framework - use cases 122
C.9 Quantitative evaluation framework - system 123
C.10 Quantitative evaluation framework - content quality 123
C.11 Quantitative evaluation framework - quality 124

List of Tables

2.1 Example of a PM dump file . . . 7
2.2 Comparison of applications by criteria . . . 16

3.1 Benefits and sacrifices to the customer . . . 20

4.1 Use cases description . . . 27
4.2 Quality attribute scenarios . . . 27
4.3 Constraints . . . 28
4.4 Architectural constraints . . . 28
4.5 Comparison of approaches by criteria . . . 34
4.6 Comparison matrix of the second approach criteria level . . . 36

5.1 Prioritized use cases . . . 41
5.2 Prioritized quality attributes . . . 41
5.3 Prioritized constraints . . . 41
5.4 Prioritized architectural concerns . . . 41

6.1 Tests for valid mathematical expressions . . . 65
6.2 Tests for invalid mathematical expressions . . . 65
6.3 Acceptance tests . . . 75

7.1 Definition of the formula's expression . . . 80
7.2 Generated data for the test case . . . 81
7.3 Accuracy test results . . . 81

A.1 Standardized matrix of the second level with relative priority . . . 99
A.2 Second level estimated processing time comparison matrix . . . 100
A.3 Standardized matrix of estimated processing time of the second level . . . 101
A.4 Second level different retention times comparison matrix . . . 101
A.5 Standardized matrix of different retention times of the second level . . . 101
A.6 Second level need for reprocessing comparison matrix . . . 101
A.7 Standardized matrix of need for reprocessing of the second level . . . 102
A.8 Second level development time comparison matrix . . . 102
A.9 Standardized matrix of development time of the second level . . . 102

B.1 Summary of patterns . . . 109
B.2 Kanban board of iteration 1 . . . 109
B.3 Kanban board of iteration 2 . . . 110
B.4 Kanban board of iteration 3 . . . 111

C.1 NEs involved in the tests . . . 113
C.2 Test definitions to register calculation times . . . 113
C.3 Collected times for UtranCell-Hour aggregation . . . 113

C.3 Collected times for UtranCell-Hour aggregation 114
C.4 Collected times for NodeFunction-Hour aggregation 115
C.4 Collected times for NodeFunction-Hour aggregation 116
C.5 Collected times for SubNetwork-Hour aggregation 116
C.5 Collected times for SubNetwork-Hour aggregation 117

List of Source Code

2.1 Example of a PM file report 8
5.1 Example of an insert statement to aggregate raw PM into cell/hour 56
5.2 Example of a query to return the formula values 58
6.1 Example of the usage of @Builder annotation 64
6.2 Example of the JpaRepository interface 66
6.3 Configuration of the aggregation's triggers 68
6.4 Example of the usage of jOOQ 68
6.5 Example of a unit test 74
6.6 Example of a integration test 74

List of Acronyms

3GPP	3rd Generation Partnership Project.
ADD	Attribute Driven Design.
ADR	Action Design Research.
AHP	Analytic Hierarchy Method.
API	Application Programming Interface.
CDE	Continuous Delivery.
CI	Continuous Integration.
CM	Configuration Management.
CQRS	Command Query Responsibility Segregation.
CR	Consistency Ratio.
CSP	Communications Service Provider.
DAO	Data Access Object.
DTO	Data Transfer Object.
EMS	Element Management System.
ETSI	European Telecommunications Standards Institute.
FCAPS	Fault, Configuration, Accounting, Performance, Security.
FFE	Fuzzy-Front End.
FM	Fault Management.
GIS	Geographic Information System.
GSMA	Global System for Mobile Communications Association.
HoQ	House of Quality.
HTTP	HyperText Transfer Protocol.
IT	Information Technology.
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector.
JDBC	Java Database Connectivity.
JPA	Java Persistence API.
KPI	Key Performance Indicator.

KQI	Key Quality Indicator.
NCD	New Concept Development.
NE	Network Element.
NMS	Network Management System.
OSS	Operation Support System.
PM	Performance Management.
PoC	Proof-of-Concept.
QEF	Quantitative Evaluation Framework.
QFD	Quality Function Deployment.
QoS	Quality of Service.
RAN	Radio Access Network.
REST	Representational State Transfer.
UML	Unified Modeling Language.
VM	Virtual Machine.
VoC	Voice of Customer.
XML	eXtensible Markup Language.

Chapter 1

Introduction

This document presents the work carried out under the Master Thesis (TMDEI) course of the Master in Software Engineering (MEI), as part of the educational offer of the Department of Informatics Engineering (DEI), present in Instituto Superior de Engenharia do Porto (ISEP) of the Politécnico do Porto (P.Porto).

This dissertation focus on the creation of a solution in a business company. This solution allows the creation and calculation of Key Performance Indicator (KPI). In this chapter, the problem is contextualized and defined. In addition, are described the objectives and the research methodology. To conclude is presented the structure of the thesis.

1.1 Context

The telecommunications sector have been growing in the past few years and becoming a fundamental building block of our society. The communication between people was shortened from days to hours, allowing countries to grow in an accelerated rhythm. Nowadays, wireless technologies are the primary method of communication.

The current networks built by the Communications Service Providers (CSPs) allow the users to communicate through data, voice, video and text. As the networks become more complex and the customer expectations keep growing, it is crucial to guarantee that the communication network tracks the demand (Li and Shen 2014). Therefore, a fast, robust, and accessible telecommunications network represents a strategic advantage for companies.

The network operators must understand what is happening at every moment to take the necessary corrective measures when a problem occurs or even prevent a potential problem from occurring. This understanding helps to improve the reliability and performance of mobile networks. This knowledge allows the companies to grow and build a strong position in the market as they become more trustworthy and the Quality of Service (QoS) keeps improving.

The information that represents the state of the network and each of its components are presented in the form of events. The events are published by the network systems, several times per day reaching sometimes, hundreds of gigabytes per day. The set of those events is called Performance Management (PM) data. It is possible to correlate the events between them to describe the components within the operator network. By doing this, we measure the performance of the network.

With PM data, it is possible to calculate KPI, which are “a set of selected indicators used for measuring the current network performance and trends" (NOKIA 2016). There are some standard definitions for KPIs developed by corporations such as International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), European Telecommunications Standards Institute (ETSI), 3rd Generation Partnership Project (3GPP) or Global System for Mobile Communications Association (GSMA) (Kuklinski and Tomaszewski 2019). However, some operators have their definitions and classifications of the KPIs. Usually, these network operators own a group of KPIs considered the most important to the success of the implemented solutions.

1.2 Problem

In the telecommunications industry, the equipment vendors have their dedicated equipment, applications and protocols that manage the communication infrastructure of a CSP. These are, typically, incompatible between themselves and require efforts to coexist and collaborate, creating a challenge in network management. Additionally, distinct operators have different characteristics, operate under diverse markets and various government regulators. All this causes difficulties when trying to create an application that can handle all the heterogeneous qualities in that ecosystem.

This study has been carried in the scope of an existing industrial solution by Celfinet[1] company named Vismon Manager[2] platform. This platform has been helping many operators to configure, planning and managing the mobile network with information extracted from the Operation Support System (OSS), namely Radio Access Network (RAN). The idea of adding the PM functionality to the Vismon Manager platform has become more attractive, influenced by the customer’s requests.

A module that allows the CSP to monitor and analyses the QoS of the network helps to realize possible wrong configurations in the network. For that, it must present network quality values that indicate a performance measure, such as calculated KPIs.

Currently, Celfinet company has a couple of operator’s specific solutions that work with PM data, developed to support a customer to keep track of the network performance incidents. However, there is no internal solution capable to support different network operators, vendors and technologies. That could be used as a base for any project/product that needs to calculate KPIs.

1.3 Objectives

The objective of this work consists of creating a framework that can create KPIs and formulas and apply them whenever it is necessary to calculate the KPIs. Therefore, the purpose is to design the architecture of the solution and create its Proof-of-Concept (PoC). To accomplish this, the following steps have been followed:

1. An analysis of the business concepts involved for a better understanding of the project context.

[1] https://www.celfinet.com
[2] https://www.celfinet.com/vismon-manager/

2. A study of different approaches to calculate KPIs consider some criteria as the aggregation of the performance counters or distinct retention times for saving data.
3. The design of the architecture of the system, with the adoption of good software engineering practices; the development of a PoC, consisting of a framework to create and calculate KPIs and formulas. The solution must comprehend the granularity period of the performance measures, the constraints associated with the import, filter and process of valid PM data and the calculation of KPI.

With this information, the research question was defined, based on PM data aggregations, since executing them before the calculation of KPIs could improve the processing time.

The research question is verified in the evaluation of the developed solution, with the defined hypothesis and tested indicators. The research question is as follows:

What is the effect of aggregation of performance counters to calculate a KPI in a shorter period?

Nonetheless, this solution must be extendable and flexible to be easily pluggable in different contexts and solutions that require the processing of PM data and the calculation of KPIs. Besides, it must be vendor and technology agnostic.

It must be ready to deploy into containerized environments to facilitate resilience and scalability provided by these environments. An architecture style appropriate to this context, like the Microservice architecture style, should serve as the base to design the system.

1.4 Research Methodology

The Action Design Research (ADR) was used as the research methodology. This method focuses on "generating prescriptive design knowledge through building and evaluating ensemble IT artefacts in an organizational setting" (Sein et al. 2011). It is composed of four stages:

1. Problem Formulation
2. Building, Intervention and Evaluation
3. Reflection and Learning
4. Formalization of Learning

In the first stage, Problem Formulation, the objectives and the research question were defined as presented in section 1.3. Besides this, to identify theoretical bases, an analysis of the business needs was carried out. It started with the problem analysis that includes the study of the relevant business concepts. The applications already available on the market were also analysed to understand which features they have in common and which were able to enhance the solution developed. Afterwards, the value analysis of the project was carried out.

The Building, Intervention and Evaluation stage is described in chapters 4, 5, 6 and 7. With the problem framing and theoretical premises done in stage one. Some KPI calculation approaches were defined, with pre-defined and on-demand aggregations. They were created based on the aggregation of counters and are described in chapter 5. Then, the Analytic

Hierarchy Method (AHP) method was applied to select the most appropriate approach considering previously defined criteria.

Herewith, it was necessary to start making decisions about the design of the IT artefact, using architectural patterns and styles and good software engineering practices. To meet the project requirements (see chapter 4), the system was designed based on the microservices architectural style, not failing to analyse other possible patterns. This pattern allowed taking advantage of its benefits, as the low coupling between the services, ease of deployment and scaling.

To evaluate the solution and to obtain the answer to the research question, a hypothesis and some indicators were defined. These indicators were used to evaluate the solution and have allowed them to appraise if the aggregations made previously permitted to return data in a shorter period.

Stage three, Reflection and Learning, is "continuous and parallels the first two stages" (Sein et al. 2011). Therefore, this stage is composed of the decisions that have been applied during the PoC realization. Those decisions are present in 5.

Lastly, in stage four, "researchers outline the accomplishments realized in the IT artefact and describe the organizational outcomes to formalize the learning" (Sein et al. 2011). Thus, chapter 8 describes this stage.

1.5 Document Structure

This chapter, **Introduction**, exposes what is necessary for the initial understanding of the project, namely its context, problem and research methodology.

The second chapter, **The State of the Art**, contains business concepts and the state of the art in existing KPI management applications.

Regarding chapter 3, **Value Analysis**, describes the value analysis of the project regarding the respective process.

In chapter 4, **Requirements and Domain Analysis**, the necessary details to understand the project theme, stakeholders and existing restrictions are presented, as well as the alternative approaches regarding counters aggregation and KPI calculation.

Chapter 5, **Architectural Design**, presents the architectural design, including the most relevant decisions and views.

Concerning 6, **Solution**, aims to denote the most relevant aspects of the PoC implementation.

Chapter 7, **Evaluation**, presents the evaluation methodology, the indicators, the evaluation tools and the results.

To conclude, the **Conclusions** chapter exposes the conclusions of the project as well as the limitations and future work.

Finally, some **Appendices** were included to add relevant details to the document.

Chapter 2

The State of the Art

This chapter presents a theoretical background regarding the PM of the network elements in a RAN, the segment of a mobile network where the study is focused, and its respective logical structure. Besides this, the concept of KPI in telecommunications, its usage and calculation are explained.

Furthermore, it presents a group of applications existing on the market to exemplify some existing features in this type of solution.

This work focuses on the RANs segment of mobile telecommunications systems. RAN segment is responsible for associating single devices to other sections of a mobile network over radio connections. This type of network is composed of several Network Elements (NEs). A NE is hardware on which a collection of processes is executed to offer services to the customer (Sathyan 2016).

An Element Management System (EMS) manages a group of NEs, collecting data from them (Sathyan 2016). Although most of the EMSs available in the market support the management of a set of NE, the ideal scenario involves only a single NE.

To manage the set of EMS presented in the RAN it is used a Network Management System (NMS). This system allows a "holistic view of the entire network" (Sathyan 2016). The EMS collects the events generated by the NE to NMS obtain information about the state of the network.

As there can be several NMS in the network, they can communicate with each other and the network Operation Support System (OSS) to pass management data. Figure 2.1 illustrates the layered architecture of the network regarding the management of NEs.

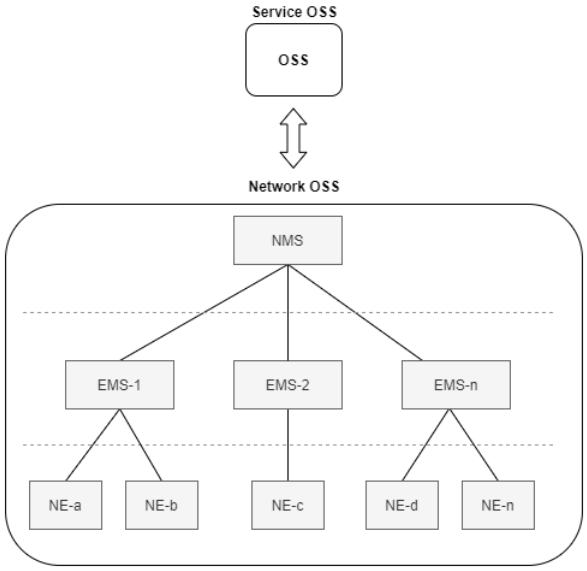


Figure 2.1: View of the network systems, adapted from (Sathyan 2016)

The EMSs provides Fault, Configuration, Accounting, Performance, Security (FCAPS) functionality. ITU-T specifies five management areas, composed of a set of functions, described in the following list (ITU-T 2000, Sinche et al. 2020):

- **Fault Management** - contains a group of functions that enables set up the "detection, isolation and correction of abnormal operation of the telecommunication network and its environment".
- **Configuration Management** - provides functions to configure the system, "such as configuration file management, inventory management, and software management".
- **Accounting Management** - "collects network resources usage information, for resource planning and/or charging".
- **Performance Management** - provides functions "to evaluate and report upon the behaviour of telecommunication equipment and the effectiveness of the network or NE".
- **Security Management** - provides functions to monitor and control the mechanisms "for secure access to network devices, resources and services".

More details on PM and KPIs are explained in the next sections.

2.1 Performance Management

The goal of PM data "is to continuously monitor and measure the performance of various subsystems" (Martinez-Mosquera and Navarrete 2020), allowing the troubleshooting and optimisation of the network. This data is dumped into files and is "collected from the network element at regular intervals by the NMS" (Sathyan 2016). It can be transferred into one or more OSS and/or NMS . Besides the discovery network's elements, download fault data, NMS allows the download of PM data. Usually, the dump file is collected at

a particular frequency, known as the granularity period (Martinez-Mosquera and Navarrete 2020).

The 3GPP provides specifications that cover cellular telecommunications technologies beyond the third generation. According to the 3GPP specification (3GPP Specification and Services 2020a), there are four valid values for the definition of granularity period: 5 minutes, 15 minutes, 30 minutes and 1 hour. In most cases, the minimum granularity period is defined at 5 minutes however, in some cases, a larger granularity period can be specified.

PM data reports are composed of series of counters. These counters are generated by the occurrence of a measured event and grouped according to their functionality. The term *measured object class* was defined by 3GPP to identify those groups (3GPP Specification and Services 2020b). Some several parameters and events can be measured and correlated between them. Table 2.1 represents an example of a report containing two example counters: `attTCHSeizures` and `attImmediateAssignProcs`, simulating the production of the data on January 31, 2021, at 14h.

Table 2.1: Example of a PM dump file

NE	Timestamp	attTCHSeizures	attImmediateAssignProcs
cell-111	2021-01-31 14:00:00	243	567
cell-222	2021-01-31 14:00:00	890	123
cell-333	2021-01-31 14:00:00	456	678

Following the 3GPP recommendation, the generated file must also contain "an identification of the measurement job that generated the report; an identification of the involved measurement type(s) and the measured network resource(s) [...]; a time stamp, referring to the end of the granularity period; [...]the result value(s) and an indication of the validity of the result value(s);[...] an indication if the scan is not complete, and the reason why the scan could not be completed" (3GPP Specification and Services 2020a). The report also contains a field that represents the reporting period, in other words, each counter is an aggregate of event values during the granularity period.

3GPP developed file format definitions for eXtensible Markup Language (XML) and Abstract Syntax Notation 1 (ASN.1). Nonetheless, telecommunication equipment vendors can have their unique formats. Listing 2.1 shows an example of a PM file report compliant with the 3GPP specification. This example contains all the elements mentioned above, identified in the following list:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="MeasDataCollection.xsl"?>
<measCollecFile [...]>
  <fileHeader fileFormatVersion="32" vendorName="Company1" dnPrefix="SubNetwork=
    CountryNN,MeContext=ND1234,ManagedElement=ND1234">
    <fileSender localDn="DC=a1.company1.com,SubNetwork=1,IRPAgent=1" elementType
    ="Node"/>
    <measCollec beginTime="2021-01-31T14:00:00+00:00"/>
  </fileHeader>
  <measData>
    <managedElement localDn="SubNetwork=CountryNN,MeContext=ND1234,
    ManagedElement=ND1234" userLabel="Node telecomville"/>
    <measInfo>
      <job jobId="1231"/>
      <granPeriod duration="PT900S" endTime="2021-01-31T14:15:00+00:00"/>
      <repPeriod duration="PT900S"/>
      <measTypes>attTCHSeizures attImmediateAssignProcs succImmediateAssignProcs
    </measTypes>
      <measValue measObjLdn="NodeFunction=1,UtranCell=CLL1234">
        <measResults>234 567 789</measResults>
      </measValue>
      <measValue measObjLdn="NodeFunction=1,UtranCell=CLL1235">
        <measResults>890 123 234</measResults>
      </measValue>
    </measInfo>
  </measData>
  <fileFooter>
    <measCollec endTime="2021-01-31T14:15:00+00:00"/>
  </fileFooter>
</measCollecFile>
```

Listing 2.1: Example of a PM file report

1. **job identification** - 1231
2. **granularity period** - 900 seconds = 15 minutes
3. **reporting period** - 900 seconds = 15 minutes
4. **measurement types** - attTCHSeizures, attImmediateAssignProcs and succImmediateAssignProcs
5. **managed element** - ND1234
6. **measured network resource** - UtranCell=CLL1234 and UtranCell=CLL1235
7. **Result value UtranCell=CLL1234** - attTCHSeizures = 234, attImmediateAssignProcs = 567, succImmediateAssignProcs = 789
8. **Result value UtranCell=CLL1235** - attTCHSeizures = 890, attImmediateAssignProcs = 123, succImmediateAssignProcs = 789

Because OSS and NMS have more functionalities than the generation of PM reports sometimes it needs to prioritize processes according to their importance. Consequently, it can lead to defects in PM report. Performance reports may arrive incomplete, partially corrupted, delayed, versioned (the same file arrives with different counter values), or even in pieces. Software that manages performance data must handle these constraints to allow to represent the operator network correct state.

The networks of the biggest CSP are composed of thousands of NE that generated thousands of performance reports which leads to enormous amounts of information regarding NEs. A PM report can contain several performance measurements. They are used to calculate KPIs defined by the network operators to simplify the access to the generated data.

2.1.1 Performance Data Aggregation

Performance measures are defined for a managed object and granularity period. A managed object can be any NE that is managed. A managed object can be aggregated into three levels: temporal, nodal and spatial. Usually, to aggregate the measures simple aggregations are performed such as sum, minimum or maximum, average or a basic count. The most used aggregation is the sum. Also, the equipment vendor must indicate what aggregation to use in each counter that needs to be identified in the equipment documentation.

Regarding temporal aggregation, a managed object can be aggregated by hour, day, week and so on. If a counter is defined at a granularity period of 15 minutes, and it is necessary to know its value per hour, the four values of the four-quarter hours ($1hour/4 = 15min$) need to be aggregated.

The nodal aggregation is executed on the elements of the RAN. Figure 2.2 illustrates the nodal aggregation. From the 4^{th} generation technology, the controller is not presented in the network architecture. The left side represents the aggregations in the 2^{nd} and 3^{rd} generations. The right side represents the aggregations in the 4^{th} and 5^{th} generations.

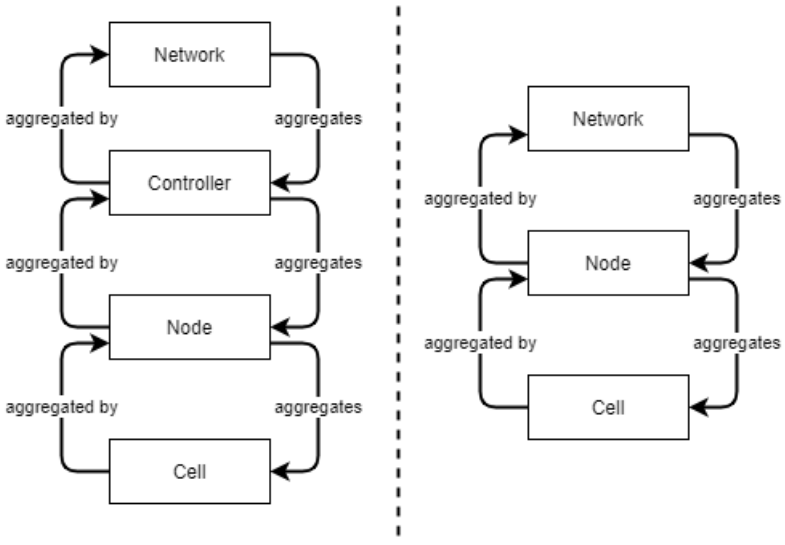


Figure 2.2: PM aggregations between the NEs

To give a practical example, let's look at the partial file in the previous section, listing 2.1. To know the unique name of the measuring element in the network, it is necessary to join two components included in the file: `managedElement.localDn + measValue.measObjLdn`. Thereby, seeing the content of example 2.1, it is possible to observe that two elements were measured, presented in the following list:

1. SubNetwork=CountryNN, MeContext=ND1234A, ManagedElement=ND1234, NodeFunction=1, UtranCell=CLL1234

2. SubNetwork=CountryNN, MeContext=ND1234A, ManagedElement=ND1234, NodeFunction=1, UtranCell=CLL1235

The unique path of the network element contains information about nodal aggregations that can be performed. In example 2.1, the measuring element is `UtranCell`, present in the last level of the path. This element can be aggregated into `NodeFunction`, `ManagedElement`, `MeContext` and `SubNetwork`. Since the elements `NodeFunction`, `ManagedElement`, `MeContext` are equal in the element path, the aggregation is only made for `NodeFunction`, because the results are the same.

Figure 2.3 represents the necessary aggregations to perform to calculate the value of the counter `attImmediateAssignProcs` for the `SubNetwork` `CountryNN`, at 14:00. The nodal aggregation defined for the counter `attImmediateAssignProcs` is SUM.

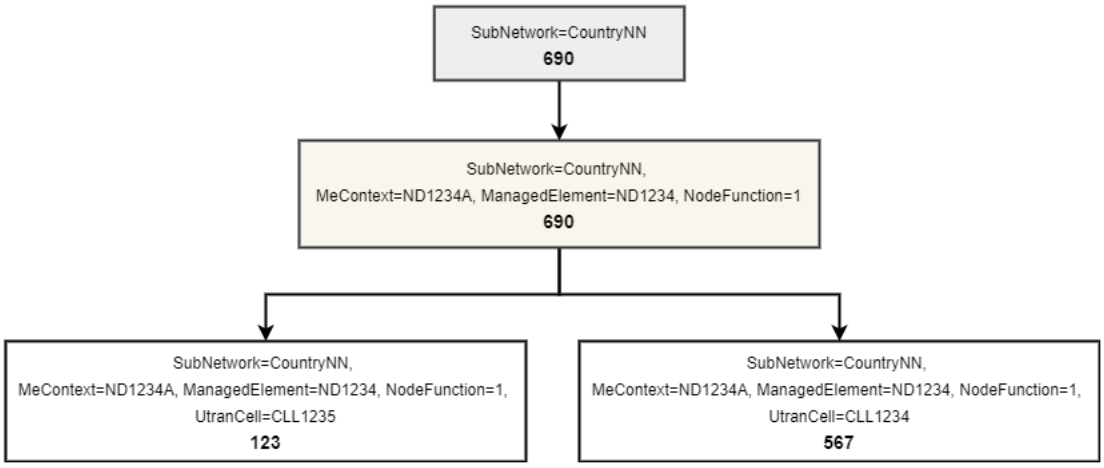


Figure 2.3: Sequence of aggregations to calculate the value of the counter attImmediateAssignProcs for the SubNetwork

Spatial aggregations are another type of aggregations that can be performed when is necessary to know a value in some cluster, neighbourhood or city. This information is contained in the Configuration Management (CM) dump files as it relates to configurations of network elements. To enrich the information with configuration data is necessary to look at the unique path of the NE.

Besides the simple aggregations presented above, network operators can define more complex aggregations. An example of that is the Peak Hour/Busy Hour aggregation. This aggregation represents the KPIs values for the busiest hours in a day. When these KPIs reach the maximum value, other calculations of other KPIs are triggered. Example: calculate KPI Voice Traffic and Availability at the hour when the KPI Dropped Call is at the maximum value defined.

2.2 Key Performance Indicators

Key Performance Indicators (KPIs) are used in several contexts to measure performance. They can be used in organizations to define what is most critical for success or to monitor the performance of the employees.

In telecommunications, as mentioned in section 1.1, KPIs are used to measure the current network performance and trends. Since the performance parameters can be correlated between them, the number of possibilities is almost infinite. Thereby, the network operators define business targets based on a set of KPIs that are considered essential to the success of the business. They provide "the guidance to define network optimisation targets" (Kreher 2006). Figure 2.4 represents the lifecycle of the definition of KPIs motivated by the network optimisation strategy of the operator (Kreher 2006).

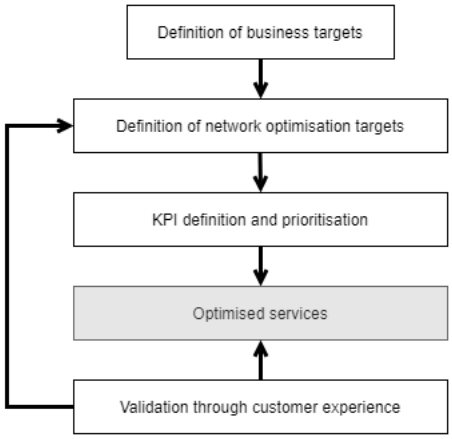


Figure 2.4: Network operator's optimisation strategy (Kreher 2006)

KPIs are usually grouped by category. 3GPP created specifications for 2^{nd}, 3^{rd}, 4^{th} and 5^{th} generations for KPI formulas categories. The following list presents some categories and their definitions:

- **Accessibility** - Accessibility is "the ability of a user to obtain a requested service from the system" (Zhang 2017).
- **Retainability** - Retainability is "the ability of a user to retain its requested service once connected for the desired duration" (Zhang 2017).
- **Integrity** - Integrity is the "property that data have not been altered in an unauthorized manner" ((ITU) 2008).
- **Availability** - Availability refers "to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval"((ITU) 2008).
- **Mobility** - "key procedure for ensuring that users can move freely within a network" (Zhang 2017).

Besides the categories defined in the previous list, 3GPP also defined reliability, maintainability and utilization as KPI categories.

The majority of the KPI formulas are simple and can have many different types of variables: constant values, counter values from the PM report, timer values or even computed KPIs (Kreher 2006). Also, the network operator can include values from CM reports that contain the configured managed objects in the network. The majority of the formulas for the KPIs are expressed in a mathematical expression. However, there are some cases that the formula is a logical condition to determine the Boolean value of the KPI.

To present an example of a KPI formula defined by 3GPP, the equation presented below shows how to calculate the KPI E-UTRAN Cell Availability, "a KPI that shows Availability of E-UTRAN Cell" (3GPP Specification and Services 2020c) in 4^{th} generation technology. The `mp` stands for the measurement period.

$$CellAvailability = \frac{mp - \sum_{cause} RRU.CellUnavailableTime.[cause]}{mp} x100 \qquad (2.1)$$

The result of this equation is the percentage of time that the cell was considered available. The formula is also valid for cells of 3^{rd} generation technology with a minor change on the measure counter name: `RRU.UTRANCellUnavailableTime`.

Although the provided specifications by 3GPP, equipment vendors can define their definitions of KPIs and network operators can elaborate their formulas definitions.

A KPI may have associated integrity that indicates the reliability of the calculated data. The integrity of the records used for aggregation can be temporal, nodal integrity or the combination of the two. Temporal integrity is calculated by dividing the number of records used to reach the KPI value by the expected number of records. The same logic is applied to nodal integrity: divide the number of NEs expected by the total number of NEs.

To illustrate a practical example: to perform a temporal aggregation of PM data in a day with only 18-hour records, the temporal integrity is $18/24 = 75\%$; to perform a nodal aggregation of performance data with only three cell records (12 cells in total), the nodal integrity is $3/12 = 25\%$. To know the total integrity of the data, then it is necessary to obtain the product of the two integrity: $(18x3)/(24x12) = 18.75\%$.

There are some standard reports and dashboards to analyse KPIs, including tables, charts or maps. There can be a report that shows the N worst NEs (e.g.: 50 worst cells of a KPI in a period) or that contains Historical values of a KPI of a NE.

Furthermore, dashboards containing KPI values allow the user to drill-down and drill-up by NE or by time:

- **Drill-down**: analysing a KPI with the `SubNetwork-Week` aggregation, the user may want to drill down by counters involved in the formula, time (day, hour, half-hour, quarter-hour) or NE (node, cell).
- **Drill-up**: analysing a KPI with the `Cell-Day` aggregation, the user may want to drill up by time (week, month) or NE (node, subNetwork).

Nevertheless, the user must have the possibility to define its filters and groupings of NEs and periods, to visualize the calculated values of the KPIs most conveniently and appropriately. To achieve this goal, platforms must store historical data, which leads to millions of records in databases. Defining different retention values for different periods and types of data help to decrease the amount of data saved.

2.3 Existing Applications

This section presents some PM applications that process and manages performance data. Since this solution was developed in a business context, the selected applications took these criteria into account. PM applications existing on the market were then researched and grouped by their functionality and complexity. The simplest ones were selected since they are more similar to the framework developed and the feature set is smaller. With this, the most relevant information was collected to briefly describe the applications, with their main features summarized. In addition, this knowledge was used to carry out the value analysis process.

During the research, were found some solutions and proposed architectures that improve the way that NMS/OSS stores and provides the performance data of the NEs. However, those were excluded from this analysis since their main functionalities are not similar to the KPI calculation framework.

2.3.1 Helix Performance Management

TEOCO is a company that provides products and services to CSPs. One of those solutions, called Helix Performance Management[1], was built for managing end-to-end performance and QoS of telecom networks of all types. Figure 2.5 represents an example of a dashboard containing a Geographic Information System (GIS) interface and some charts.

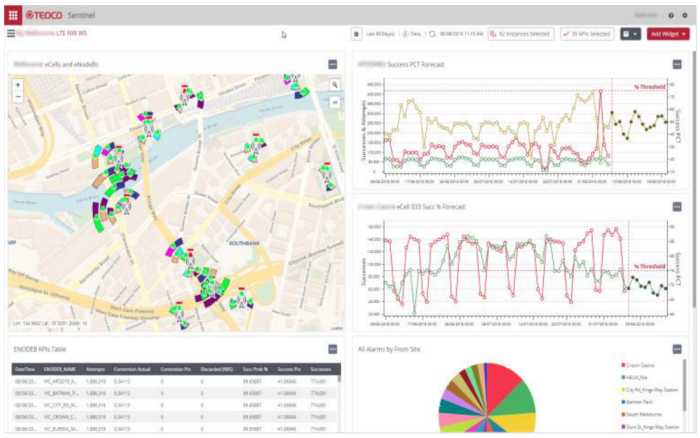


Figure 2.5: TEOCO Helix PM KPI dashboard of cells and nodes of 4^{th} generation technology, from (*HELIX PERFORMANCE Gain deep network insights* n.d.)

The following list sums the set of functionalities of the product:

- Process large volumes of data - this system collects and analyses millions of performance measurements by day. Supports Big Data storage using the Hadoop framework and relational databases.
- Calculation of metrics - this application can calculate KPIs and Key Quality Indicators (KQIs), enhancing system performance and traffic-analysis calculations.

[1] https://www.teoco.com/products-services/service-assurance/performance-management/

- Improves the resolution of network performance issues - the data processing and analytics algorithms can be applied to the management of service-oriented KPIs and KQIs.
- Machine Learning algorithms to improve network performance and to understand the network behaviour.
- Adjust performance thresholds automatically and generates alarms based on thresholds.
- User Interface that centralizes access to the Helix Performance Management modules.

2.3.2 VantagePM

P.I. Works provides products for mobile network planning, management and optimization. For mobile operators, in the context of performance management,they offer a solution called VantagePM[2]. The following list presents some of the solution functionalities:

- Multi-vendor and multi-technology.
- Combines and correlates several data types from distinct domains as PM, CM, Fault Management (FM), among others.
- Vendor agnostic KPIs to detect possible problems and trigger alarms.
- Interactive and flexible dashboards for any type of user with drill-down and drill-up functionalities.
- Report and map visualization to enhance network performance analysis.

2.3.3 Comarch Performance Management

Comarch develops Information Technology (IT) products for multiple industries. The Comarch Integrated Assurance[3] is composed of a module, Comarch Performance Management, that provides a centralized point of performance monitoring with Big Data integration. Figure 2.6 represents a custom dashboard of the product, containing charts with the KPIs values.

- Multi-vendor and multi-technology.
- Holistic view of the physical and virtual network.
- Integrates KQIs with thresholds generated by processing events from different sources.
- Artificial Intelligence and Machine Learning algorithms to improve network performance by defining baselines and performing automatic detection of anomalies.
- Pre-prepared standard patterns of parameter correlation.
- Supports 5G and IoT services monitoring.

[2]https://www.piworks.net/products#VantagePM
[3]https://www.comarch.com/telecommunications/service-assurance/performance-management/

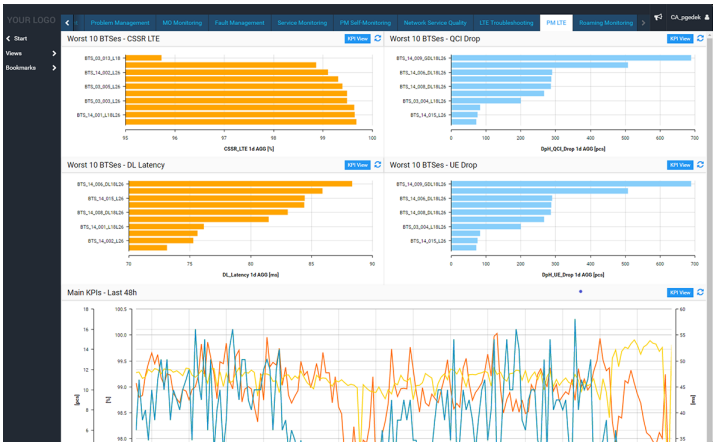


Figure 2.6: Example of a custom dashboard in Comarch Performance Management, from (Comarch 2021)

2.3.4 NetPM

Digitata Networks offers services and solutions to assist mobile network operators in monitoring, controlling and auditing their networks. NetPm[4] monitors KPIs and trends in a network, providing end-to-end performance monitoring. It offers distinct views for KPIs, like drill-down functionality or reports.

- Multi-vendor and multi-technology.
- Integration and correlation of different sources to enrich data, as CM data contains configuration information.
- Analysis of KPIs and counters over a specified time frame. It comes with predefined KPI configurations by 3GPP standards yet allows to customize them.
- GIS visualization of KPIs.

2.3.5 Network Insight

Yuvo supplies several solutions related to networks. One of its solutions, Network Insight RAN Analytics[5], manages performance data along with other types and can correlate all RAN data.

- Multi-vendor and multi-technology.
- Centralized platform - correlation of PM, CM and FM to understand the root cause of the problems.
- Normalized KPI formulas according to the 3GPP standards.
- Machine learning algorithms to identify NEs with problems that could result in a bad QoS.
- Alarm notification whenever a KPI is degraded.

[4]https://www.digitatanetworks.com/netpm/
[5]https://www.yuvo.net/ran-analytics

- Historical trend of KPI.

2.3.6 Summary of Applications

Through the analysis made on available information on the existing applications on the market, it is possible to conclude that there are some applications more complex than others. Some applications support Big Data and include Machine Learning algorithms and Artificial Intelligence in calculating KPIs and problem detection. Despite the size of the feature collection, almost all have common functionalities. The comparison of the applications is shown in table 2.2, based on the following criteria:

1. Multi-vendor and multi-technology support
2. Creation of KPIs and formulas
3. Calculation of KPIs
4. Combination with other types of data
5. Alarm notification for problem detection

Table 2.2: Comparison of applications by criteria

Application	**Criteria 1**	**Criteria 2**	**Criteria 3**	**Criteria 4**	**Criteria 5**
Helix Performance Management	✔	✖	✔	✖	✔
Vantage PM	✔	✖	✔	✔	✔
Comarch Performance Management	✔	✖	✔	✔	✔
NetPM	✔	✔	✔	✔	✖
Yuvo Network Insight	✔	✖	✔	✔	✔

From the previous table, it is possible to conclude that an application that does not support multi-vendor and multi-tech is at a disadvantage compared to what exists on the market. Also, it is common that PM applications combine other data types to help the user to detect possible sources of problems. Alarms notifications for problem detection also come from processing alternative types of data, such as fault management data.

Also, the previous table shows that only one application, NetPM, allows the creation of custom formulas for KPIs, besides the standard formulas. According to table 2.2, this characteristic is uncommon in the analysed PM tools that can enhance the KPI calculation framework.

The developed framework covers the most important criteria of table 2.2, such as multi-vendor and multi-technology support and the creation of KPIs and formulas. With this framework, the Vismon Manager platform can present itself as a serious concurrent in this market segment.

Chapter 3

Value Analysis

The value analysis process improves the profitability of a product, using many techniques to achieve this objective. The analysis "concerns the function of a product to meet the demands or application needed by a customer" (Rich and Holweg 2000). This chapter starts by describing the innovation process, followed by the value proposition ending with the Quality Function Deployment (QFD).

3.1 Innovation Process

In the past years, many companies have improved their innovation process by adopting developed models to help project management. The innovation process is divided into three parts: "the front end of innovation, the new product development process and commercialization" (P. Koen 2013). The New Concept Development (NCD) was developed to fill the lack of common terms and definitions for Fuzzy-Front End (FFE), providing a unified language. NCD model is divided into three distinct areas: the centre of the model, the inner part and the external environmental factors.

The inner area defines the five controllable activity elements of the FFE: opportunity identification, opportunity analysis, idea generation and enrichment, idea selection, and concept definition (P. A. Koen et al. 1996). In this analysis, the Opportunity Identification and Opportunity Analysis were used for the innovation process that was leveraged in the study.

Opportunity Identification
This element is used for the organization to identify "opportunities that it might want to pursue" (P. A. Koen et al. 1996) and is usually driven by the business goals. Nonetheless, it can also be driven by customer requests to identify a new sales approach, business upgrade or even a new manufacturing process. To identify the opportunity, several techniques are used, such as brainstorming or mind maps.

Celfinet identified the inclusion of a PM feature in the Vismon Manager platform as an opportunity to pursue. This new feature would allow capturing competitive advantage as a result of the organization's roadmap. With this, some ideas were exchanged between customers, possible customers and telecom engineers that work daily with the platform. The feedback was strongly positive.

It was noticed that the addition of this feature could help the Vismon Manager platform to gain advantages comparing to other solutions and could be a crucial point in the negotiation process. Therefore, the creation of a generic base that manages PM data could help in the

integration of this feature in the product.

Opportunity Analysis

This element is used to translate the "opportunity identification into specific business and technology opportunities" (P. A. Koen et al. 1996). The effort is invested in searches and studies to understand market trends. Many of the tools used in the previous element are used in the opportunity analysis like roadmaps, technology and competitive trend analysis.

This opportunity is included in the segment of telecommunications, specifically in CSPs. The mobile networks are evolving to include the 5^{th} technology generation, Wi-Fi 6 and IoT, with more and more people subscribing to the services. By about 2023, "5G will likely coexist with the many other cellular mobile and Wi-Fi standards, as well as wired standards, that are widespread today" (Lee et al. 2019).

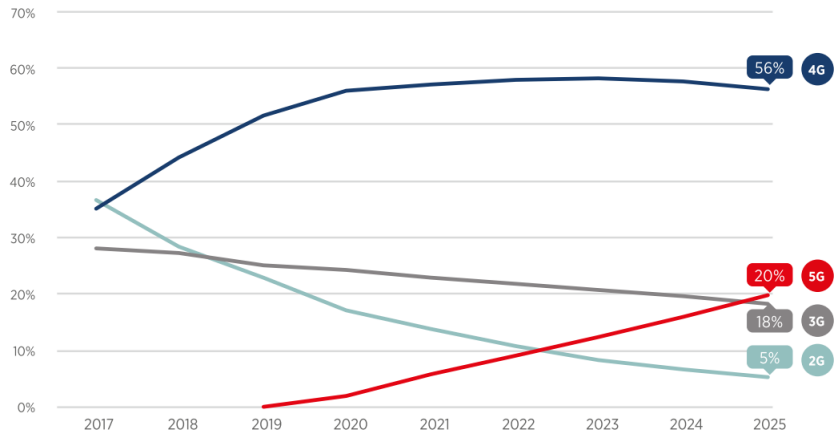


Figure 3.1: Evolution of technology connections (% of connections) (GSM Association 2020)

As shown in figure 3.1, 5G networks will continue to expand. However, it only represents 20% of all connectivity. The networks evolve, yet it is not possible to make a complete switch of the entire infrastructure, with 24% of connections still 2G and 3G. With all the heterogeneity of mobile networks, PM solutions help CSPs to monitor the network performance, processing data from all technologies.

Network operators need continuous investments to keep up with the demand and to provide the services that the consumers and businesses expect (GSM Association 2020). Several rural areas and underdeveloped countries do not yet have network coverage capable of meeting all consumer needs. With the Covid-19 pandemic, many CSPs realized that they have to invest in those areas so that the populations living there can access the same services provided in urban areas. Moreover, the demand for mobile services in more densely populated areas has also increased.

According to the last Mobile Economy Report, developed by GSMA, "operators are expected to invest around $1.1 trillion worldwide between 2020 and 2025 in mobile CAPEX" (GSM Association 2020). Also, it is expected that, by 2025, mobile Internet subscriptions increase to 1.2 billion people. To support this, figure 3.2 shows the growth of global mobile data usage by 2025. Is it possible to observe that the data usage will grow almost fourfold,

"purred by increased smartphone adoption and availability of affordable high-speed network services" (GSM Association 2020).

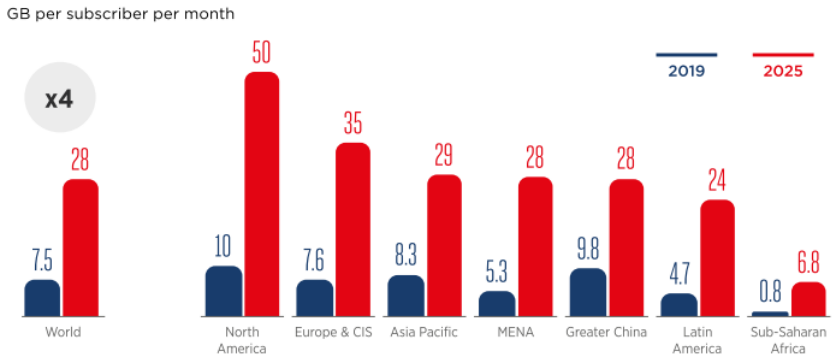


Figure 3.2: Growth of global mobile data (GSM Association 2020)

With this, a solution that calculates KPIs based on the performance data of the mobile network gains importance, as it helps operators in their growth by monitoring the network performance and consequently retaining customers with the quality of the service provided.

Furthermore, the development of a generic solution that can calculate KPIs provides Celfinet with a solid base to develop products and solutions that need to obtain KPIs values. Adding this component to the Vismon Manager platform, it is possible to include a PM feature, allow users to configure network parameters, taking into account the values obtained for the different calculated KPIs.

To achieve this goal, Celfinet solutions can be refactored to reuse this framework and facilitate the detection of errors that may occur.

3.2 Value

The concept of value is difficult to understand and conceptualize and has been defined in the distinct theoretical background as need, desire, interest, beliefs, attitudes and preferences (Nicola, Ferreira, and Pinto Ferreira 2012).

The value of this solution is the calculation of KPIs and their visualization in every NE, time or place, helping CSPs to understand the QoS that is being provided to their users.

The perceived value can be considered as the relationship between the perceived benefits and perceived sacrifices. Hence, distinct customer segments have distinct perceptions of the same product or service (Ulaga and Eggert 2006).

Table 3.1 was created to evaluate the perceived value, presenting the benefits and sacrifices associated with this project.

The solution allows customers to create KPIs for the system to calculate only the indicators that matter for the CSP's business. Controlling this, the list of KPIs is clean and does not have unwanted definitions, resulting in greater concentration at the time of a problem detection or network analysis.

Table 3.1: Benefits and sacrifices to the customer

Benefits		**Sacrifices**
Attributes	Outcomes	
1. Creation of the KPIs 2. Visualization of the KPI by NE, time and space	1. Only calculates the defined KPIs 2. Reduces the effort in the analysis of the network performance 3. Possible increase in the provided QoS 4. Possible increase in the number of network users	1. Adaptation to the new functionality 2. System's cost

Moreover, the customer takes advantage of the counters aggregation performed when visualizing the calculated KPI by any managed element in the network, period or space. Consequently, it is easier to detect configuration problems in network objects.

3.3 Value Proposition

The "value proposition element is an overall of a firm's bundle of products and services that together represent a value for a specific customer segment" (Osterwalder and Pigneur 2003). It must be credible and able to communicate the message succinctly and clearly. This element demonstrates the relationship between the customer's needs and the benefits of using the system.

This product is a solution to create and calculate the KPIs. With this framework, Vismon Manager will have a PM functionality, allowing the CSPs or network operators who desire to monitor and analyse the QoS of the network to achieve such requests. Performing aggregations in the performance counters on several levels, the solution can calculate the result for the KPIs by NEs or time.

This helps the customer to focus on the KPIs for success and monitor their value through time or country regions. In addition, it helps to understand the root problems that could be degrading the QoS.

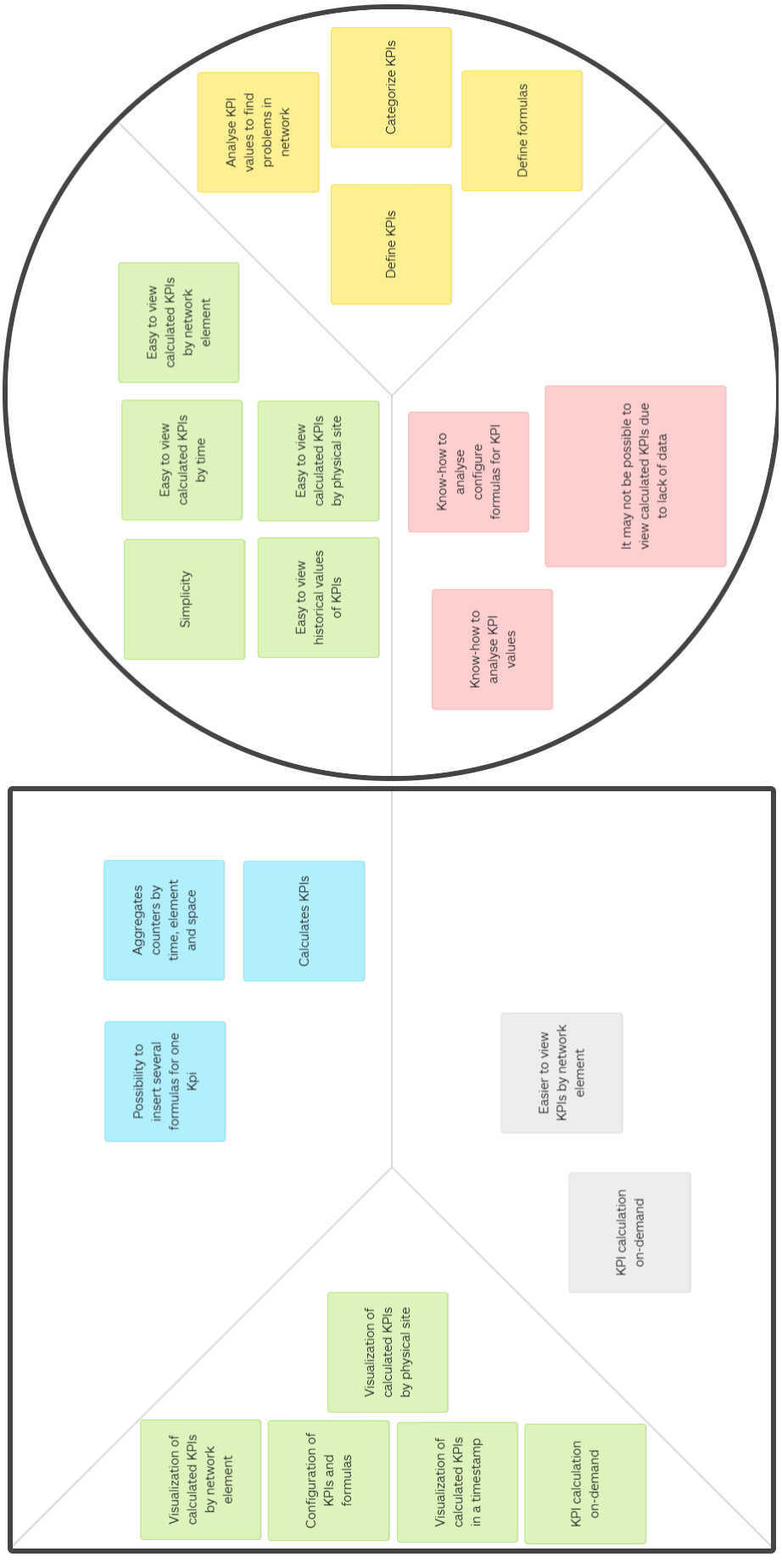


Figure 3.3: Value proposition

3.4 Quality Function Deployment

Quality Function Deployment (QFD) is a methodology that empathizes with the client's necessities or expectations and is translated into technical specifications. This "effectively communicates customer needs to multiple business operations throughout the organization including design, quality, manufacturing, production, marketing and sales" (International 2019).

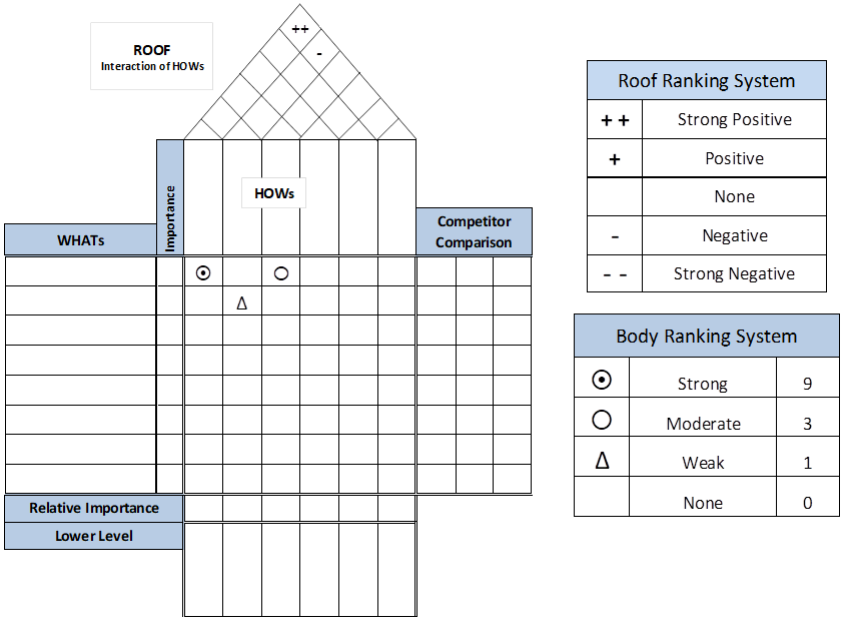


Figure 3.4: Quality Function Deployment - House of Quality example from Quality-One

The House of Quality (HoQ) allows a straight comparison between the product/project and the competition, according to Voice of Customer (VoC).

Looking at the QFD with the HoQ design tool, represented in figure 3.4, the customer's wants are described in WHATs and the design or technical requirements in HOWs. The customer needs have an associated importance factor. Within the room of the house, "the HOWs are ranked according to their correlation or effectiveness of fulfilling each of the WHATs" (International 2019), to indicate a strong, moderate or weak correlation.

In conformance with the project context, this method was developed to ensure quality and potential for the customer. Therefore, the HoQ was developed for the framework that calculates the KPI, represented in figure 3.5.

Relationships		Correlations		Direction of Improvement	
Strong	●	Positive	+	Maximize	▲
Moderate	○	Negative	–	Target	◇
Weak	▽	No Correlation		Minimize	▼

Row #	Weight Chart	Relative Weight	Customer Importance	Maximum Relationship	Customer Requirements (Explicit and Implicit) \ Functional Requirements	Import and process PM	Support different vendors and technologies	Allow the creation of formulas for KPIs	Visualization of KPIs in a time frame	Visualization of KPIs in any element	Adopt open source technologies	Our Product	Competitor #1: *NetPM*	Competitor #2: *Yuvo*	Competitor #3: *Vantage PM*	Row #
					Column #	1	2	3	4	5	6	Customer Competitive Assesment				
					Direction of Improvement	◇	▲	▲	◇	◇	◇					
1		26%	5	9	Easy to adapt to different telecom operators	●	●	○	▽	▽		5	5	5	5	1
2		21%	4	9	Customization of KPIs		○	●	○	○		5	5	3,5	3	2
3		26%	5	9	Calculate of KPIs	●	○	●	●	●		4	4	4	4	3
4		11%	2	3	Flexible and adaptable solution	○	○				○	4	4	4	4,5	4
5		16%	3	9	Low cost						●					5
					Target	Design and build a module to import, process and aggregate data	Design and build componentens that supports multi vendor/tech	Design and build a module to create KPIs and formulas	Design and build an interface that provides calculated KPIs by any period	Design and build an interface that provides calculated KPIs by an element	All technogies to use must be free					
					Max Relationship	9	9	9	9	9	9					
					Technical Importance Rating	505,3	410,5	505,3	326,3	326,3	173,7					
					Relative Weight	22%	18%	22%	15%	15%	8%					
					Weight Chart											
					Our Product	5	4,5	5	4	4	4					
					Competitor #1:NetPM	3	5	4,5	5	5	2					
					Competitor #2: Yuvo	3	5	3	4	4	2					
					Competitor #3: VantagePM	3	5	3	4	4	2					
					Technical Competitive Assesment											
					Column #	1	2	3	4	5	6					

Figure 3.5: Quality Function Deployment - House of Quality of the project

Chapter 4

Requirements and Domain Analysis

This chapter focuses on the business requirements analysis. Firstly, the details about the concepts and requirements are presented, with the domain model. Lastly, it presents the analysis of different approaches to calculate KPIs and the application of the AHP method.

4.1 Requirements

To allow the calculation of KPIs is necessary to load the PM data from the CSP. The processing of performance data has to pay attention to the common problems in performance dump files. They may arrive delayed, out-of-order, versioned (the same file is available with different counter values), incomplete, (partially) corrupted, duplicated, in pieces or do not arrive at all. These constraints generate a problem when performing a counter aggregation. PM data is always associated with a NE or a managed element.

A KPI can be calculated from raw or aggregated data. To calculate a KPI with aggregated counters, each one presented in the dump file must have an associated temporal and nodal type of aggregation. Frequently, the vendor's documentation contains the type of aggregation to apply in each counter. Also, it may incorporate information about spatial aggregations, since temporal and nodal aggregations are the most used. The system must have different retention times for the aggregated data. There are some criteria to take into account to aggregate performance counters and calculate KPIs, such as the velocity of the KPI calculation, the necessity of reprocessing aggregations and its complexity.

A KPI has a name, a description and a unit of measure. It can have many formulas associated. KPIs can also be grouped by category by associating a tag. A tag can be a simple word indicating a functionality/category as Accessibility or Performance, as explained in section 2.2.

A formula must be unique for a set of attributes: vendor, technology, release, resource and granularity period, to avoid ambiguity between formulas. Thereby, the system must only return zero or one formula for vendor/technology/release/resource/granularity period. A resource refers to an element in the network that belongs to a specific software release provided by the vendor.

To calculate the value of a KPI, a formula has an expression to be evaluated as a mathematical expression:

$$\frac{100 x attr_1}{attr_2} \text{ or } \log_2 x attr_2$$

There could be also logical expressions to evaluate KPIs, like `IF attr1 > 100 THEN 1 ELSE 0` however they are not included in this work.

The system must also allow the creation of different versions of the same formula, always ensuring that there are no duplicated formulas. To this end, there can be several formulas disable, yet there can only be one enabled formula.

A KPI value is calculated per NE and timestamp. In addition, it is possible to calculate a KPI for a group of NEs or a time interval. When calculating a KPI, a user must indicate the resource and the granularity period desired. For example, `Cell-Day` or `SubNetwork-week`. Besides, the user must have the possibility to filter and group the result by NE or timestamp/time interval.

The results returned to the user must contain the value of the KPI (if it is decimal, it must have two decimal places), the KPI unit, the related timestamp and network resource, the granularity period and temporal integrity. Nonetheless, the system must protect the amount of data returned to prevent overloading the system.

The technical documentation of the solution must be produced and include the analysis and design. The code must comply with the documentation.

4.1.1 Use Cases

To provide the design of the system in a standard way, all the diagrams presented in this document were developed using the Unified Modeling Language (UML) as the modelling methodology. Figure 4.1 illustrates use cases that the Vismon administrator and the ordinary user will perform on this system. Each use case is described in table 4.1.

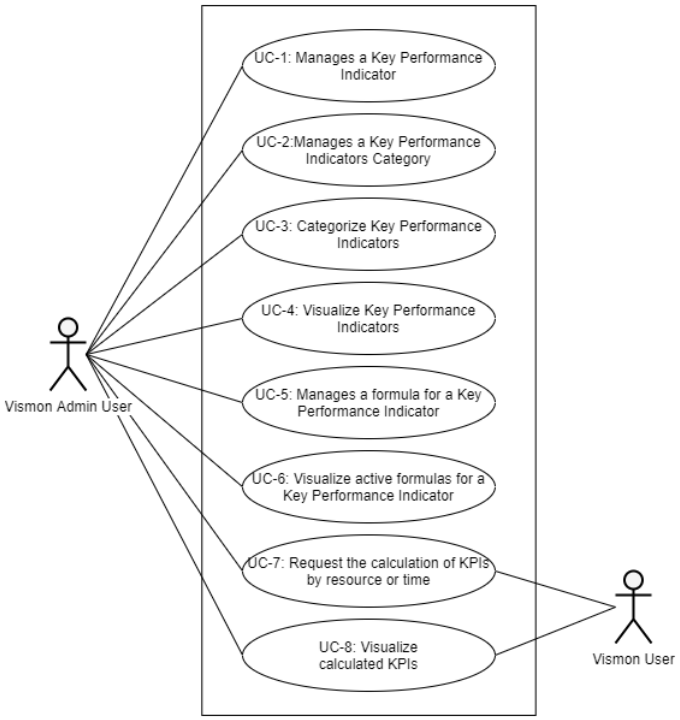


Figure 4.1: Use case diagram

Table 4.1: Use cases description

Use Case	Description
UC-1: Manages a KPI	The Vismon admin user adds or removes a KPI.
UC-2: Manages a KPIs category	The Vismon admin user adds or removes a category.
UC-3: Categorizes KPIs	The Vismon admin user adds or remove a category from a KPI.
UC-4: Visualizes KPIs	The Vismon admin user visualizes the created KPIs.
UC-5: Manages a formula for a KPI	The Vismon admin user adds or updates a formula for a KPI.
UC-6: Visualize active formulas for a KPI	The Vismon admin user visualizes the created formulas of a KPIs. The system returns 0 or 1 formula for a KPI, per resource and granularity period.
UC-7: Request the calculation of KPIs by resource or time	Vismon users can request the calculation of KPIs by resource and granularity period, with the possibility to apply filters and group the results.
UC-8: Visualize calculated KPIs	Vismon users can visualize calculated KPIs.

4.1.2 Quality Attributes

A quality attribute is a "measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders" (Bass, P. C. Clements, and Kazman 2013). Table 4.2 represents the quality attributes scenarios that are fixed for the system.

Table 4.2: Quality attribute scenarios

ID	Quality Attribute	Scenario
QA-1	Maintainability	It is necessary to add new features to the system. There must be a clear separation of responsibilities between the components for a faster update.
QA-2	Scalability	When there is an increase in demand for resources, the system is capable to handle then without an impact on the performance of the system.
QA-3	Manageability	The system logs its activity and events to a centralized database, in order to improve its monitoring and error detection.

4.1.3 Constraints

A constraint "is a design decision with zero degrees of freedom" (Bass, P. C. Clements, and Kazman 2013). Table 4.3 presents some design decisions that were already been made for this project by the Celfinet company.

Table 4.3: Constraints

ID	Constraint
C-1	Use open-source technologies in the development of the framework.
C-2	Use component Parameter Manager to configure attributes that need to be processed and the dump file or group that holds those attributes, destination database connection and type for the dump.
C-3	Use component Topology Manager to configure the network resources, with associated release, vendor and technology.

Using open-source technologies means that this project has a reduced cost, as there is no need to purchase licenses. Regarding the use of the Parameter and Topology Manager components, it reduces the effort to create a solution that adapts to various operators, vendors and technologies as they were developed to be generic and used in different projects.

4.1.4 Architectural Concerns

Architectural concerns "encompass additional aspects that need to be considered as part of the architectural design but that are not expressed as traditional requirements" (Cervantes and Rick Kazman 2016). Table 4.4 presents some architectural concerns that need to be considered as part of architectural design.

Table 4.4: Architectural constraints

ID	Architectural Concern
AC-1	Define an overall system structure, considering using the microservices architectural style as a base for the design.
AC-2	Each component, if needed, must have its own database.
AC-3	Asynchronous communication between services should be privileged.

These architectural concerns have been chosen by the Celfinet company so that the developed software meets the philosophy present in the software unit.

4.2 Domain Model

Figure 4.2 presents the conceptual model of the KPI creation and calculation framework, specifying the business concepts. This model contains all the concepts of the KPI domain.

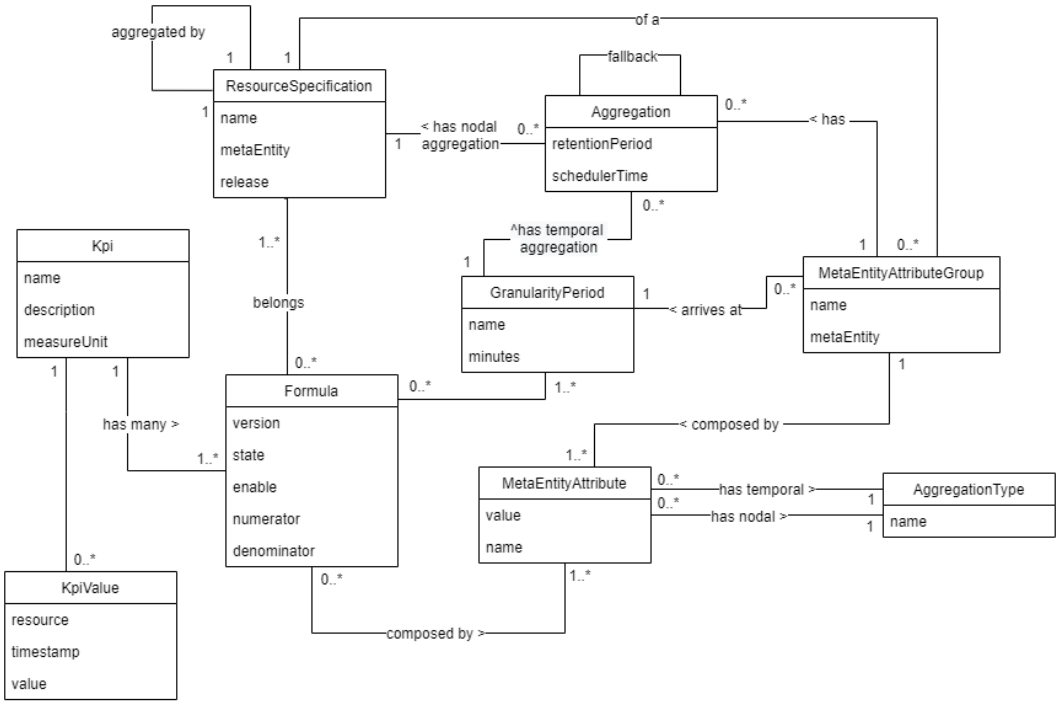


Figure 4.2: Domain model

A **ResourceSpecification** concerns the specification of a network resource, such as a cell, node, or controller. Each specification is unique for metaEntity[1] and release. A resource is an instance of **ResourceSpecification**, in other words, it is a NE. Each specification can have zero or more **MetaEntityAttributeGroup** configurated.

A **MetaEntityAttributeGroup** represents a set of counters that are configurated to be processed and imported. A **MetaEntityAttribute** represents, in this context, a PM measure. These attributes are composed of nodal and temporal aggregations types, for example, SUM, AVERAGE or MIN. This concept corresponds to the **AggregationType**.

A **Granularity Period** is a concept that defines the frequency of PM data. This concept is also used to define the granularity of each **Aggregation**.

A **MetaEntityAttributeGroup** can have many **Aggregation** definitions to perform aggregations in several levels of the network and periods. An **Aggregation** is composed of a ResourceSpecification, a GranularityPeriod, the retention period and the scheduler time. The **ResourceSpecification** defines the nodal aggregation of the data. The **Granularity Period** defines the temporal aggregation.

A **KPI** is characterized by its name, description and unit of measure. **KPI** can have one or more formulas however, there can be only one formula enable per **ResourceSpecification** and **Granularity Period** to eliminate possible ambiguities.

A **Formula** represents a mathematical expression to calculate the **KPI**. This expression is divided into numerator and denominator and these are composed of variables and operators.

A **KPIValue** represents the value of the **KPI** calculated following a request from a Vismon user or any other component. This value is related to a resource and a timestamp.

[1]MetaEntity is an abstraction for the representation of the vendor and tech of the resource specification.

The KPI domain it can be divided into three subdomains: KPI calculation, KPI creation and Performance data. Figure 4.3 shows the relation between the subdomains.

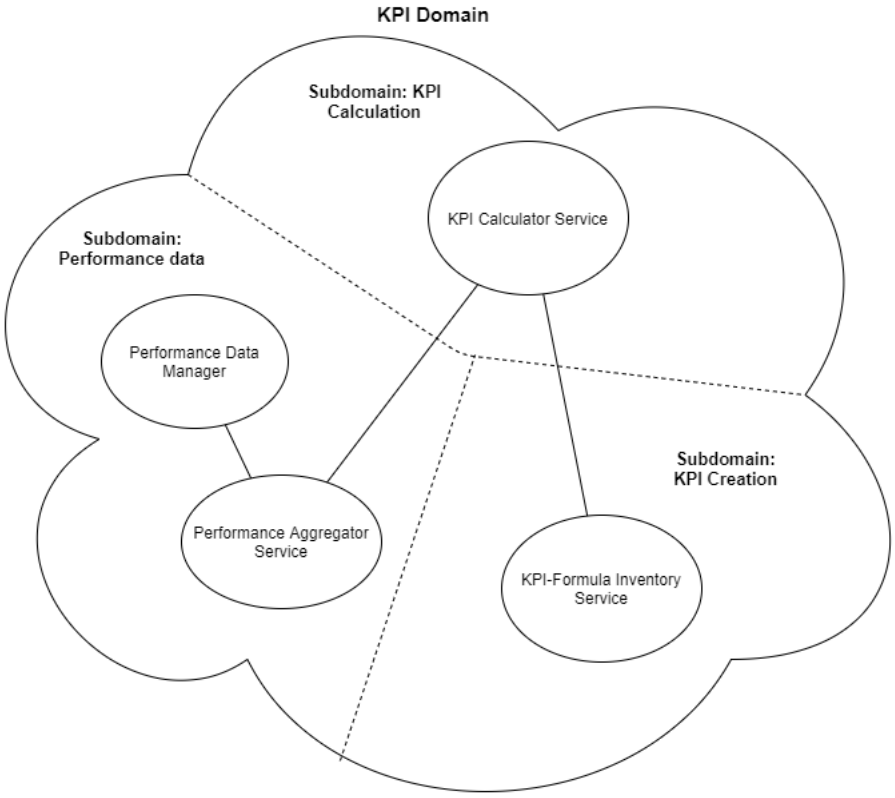


Figure 4.3: KPI Domain and subdomains

To illustrate the flow of control in use case 7 it was created an activity diagram. This diagram in figure 4.4 is related to the calculation of KPIs on the KPI Calculator service.

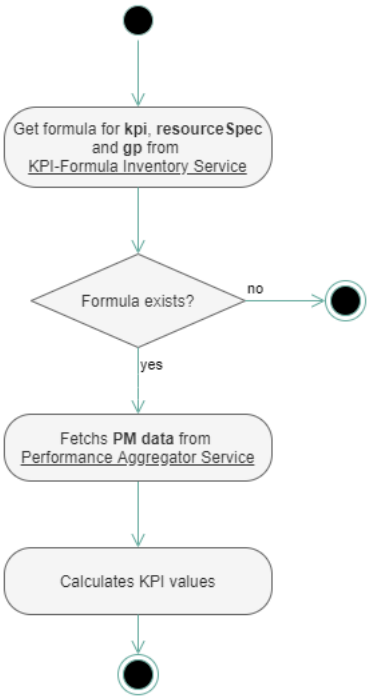


Figure 4.4: Activity diagram of the KPI calculation

4.3 Approaches to Aggregate Performance Data

The present section has, as focus, the analysis of approaches to aggregate PM data. These approaches also influence the calculation of KPIs. The AHP was applied to justify the choice of the most appropriate strategy.

For the network operators to analyse the QoS provided, they need to understand the meaning of the quality indicators. To do this, quality indicators have to be calculated for the NEs at a certain period.

The approaches were built taking into account some factors, such as the possibility to configure different retention periods for the processed data, the estimated velocity when calculating the KPIs and the possibility of the calculation being carried out with performance counters aggregated at different levels. Thus, to aggregate the data and calculate KPIs some approaches were designed.

The definition of the approaches was driven, not only but also to support the answer to the research question defined in section 1.3.

4.3.1 Approach 1 - Aggregations on-demand

This approach, represented in figure 4.5, has as base the aggregations of PM data in runtime when is necessary to calculate a KPI. Performance dump files are loaded into a database that contains all the processed data. These data refer to the performance counters that are presented in the dump.

Figure 4.5: Approach one to calculate KPIs

When is necessary to calculate a KPI, the necessary aggregations are executed in runtime and the calculation is performed. If the data set is too large the calculation of KPIs can be slow and take some time.

With this approach, when performance data arrives late, versioned or duplicate, the system only needs to check if the data exists for the correspondent timestamp and resource,and perform an update.

Regarding the possibility of configuring different retention times for different levels of elements in the network, it is not possible in this approach. As the data did not suffer any aggregation, all of them belong to the same level of resource specification and granularity period. Nonetheless, having one database with several historical data increases disk space because the minimum retention period is equal to the retention period of the highest NE level. For example, if the retention period is one year for the Network level, the system needs to keep all the imported data from that year.

4.3.2 Approach 2 - Pre-defined Aggregations

Like the previous approach, this also uploads the performance data from dump files into a database containing the processed counters. In this situation, there is a process that collects the processed data, performs the necessary aggregations and places the transformed data in another database as a table. This sequence reduces processing time when calculating a KPI because the counters are already aggregated and ready for consumption. Figure 4.6 represents the current approach.

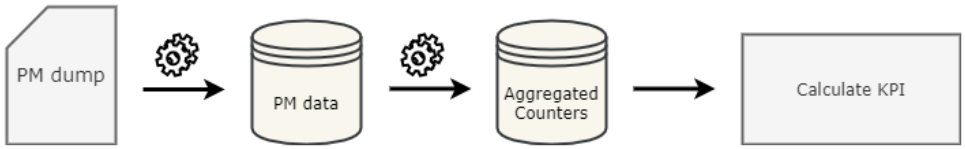


Figure 4.6: Approach two to calculate KPIs

When performance data arrives duplicated, we can assume that the latest data has the correct measured values. It means that the oldest data is presumed invalid. With this approach, if the data takes longer to arrive or arrives duplicated, some paths could be followed:

1. Discard the data if the aggregations have already been performed;
2. Configured aggregations are executed again for all levels and time intervals to include the new values. There must be a time interval in which inconsistencies are accepted in order not to overload the system;

3. Aggregations are configured to run after N minutes or N hours from the moment it is possible to do the aggregation. Inconsistencies are accepted until the defined period is completed.

In addition, it is possible to present different retention times for different types of aggregation. For example, a `SubNetwork-Week` aggregation can have a retention time of two years while a `Cell-Day` aggregation can have a retention time of one year. With this setting, it is possible to reduce disk space since the database that contains the processed counters does not need to have historical data.

Concerning Approach 1, this approach is a little more complex since there must be at least one more component to process the aggregations.

4.3.3 Approach 3 - Pre-defined and on-demand Aggregations

This approach also uploads the performance information into a database containing the processed performance counters. Alongside this, there is a process that collects data from N days ago and performs the necessary and possible aggregations, to be consumed when calculating the KPI. Consequently, it prevents the aggregations from being carried out with inconsistent data since it is only accepted up to N days after it arrives. Usually, similar applications accept inconsistencies in the raw data for up to three days. Figure 4.7 represents the current approach to calculate a KPI.

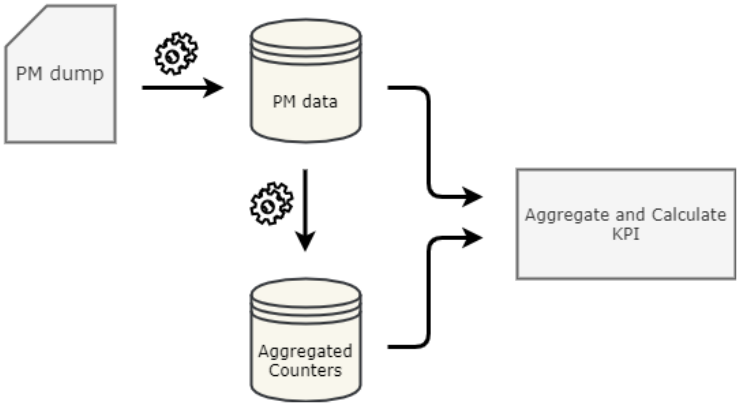


Figure 4.7: Approach three to calculate KPIs

The process for calculating a KPI is more complex than previous approaches, as it has to merge the volatile data with the already aggregated one. For example, if a user wants to know the value of a KPI for the current week, it is necessary to collect the data (not aggregated) from the last N days and make the necessary aggregations. Then, the data already aggregated is obtained, combined with the aggregations made, to obtain the final KPI value for the current week.

With this approach, updates or corrections of performance data are never ruled out and there is no need to reprocess aggregations.

Same as approach two, it also allows for different retention times for distinct types of aggregation.

4.3.4 Approach 4 - Pre-defined Aggregations and KPI Calculations

This approach is very similar to the second approach in that the data is already ready for consumption. Performance data is load into a database and a process collects that data, perform the necessary aggregations, calculates the KPIs and places the results in another database. This logic considerably reduces the calculation time since the KPIs are already calculated and ready for consumption. Figure 4.8 illustrates the current approach.

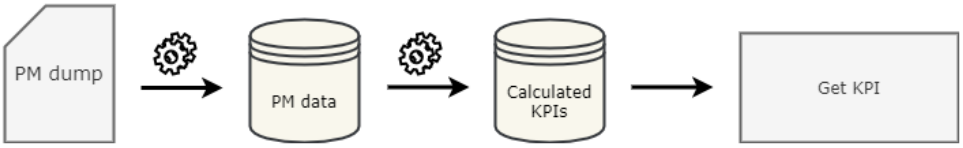


Figure 4.8: Approach four to calculate KPIs

Regarding data inconsistency, the possible options for approach 2 are also applied here. However, depending on the option chosen, it may be necessary to recalculate KPIs. In addition, as it is possible to define and change formulas for a KPI, the results obtained may become obsolete when changes occur.

This approach also allows different retention times for different types and periods of calculated KPIs, as the approaches 2 and 3.

4.3.5 Approaches comparison

The comparison between the approaches is represented in table 4.5 and was based on the following criteria:

1. Complexity of the solution - the ↑ stands for high complexity and the ↓ low complexity.
2. Configuration of different types of data retention - the ↑ allows the configuration. The ↓ does not allow it.
3. Aggregation of performance counters at runtime - the ↑ stands for the aggregation at runtime. The ↓ stands for pre-aggregations.
4. Estimated processing time when calculating KPI - the ↑ stands for a higher estimation of the calculation duration and the ↓ for a reduced duration.
5. Need to reprocess the aggregations performed due to data correction - the ↑ stands for the need for re-aggregating counters. The ↓ does not need to re-aggregate counters.

Table 4.5: Comparison of approaches by criteria

Approach	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5
Approach 1	↓	↓	↑	↑	↓
Approach 2	↓	↑	↓	↓	↑
Approach 3	↑	↑	↑	↑	↓
Approach 4	↓	↑	↓	↓	↑

Analysing table 4.5, approach 1 does not allow for different retention times and implies that the aggregations are executed on-demand. Approach 4, although it seems to be faster than approaches 2 and 3, can lead to obsolete data. Although approach 3 seems the most correct in terms of ensuring data consistency, it is quite complex. Consequently, it increases the development time. Thereby, it is concluded that approach 2 is the most indicated. This choice is justified in the next section.

4.3.6 Alternative Approaches

With the alternatives to the KPIs calculation presented above, the AHP method was explored to prioritize and select the approach to adopt.

The AHP method was developed by Saaty, and "is a powerful multicriteria decision-making tool that has been used in numerous applications in various fields of economics, politics and engineering" (Leal 2020). This method allows the use of qualitative criteria as quantitative in the evaluation process. Thus, it divides the decision problem into hierarchical levels to facilitate its understanding and evaluation. This method is composed of some phases explained below:

1. Set the principal purpose of the decision-making process - construction of the hierarchical decision tree;
2. Comparison between the hierarchy criteria by establishing priorities between the elements;
3. Relative priority of each criterion;
4. Evaluate the consistency of relative priorities;
5. Construction of the parity comparison matrix for each criterion, considering each one of the selected alternatives;
6. Obtain composite priority for alternatives;
7. Choice of the alternative.

It was decided to test the alternatives that best meet the criteria defined in table 4.5, corresponding to the comparison between approaches to calculating the KPIs. Thus, four criteria were selected to apply the method. Therefore, the hierarchical decision tree in figure 4.9 was elaborated, consisting of:

- **Objective** - Choose approach for aggregating data and calculatingKPIs;
- **Criteria** - Estimated Processing Time, Different retention times, Need for reprocessing and Development time.
- **Alternatives** - Approach 1, 2, 3 and 4.

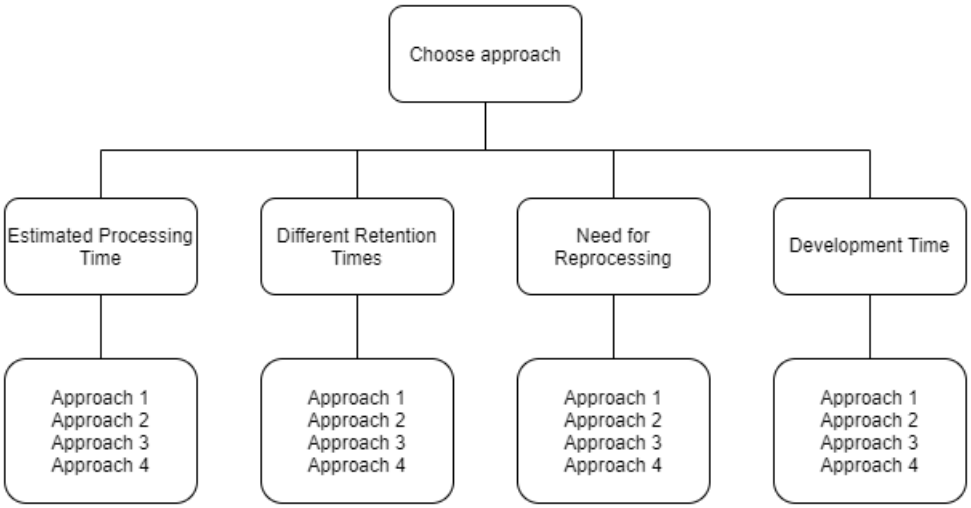


Figure 4.9: Hierarchical approach decision tree

To compare the criteria table 4.6 was constructed, containing the related priorities. This concerns phase 2 of the method applied.

Table 4.6: Comparison matrix of the second approach criteria level

	Estimated Processing Time	**Different retention times**	**Need for reprocessing**	**Development time**
Estimated Processing Time	1	2	4	1/2
Different retention times	1/2	1	2	1/4
Need for reprocessing	1/4	1/2	1	1/6
Development time	2	4	6	1

For phase 3, the standardized matrix of criteria was constructed and the relative priorities for the criteria were calculated and are presented in Annex A, section A.1. Relative priorities are summarized in the following list:

- **Estimated Processing Time** - 0.28;
- **Different retention times** - 0.14;
- **Need for reprocessing** - 0.07;
- **Development time** - 0.51.

From the results obtained, it is possible to observe that the first criterion is the development time, followed by the estimated processing time and different retention times, ending with the need for reprocessing.

In phase 4, the Consistency Ratio (CR) of the approach criteria is calculated to assess the consistency of judgments concerning large samples of completely random judgments.

With the calculations present in Annex A, section A.2, it is observed that the CR value is less than 0.1. So it is possible to conclude that the values of the relative priorities of the approach criteria are consistent.

In phase 5, the parity comparison matrices for each criterion were built taking into account the alternatives previously chosen. Phases 5 matrices are presented in Annex A, section A.3. The priorities from the comparison for each criterion, calculated in phase 5, are presented in figure 4.10.

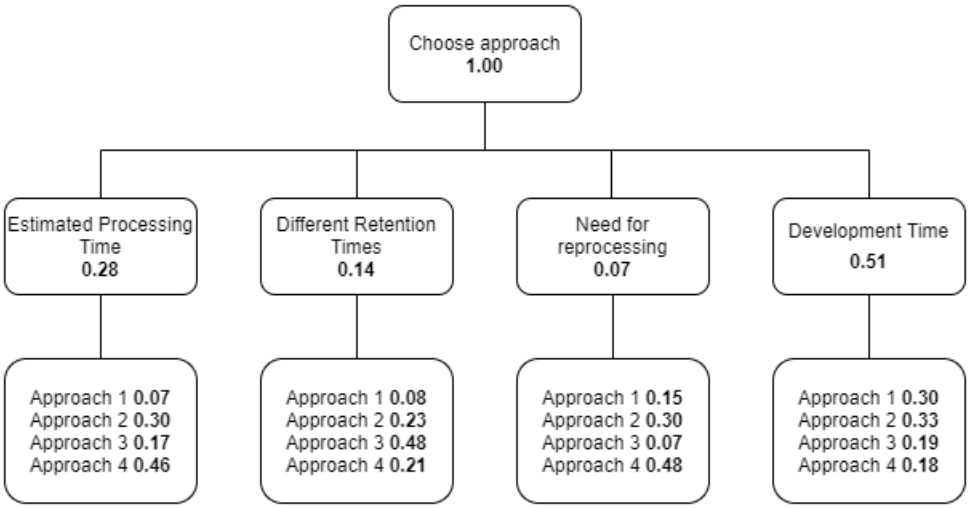


Figure 4.10: Hierarchical approach tree resulting from calculations

With the priorities from the comparison (see Appendix A), its built the priority that corresponds to the result between the multiplication of the priority matrix and the weights criteria.

$$\begin{bmatrix} 0.12 & 0.09 & 0.15 & 0.30 \\ 0.30 & 0.23 & 0.30 & 0.33 \\ 0.17 & 0.48 & 0.07 & 0.19 \\ 0.46 & 0.21 & 0.38 & 0.18 \end{bmatrix} x \begin{bmatrix} 0.17 \\ 0.29 \\ 0.23 \\ 0.31 \end{bmatrix} = \begin{bmatrix} 0.17 \\ \mathbf{0.29} \\ 0.24 \\ 0.28 \end{bmatrix} \tag{4.1}$$

Approach 2 appears as the most indicated approach for aggregate PM data and calculating KPIs according to the defined criteria and respective importance.

Chapter 5

Architectural Design

This chapter presents the architectural drawing of the proposed system as a solution. To satisfy the requirements presented in chapter 4, the design of the system was done using Attribute Driven Design (ADD) method.

The ADD is based on iterations, and at the end of each iteration, a Kanban board is present to summarize the state of the requirements: Not Addressed, Partially Addressed and Completely Addressed.

5.1 Attribute Driven Design

This method "provides detailed, step-by-step guidance on the tasks that have to be performed inside the design iterations" (Cervantes and Rick Kazman 2016). There are no defined amount of iterations to perform, but ideally, more than one should be considered.

ADD follows a recursive design that allows decomposing every element of the system applying architectural patterns that satisfy the drivers. Next, it is provided with an overview of the steps and artefacts that are associated.

Step 1: Review Inputs
Initially, it is necessary to verify if there is sufficient information about the primary functional requirements, quality attribute scenarios, architectural constraints, and concerns. The requirements need to be prioritized according to established goals and quality attributes and ordered by priority to determine the elements to focus on during iterations (Cervantes and Rick Kazman 2016).

Step 2: Establish the Iteration Goal by Selecting Drivers
Since this method is based on an iterative approach, each iteration focuses on achieving a specific goal. Typically, a goal involves designing to satisfy a subset of drivers. "For this reason, when performing design, you need to establish a goal before you start a particular design iteration" (Cervantes and Rick Kazman 2016).

Step 3: Choose One or More Elements of the System to Refine
To satisfy the drivers it is required the production of one or more architectural structures. These structures are composed of interrelated elements that are usually obtained by refining others previously identified. (Cervantes and Rick Kazman 2016).

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers
When choosing a design concept, one should require the identification of alternatives that

can be used to achieve the iteration goal. The alternatives should be indicated, as well as their benefits and constraints.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces
Once the design concepts are selected, it is necessary to instantiate elements. After this instantiation, responsibilities need to be allocated to each element. The elements that have been instantiated also need to be connected to allow collaboration. These relationships between the elements allow the exchange of information through some interfaces that need to be specified (Cervantes and Rick Kazman 2016).

Step 6: Sketch Views and Record Design Decisions
At this step, the design decisions are done for the iteration. Nevertheless, actions may not have taken to ensure that the views are preserved. The views need to be stored to be revisited in the following iterations. Besides the views, the reasons for the design decisions that were made during the iteration need to be recorded to facilitate later analysis.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose
At this step, a partial design was created that addresses the goal for the iteration. The review of the iteration is important "to avoid unhappy stakeholders and later rework" (Cervantes and Rick Kazman 2016).

For the design of the necessary views of the system, it was employed Krutchen's 4 + 1 view model, used to describe and demonstrate the system from various perspectives (Kruchten 1995):

1. **Logical View** - supports the functional requirements, in other words, the services that the system should provide and relationships between the software components. A component diagram can be used to represent this view;
2. **Process View** - captures the process, flow of information, demonstrates the responsibilities and collaborations of the logical elements. A sequence diagram is used to illustrate this view;
3. **Development View** - focuses on the organization of the software modules of an application and its concerned with software management;
4. **Deployment/Physical View** - takes into account the physical deployment of processes and components, as well as the connections between these components. Deployments diagrams are used to illustrate this view;
5. **Scenarios** - uses a small set of use cases to summarize the architecture description, describing sequences of interactions between objects and processes.

5.2 ADD Step 1: Review Inputs

This system is a greenfield system since it is a project to develop for a new environment. Thereby, the purpose is to produce a design with enough detail to support the construction of the system.

In this step, the inputs were reviewed and prioritized to be considered as drivers for the design. The requirements were prioritized considering two dimensions: the importance of

the requirement concerning the success of the system and the degree of technical risk associated with the scenario. The importance of the requirements was defined as having in account a logic sequence to have the KPIs calculated and to meet Celfinet's objectives regarding software standards.

The chosen rate of importance has three values: **H** (high), **M** (medium), and **L** (low). Requirements are prioritized in tables 5.1, 5.2, 5.3 and 5.4.

Table 5.1: Prioritized use cases

Use Case	**Importance**	**Technical Risk**
UC-1: Manages a KPI	H	M
UC-2: Manages a KPIs Category	L	M
UC-3: Categorize KPIs	L	M
UC-4: Visualize KPIs	M	L
UC-5: Manages a formula for a KPI	H	H
UC-6: Visualize active formulas for a KPI	M	L
UC-7: Request the calculation of KPIs by resource or time	H	H
UC-8: Visualize calculated KPIs	M	L

Table 5.2: Prioritized quality attributes

Scenario ID	**Importance**	**Technical Risk**
QA-1: Maintainability	H	H
QA-2: Scalability	M	M
QA-3: Manageability	L	L

Table 5.3: Prioritized constraints

Constraint ID	**Importance**	**Technical Risk**
C-1: Open-source technologies	M	L
C-2: Use component Parameter Manager	H	M
C-3: Use component Topology Manager	H	M

Table 5.4: Prioritized architectural concerns

Arquitetural Concern ID	**Importance**	**Technical Risk**
AC-1: Define an overall system structure, considering using the microservices architectural style as a base for the design	H	H
AC-2: Each component, if needed, must have its own database	H	M
AC-3: Asynchronous communication between services should be privileged	M	M

These prioritized drivers serve as input for the next presented iterations. It is not necessary to prioritize drivers in all iterations, as they have been prioritized here.

5.3 Iteration 1: Establishing an Overall System Structure

In this section, the results of the activities performed in each of the steps of ADD are presented regarding the first iteration of the design process.

Step 2: Establish the Iteration Goal by Selecting Drivers

The goal of this iteration is to establish a high-level design of the system structure. According to the defined goal and since it is the first iteration, all drivers were taken into consideration, especially architectural concerns and constraints.

Step 3: Choose One or More Elements of the System to Refine

Considering that this is a development for a greenfield system, the element to refine is the entire system. The refinement is performed through decomposition. Figure 5.1 represents the context of the system.

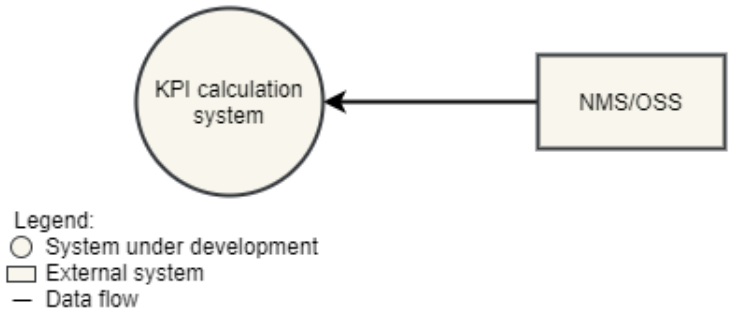


Figure 5.1: Context diagram for KPI calculation system

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

To start structuring the system, it is necessary to select architecture styles or patterns to guide the construction of the system. Besides, a base deployment design and the selection of the technologies to use were selected. Next, design decisions are summarized.

The logical structure of the system

Rationale
The microservices architecture style (see section B.1) improves the maintainability of the system (QA-1) since this style is based on the development of small and autonomous services modelled around the KPI domain. Besides, small and independent services allow scaling only the services that need to be scaled (QA-2). Also, in this style, each service should have its

private data, in other words, it has its database (AC-2). Furthermore, this pattern should be used (AC-1).

Discarded Alternatives
The layered architecture pattern (see section B.1) is a well-known pattern and simple to develop. However, applications that are implemented with this pattern are difficult to scale and difficult to change when it is necessary to change something in the system.

Microkernel architectural pattern (see section B.1) was discarded, mainly due to the complexity of development and the lack of knowledge about this pattern. Also, most implementations of this pattern are single units which makes it difficult to scale what is necessary.

The physical structure of the system

Rationale
Microservices architecture pattern was applied to architect the system as a group of services. To deploy the system, a service instance per container was chosen because "it is straightforward to scale up and down a service by changing the number of container instances" (Richardson 2020b), isolating each service instance.

Discarded Alternatives
To deploy a system based on microservices architecture, there are some patterns besides the service instance per container, as multiple service instances per host, single service instance per host and service instance per Virtual Machine (VM) (Richardson 2020a).

Multiple service instances per host were discarded since it is not possible to isolate one instance and it is not straightforward to scale. A single service instance per host can be less efficient in resource consumption since it is multiple hosts are created. In addition, the pattern service instance per VM was also discarded because is not intended to created VMs (building an image of a VM can be slow).

Technologies

Rationale
Develop all services using Spring Boot with Java Persistence API (JPA), since are open source technologies (C-1). Spring makes Java programming simple and quicker and has built-in features for the development of microservices. Perform asynchronous communication between the components using Apache Kafka.

PostgresSQL is a relational open-source database and is used to persist formulas and performance data. Besides this, logging entries are saved in a PostgresSQL database to simplify.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The system is composed of six components to fulfil its requirements. Each component corresponds to a microservice. The following list contains the services that were already developed by Celfinet company:

- **TopologyManager** - this component is responsible to configure the network elements specifications with the associated release, vendor and technology and their relationships. With this component, system supports multiple vendors and technologies, making it generic. Each resource specification is identified by its name, its metaEntity and release.
- **ParameterManager** - this component provides information about the counters defined for processing, as well as the configured granularity period and the mathematical operations of each aggregation (sum, average, ...). The set of counters is configured as a MetaEntityAttributeGroup and a counter as a MetaEntityAttribute. Each metaEntityAttributeGroup is associated with a resource specification which is configured in `TopologyManager`.
- **PerformanceDataManager** - the responsibility of this component is to import and save PM data.

With the support of the services described above and the subdomains identified in section 4.2, this system contains three more components to satisfy the drivers. The responsibilities are described in the following list:

- **KPIFormulaInventory** - this service is responsible for managing KPIs and formulas. Without formulas configured in the inventory, it is not possible to calculate KPIs. It also allows the categorization of KPIs. This component reflects the service identified in the KPI Creation subdomain.
- **PerformanceDataAggregator** - this component is responsible for executing pre-defined aggregations with PM data. This service knows which aggregations to apply on each MetaEntityAttribute and how to employ them. Besides this, it also has the responsibility of obtaining PM data to calculate KPIs. This component reflects the Performance Aggregator Service identified in the Performance data subdomain.
- **KPICalculator** - with this component it is possible to calculate KPIs in any granularity period and by any resource. The component needs to know which formula to use and for that, he needs to communicate with the `KPIFormulaInventory`. Knowing the formula, it is necessary to know the value of the mathematical expression involved, so it consumes information from `PerformanceDataAggregator`. This communication is asynchronous. This component reflects the service identified in the KPI Calculation subdomain.

Each service exposes an Application Programming Interface (API) based on the Representational State Transfer (REST) architecture style. The API allows the services to consume information from other services when it is intended.

Regarding the deployment of the system, there will be two servers to allocate the components:

- **Application Server** - this server is responsible to hold the application containers.
- **Database Server** - this server contains all the databases that are necessary for the correct functioning of the application.

Logging

To log the relevant information in a centralized database, a simple data structure has been defined to standardize the format.

- **Request identification** - this field was included to possible correlate the log entry to any data or request.
- **Component** - this indicates the source of the messages, as `KPICalculator` or `KPI FormulaInventory`.
- **Timestamp** - the instance of the log entry, example: 2021-04-21 10:00:00.000.
- **Level** - this field identifies the level of the log, such as Info, Debug Warning, Error, among others.
- **Message** - a simple phrase that sums the log.
- **Details** - an optional field that provides more detail about the message, such as stack trace errors.

Step 6: Sketch Views and Record Design Decisions

Figure 5.2 shows the logical view of the system, having into account the defined components in the previous step. Regarding the relationship between the components, the utilized communication protocols are the HyperText Transfer Protocol (HTTP) in synchronous communication and the TCP network protocol in asynchronous communication.

The message broker component represents the intermediary that forwards the messages between the components. The communication with the databases is done using the Java Database Connectivity (JDBC) protocol.

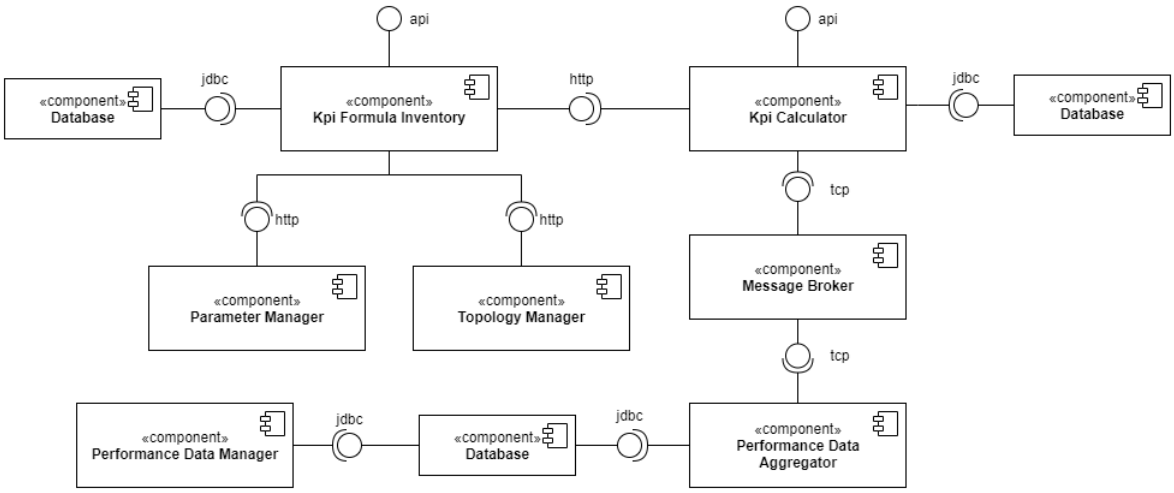


Figure 5.2: Logical view of the system

The deployment diagram represented in figure 5.3 shows the allocation view containing the previously defined components.

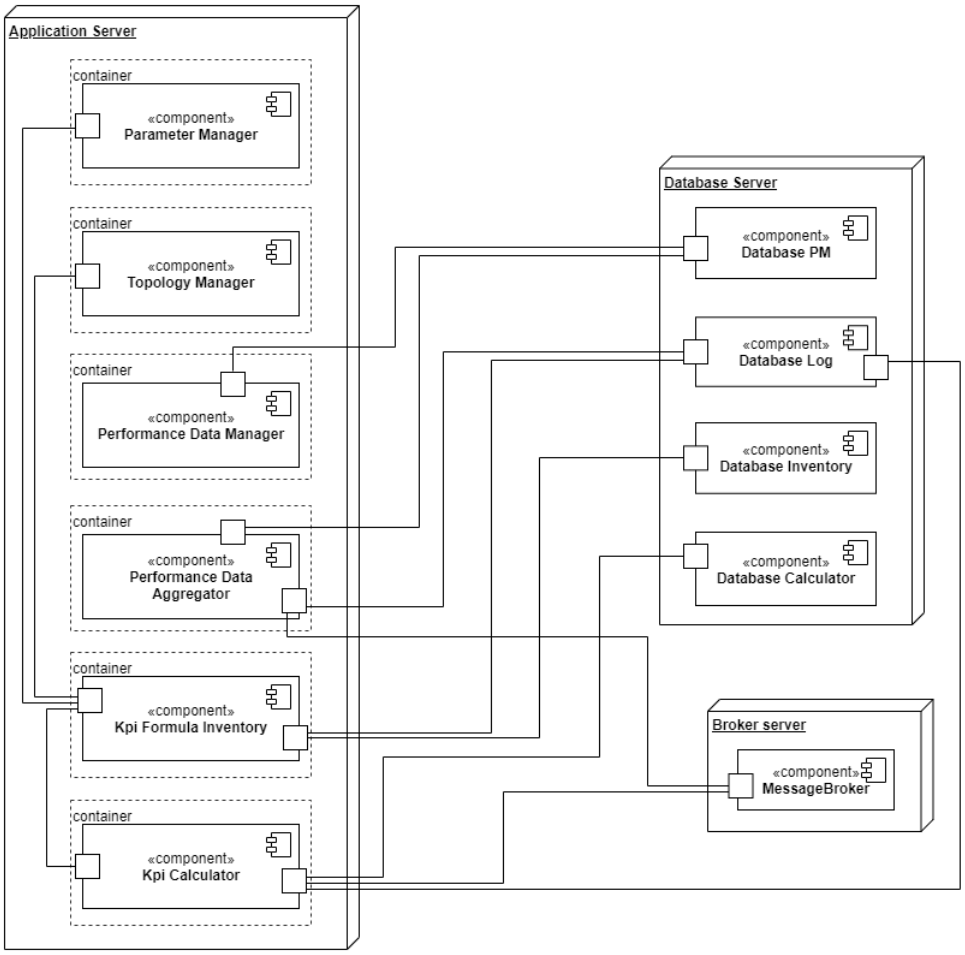


Figure 5.3: Deployment diagram of the system

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.2, presented in Annex B represents a Kanban board that sums the design process. The items QA-1, QA-2 and AC-1 were completely addressed in this iteration.

5.4 Iteration 2: Support creation of KPIs and formulas

In this section, the results of the activities performed in each of the steps of ADD are presented regarding the second iteration of the design process.

Step 2: Establish the Iteration Goal by Selecting Drivers

The goal of this iteration is to support the use cases associated with the creation of KPIs and formulas by identifying structures and establishing the architectural design. These use cases include the UC 1 to 6, defined in table 4.1

- UC-1: Manages a KPI
- UC-2: Manages a KPIs Category
- UC-3: Categorizes KPIs
- UC-4: Visualizes KPIs
- UC-5: Manages a formula for a KPI
- UC-6: Visualize active formulas for a KPI

Although UC7 - Request the calculation of KPIs by resource or time - has a higher priority than some of the selected use cases for this iteration, it is dependent on creating formulas and KPIs. Thus, UC7 was addressed in iteration 3.

Step 3: Choose One or More Elements of the System to Refine

In this iteration, the element to refine is the `KPIFormulaInventory`, since this is where the KPIs, formulas and their categories or tags will be stored.

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

In the previous iteration, it was defined the architecture pattern and the technologies used for the development of the microservices.

The internal structure of the component

Rationale
`KPIFormulaInventory` is structured based on a layered architecture to have a clear separation of concerns, low coupling and high cohesion between the software elements.

Domain model concepts to objects

Rationale
The domain model is present in section 4.2 and represents the business concepts that are candidates to become entities. With this model, concepts related to KPIs and formulas are mapped into system's classes with the attributes, relationships and operations.

Data persistence

Rationale
`KPIFormulaInventory` has its private relational database to store its data. JPA specification is used to define how to persist the entities in the database and their relationships in an easier way. For this, it is necessary to use annotations in Java objects. JPA annotations have agree with the relationships presented in the domain model.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

`KPIFormulaInventory` is composed of three internal and logical layers that are the most commons and are presented in the following list:

- **Facade layer** - this layer provides an abstraction of the system to external clients and accesses methods presented in the Service layer.
- **Service layer** - this layer contains the business logic necessary to perform the use cases related to the management of KPIs, formulas and tags. This layer access to methods presented in the Repository layer.
- **Repository layer** - this layer contains the logic to create, read, update and remove KPIs, formulas and tags from the database, using the JPA framework.

Step 6: Sketch Views and Record Design Decisions

Figure 5.4 exposes the `KPIFormulaInventory` component that is being refined in the current iteration.

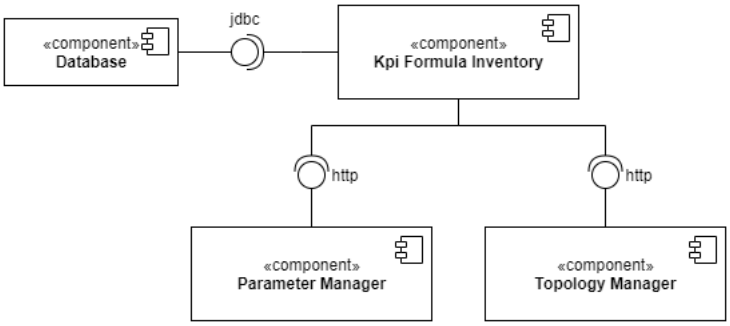


Figure 5.4: Logic view of KPIFormulaInventory

UC-1: Manages a KPI

The sequence diagram presented in figure 5.5 represents the process view of adding a KPI in the `KPI Formula Inventory`.

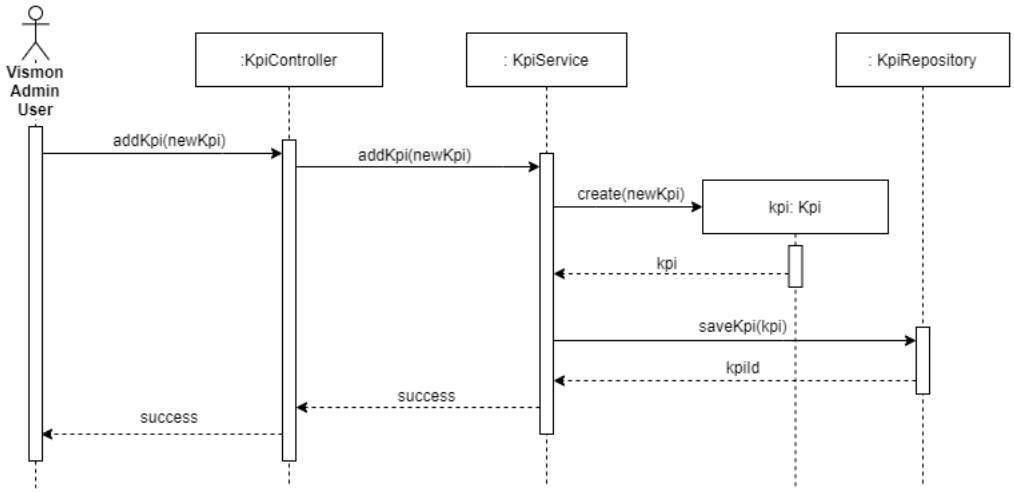


Figure 5.5: Sequence diagram for adding a KPI

The sequence diagram presented in figure B.4, in Annex B, represents the process view of removing a KPI. When removing a KPI, all the formulas must be removed since they can only exist if the KPI exists.

UC-2: Manages a KPIs Category

The sequence diagram presented in figure 5.6 represents the process view of adding a tag in `KPIFormulaInventory`.

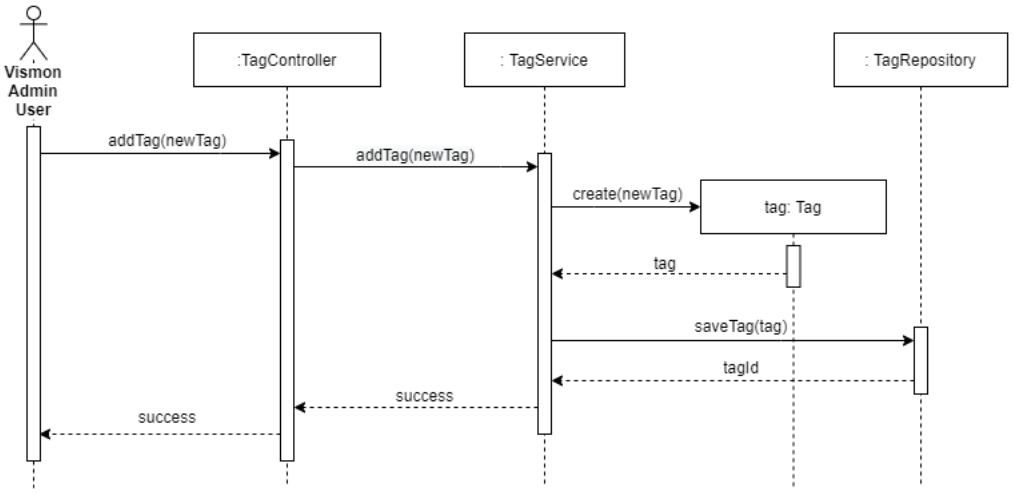


Figure 5.6: Sequence diagram for adding a tag

The sequence diagram presented in figure B.5, in Annex B, represents the process view of removing a tag. When removing a tag it is also necessary to eliminate the associations with the KPIs.

UC3: Categorizes KPIs

The sequence diagram in figure 5.7 represents the process view of how to classify a KPI by associating a tag.

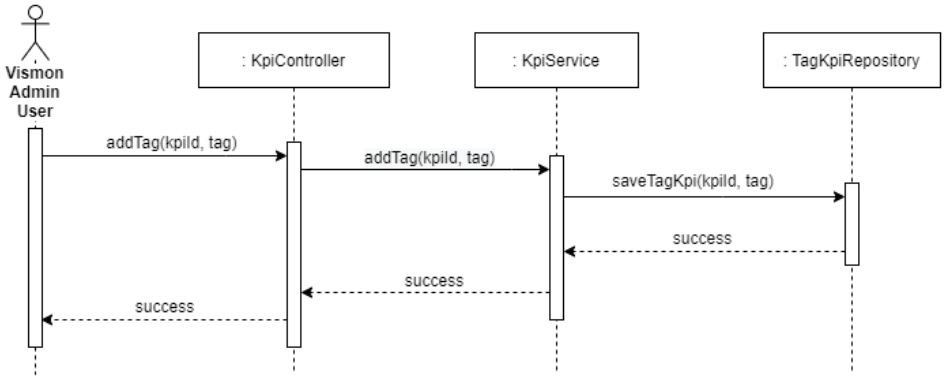


Figure 5.7: Sequence diagram for tag a KPI

UC4: Visualizes KPIs

The sequence diagram in figure 5.8 represents the process view to get all the KPIs created.

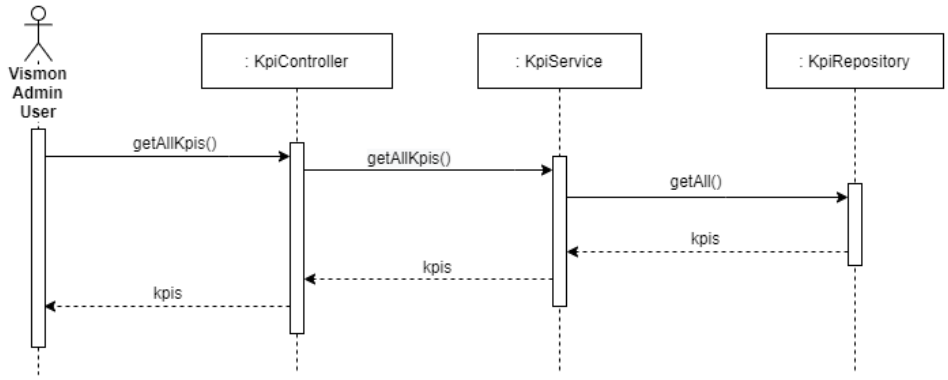


Figure 5.8: Sequence diagram for get all KPIs

UC-5: Manages a formula for a KPI

The sequence diagram presented in figure 5.9 represents the process view of adding a mathematical formula to a KPI in `KPIFormulaInventory`. In the universe of KPI formulas, mathematical formulas are the most common. Therefore, the process of calculating a KPI was designed with this type of formula.

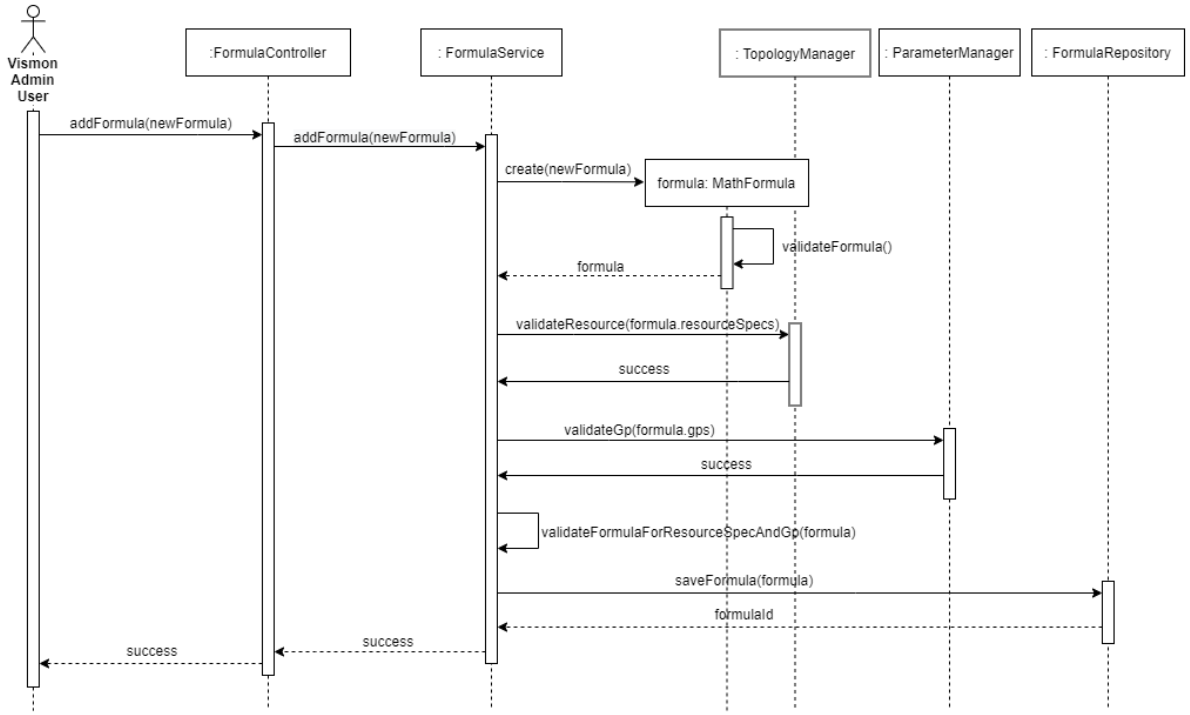


Figure 5.9: Sequence diagram for adding a formula

The following list explains what is the purpose of the specified methods in figure 5.9:

- `validateResource(resourceSpecs)` - this method has the responsibility to check if the formula's resources are valid and if they have an aggregation relationship, as explained in section 2.1.1. This validation is done in `TopologyManager` since this

component knows which resources are configured and their aggregations. Each resource specification is composed by its name, its metaEntity (corresponds to a unique vendor and tech identifier) and release.

- `validateGp(gps)` - this method has the responsibility to check if the formula's granularity periods are valid. This validation is done in `ParameterManager` since this component knows which granularity periods are valid.
- `validateFormulaForResourceSpecAndGp(formula)` - this method essentially verifies if the formula is unique for the KPI, resource specification and granularity period. The system must always ensure that there is no ambiguity between formulas, so when necessary, always return 0 or 1 formulas.

The communications between the `ParameterManager` and `TopologyManager` were defined as synchronous since no data is updated and these components only need to check data in the database.

UC-6: Visualize active formulas for a KPI

The process view of getting all active formulas for a KPI is present in the sequence diagram of figure 5.10.

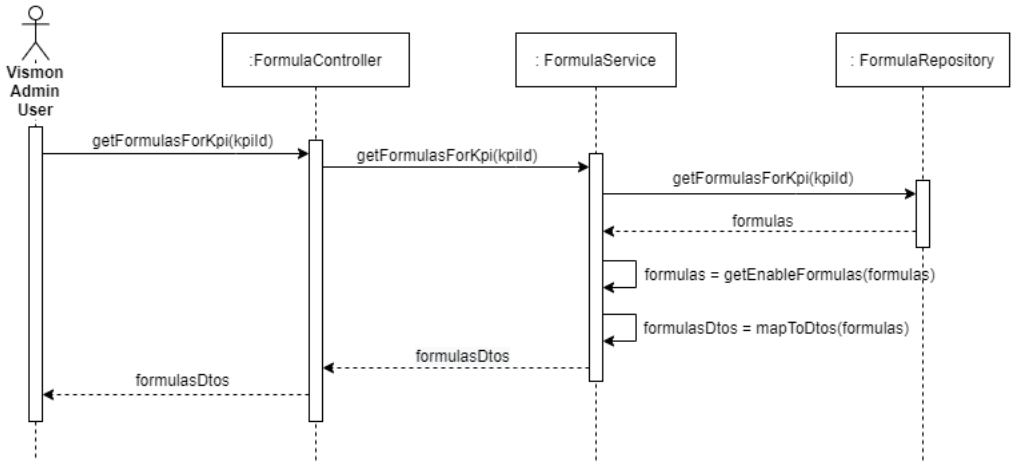


Figure 5.10: Sequence diagram for getting a KPI's formulas

Database

All the data produced in `KPIFormulaInventory` is persisted in a relational database. Figure 5.11 represents the data model of this service.

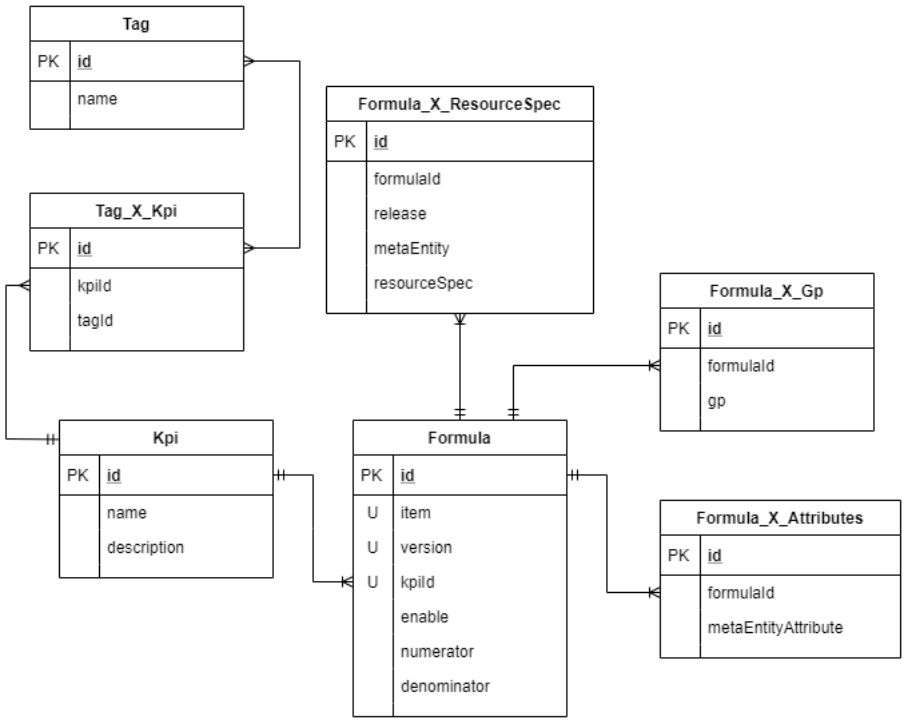


Figure 5.11: Data model of the KPIFormulaInventory

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.3, presented in Annex B represents a Kanban board that sums the design process. The items UC-1 to 4 and UC-6 were completely addressed in this iteration.

5.5 Iteration 3: Support the calculation of KPIs

In this section, the results of the activities performed in each of the steps of ADD are presented regarding the third iteration of the design process.

Step 2: Establish the Iteration Goal by Selecting Drivers

The goal of this iteration is to support use case 7 and 8 - Calculate KPIs and visualize calculated KPIsIn addition, this iteration also supports the remain quality attributes and constraints, the identification of structures and establishing the architectural design for this use case.

Step 3: Choose One or More Elements of the System to Refine

In this iteration, the elements to refine are the `KPICalculator` and the `PerformanceData Aggregator` which were identified in iteration 1.

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

To calculate a KPI, some approaches were defined and are presented in section 4.3. Therefore, the design for this use case was performed considered the chosen approach, Approach 2 with pre-defined aggregations (see section 4.3.2).

The internal structure of the components

Rationale
`KPICalculator` and `PerformanceDataAggregator` components follow the same logic as the `KPIFormulaInventory`, in other words, its internal structure is based on a layered architecture.

Domain model concepts to objects

Rationale
The domain model is present in section 4.2 and represents the business concepts that are candidates to become entities. With this model, concepts related to aggregations are mapped into the system's classes with the attributes, relationships and operations.

Tactic to aggregate PM data

Rationale
Aggregations are configured to be executed periodically at a certain time. When the aggregation is being executed, it only uses a set of data of a certain instant. This instant is calculated by subtracting N amounts of the granularity period defined. For example: if the aggregation is configured for the granularity period **hour** and **N=2**, then the instant to be aggregated is: **current date minus 2 hours**.

The aggregation of the lowest resource specification level is configured to run periodically for each granularity period intended. When the configured aggregation is executed it triggers the aggregations that are dependant on this one. An example is represented in figure 5.12.

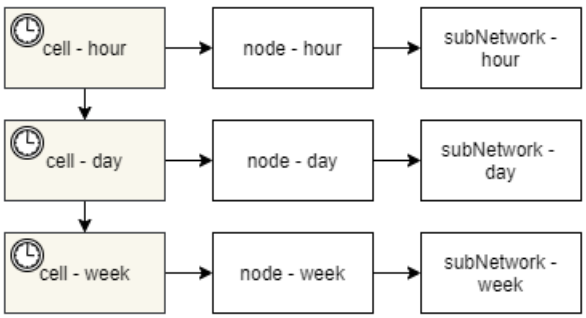


Figure 5.12: Example of aggregations to execute

Discarded Alternatives
The discarded alternatives to aggregate PM data are presented in section 4.3.2, alongside the chosen one. These were discarded because were considered more complex to include in the PoC.

Tactic to calculate KPIs

Rationale
The calculation of a KPI involves mainly two components: `KPICalculator` and `Performance DataAggregator`. The `KPICalculator` provides the formula, expressed as numerator and denominator, to the `PerformanceDataAggregator` component. This one builds a query that gives back the calculated numerator and denominator from the database and passes the information back to the `KPICalculator` to finish the calculation. The communication between these services is asynchronous and simple events are registered to keep track of the calculation process.

Discarded Alternatives
Provide the attributes that are part of the formula expression to the `KPICalculator` component could increase the memory consumption when messages are being exchanged since the formula can include dozens of attributes.

Provide the attributes or the formula numerator and denominator to the `KPICalculator` component in a synchronous call could increase the memory consumption and blocking the component when waiting for the response of the `PerformanceDataAggregator`.

Data persistence

Rationale
`PerformanceDataAggregator` has its private schema in a relational database to store the performed aggregations and the configurations of the pre-defined aggregations. There was a need to make a trade-off between the complexity and the time spent when aggregating data and having a private database for `PerformanceDataAggregator`. With this, `PerformanceDataManager` assures that `PerformanceDataAggregator` has only read permissions of the necessary data to create the aggregations.

`KPICalculator` has its private database to store the calculated KPIs and calculation events.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

`KPICalculator` is responsible to calculate KPIs with associated temporal integrity and save them in a database. Each calculation has an identification. This identity allows the service to know which data belongs to a certain calculation request.

`Performance Data Aggregator` has the responsibility to pre-aggregate PM data. He also provides the aggregated data to the `KPICalculator` when is necessary to perform calculations.

`KPICalculator` and `PerformanceDataAggregator` are composed of three layers. The layers and their responsibilities are presented in the following list:

- **Facade layer** - provides the necessary methods to be consumed by external clients.
- **Service layer** - this layer contains the business logic necessary to perform the use cases.
- **Messaging Layer** - this layer provides the necessary methods to subscribe and publish messages in Kafka topics.

- **Repository layer** - this layer contains the logic to execute aggregations in the database and save relevant information using the JPA framework.

Step 6: Sketch Views and Record Design Decisions

Figure 5.13 exposes the components that are being refined in the current iteration.

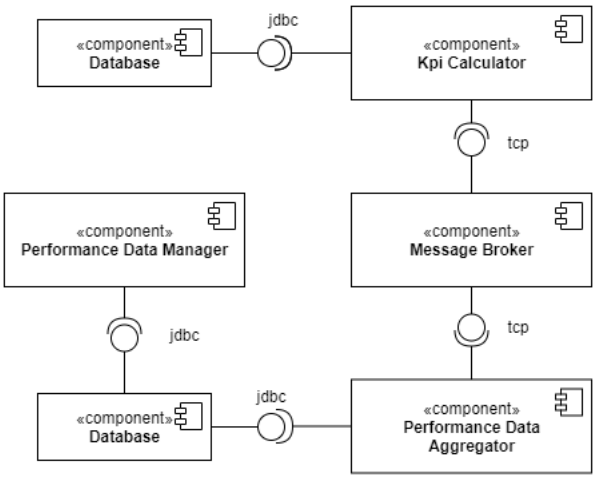


Figure 5.13: Logic view of KPICalculator and PerformanceDataAggregator

As mentioned in step 4, aggregations are configured to execute periodically. Figure 5.14 represents the sequence diagram of the process of aggregate PM data.

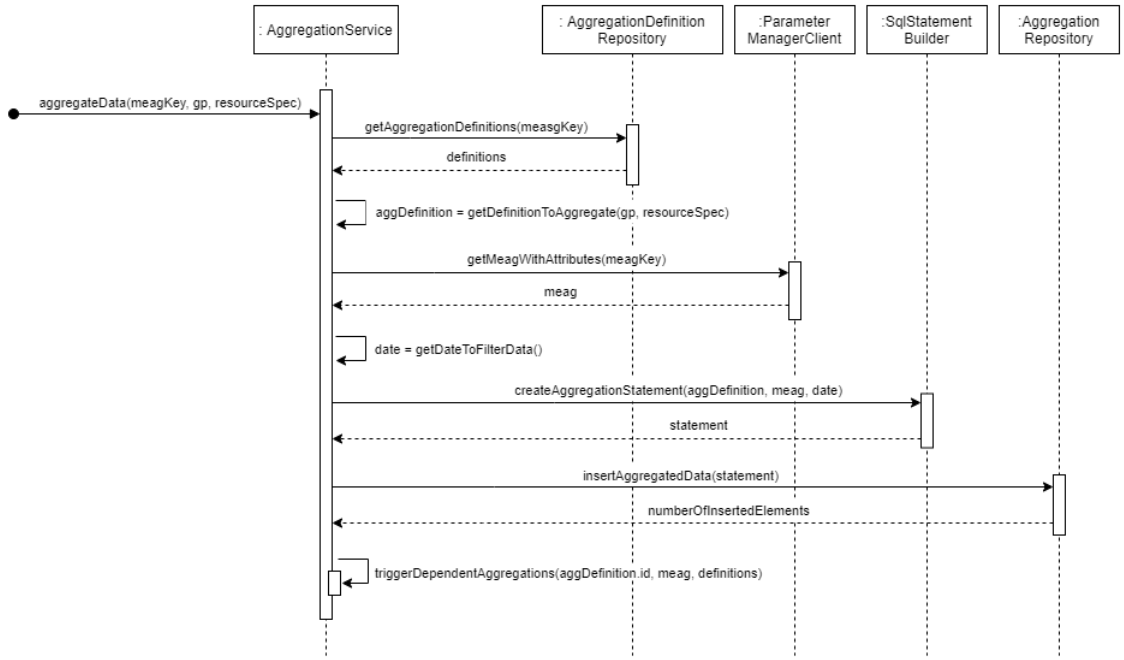


Figure 5.14: Sequence diagram for aggregate values

The following list describes the responsibilities of each method presented in the aggregation process:

- `getAggregationDefinition(meagKey)` - returns the aggregations that are configured for the requested `meagKey`.
- `getMeagWithAttributes(meagKey)` - used to obtain the values of the attributes that belong to the `meagKey`. With this, it is possible to aggregate any data set, as the system can dynamically infer which attributes to use.
- `getDateToFilterData()` - this method returns a timestamp that corresponds to the data to be aggregated.
- `createAggregationStatement(aggDefinition,meag,date)` - his responsibility is to build an insert statement to execute in a database, to aggregate a set of PM data. Listing 5.1 presents an example of an insert statement. This one is to execute a temporal aggregation of the granularity period hour from the data presented in meag_cell_raw into the table meag_cell_hour.

```
insert into "meag_cell_hour"
select
    sum(distinct resource_path) as "cntElements",
    count(*) as "cntRecords",
    trunc_date("hour", beginTime) as "timestamp",
    resource_path as "resource_path",
    sum(attr1) as "attr1",
    sum(attr2) as "attr2"
from "meag_cell_raw"
where beginTime = '21-04-2021'
group by trunc_date("hour", beginTime), resource_path
```

Listing 5.1: Example of an insert statement to aggregate raw PM into cell/hour

- `insertAggregatedData(statement)` - executes the insert statement in the database to create the aggregated data.
- `triggerDependentAggregations(id,meag, definitions)` - his responsibility is to execute other aggregations which are dependent on the aggregation that has just been executed.

UC-7: Request the calculation of KPIs by resource or time

The calculation of the KPIs is made asynchronously between the `KPICalculator` and `PerformanceDataAggregator`. The sequence diagram in figure 5.15 represents the process view of the request to calculate KPIs made by a Vismon user. The vismon user can also add the filter for a time range and a list of resources to the KPIs calculation request. This is not represented in diagram 5.15 to simplify the visualization.

`KPICalculator` has also a simple cache that is composed of KPI's formulas that were used in recent requests. The process view of getting a KPI's formula is presented in figure B.6.

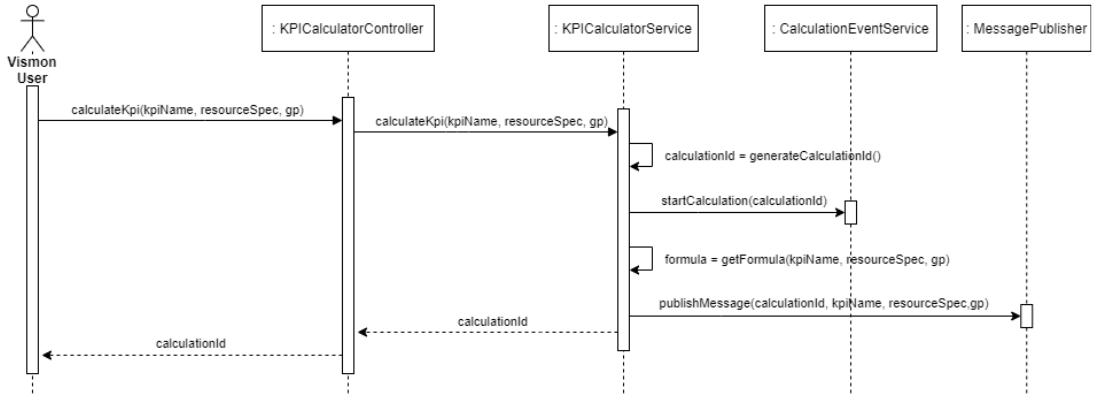


Figure 5.15: Sequence diagram for calculating a KPI

The following list describes the responsibilities of the relevant methods presented in the process presented above:

- `startCalculation(calculationId)` - adds an entry to a table with the calculation events, indicating that an event has started for the indicated calculationId.
- `publishMessage(calculationId,kpiName,resourceSpec,gp)` - publish a message to a topic that is being subscribed by `PerformanceDataAggregator`. The message must contain the KPI name, the resource specification, the granularity period and the resources filter. It also can have information about the time range for the calculation as well as group options.

When a message arrives at the topic that `PerformanceDataAggregator` is subscribing to, it fetches the formula values and publishes the result in a distinct topic to `KPICalculator` finishes the calculation. The diagram that is present in figure 5.16 represents this process.

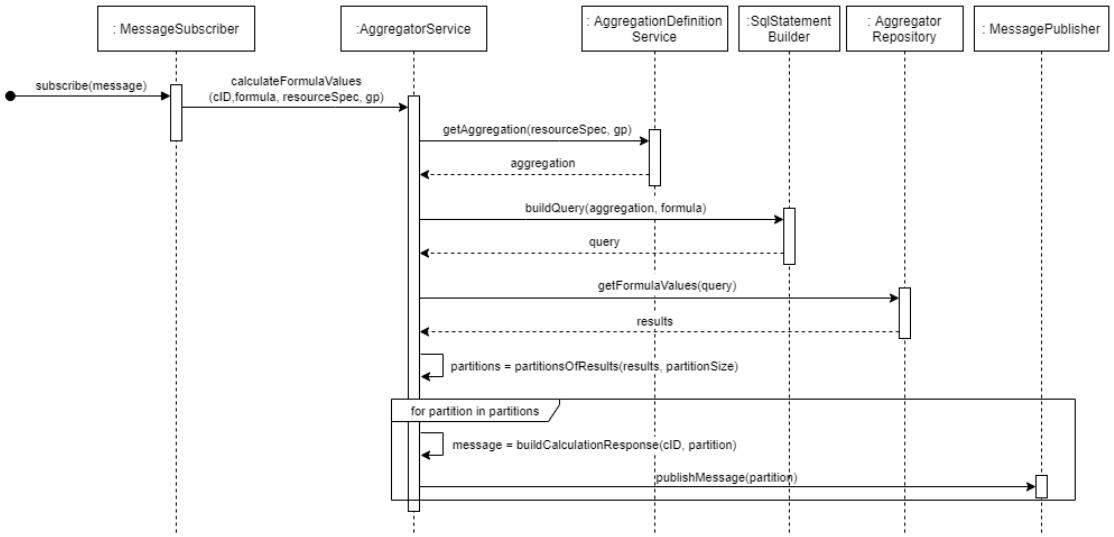


Figure 5.16: Sequence diagram for calculating a KPI

The following list describes the responsibilities of each method presented in figure 5.16 diagrams.

- `getAggregation(resourceSpec,gp)` - retrieves the aggregation definition of the resource specification and granularity period requested.
- `buildQuery(aggregation,formula)` - this method builds a query to be executed in the database with the aggregated values that will return the formula numerator and denominator.
- `getFormulaValues(query)` - executes the built query to retrieve the aggregated data. Listing 5.2 shows an example of a query that is built to get the formula values by the hour.

```
select
    cntElements,
    cntRecords,
    "timestamp",
    resource_path,
    100*(attr1+attr3) as "numerator",
    attr2 as "denominator"
from "meag_cell_hour"
where resource_path = 'cell_xyz'
    and "timestamp" between '2021-04-21 00:00:00' and '2021-04-21 11:00:00'
group by "timestamp", resource_path
```

Listing 5.2: Example of a query to return the formula values

- `publishMessage(partition)` - its responsibility is to publish the results of the executed query into a topic that `KPICalculator` is subscribing to. The results published in the topic contain information about the NE, timestamp, the value of the numerator and denominator and information about the aggregations to calculate temporal integrity.

Finally, figure 5.17 represents the last sequence of executed steps when the formula values arrive at the topic that the `KPICalculator` is subscribing.

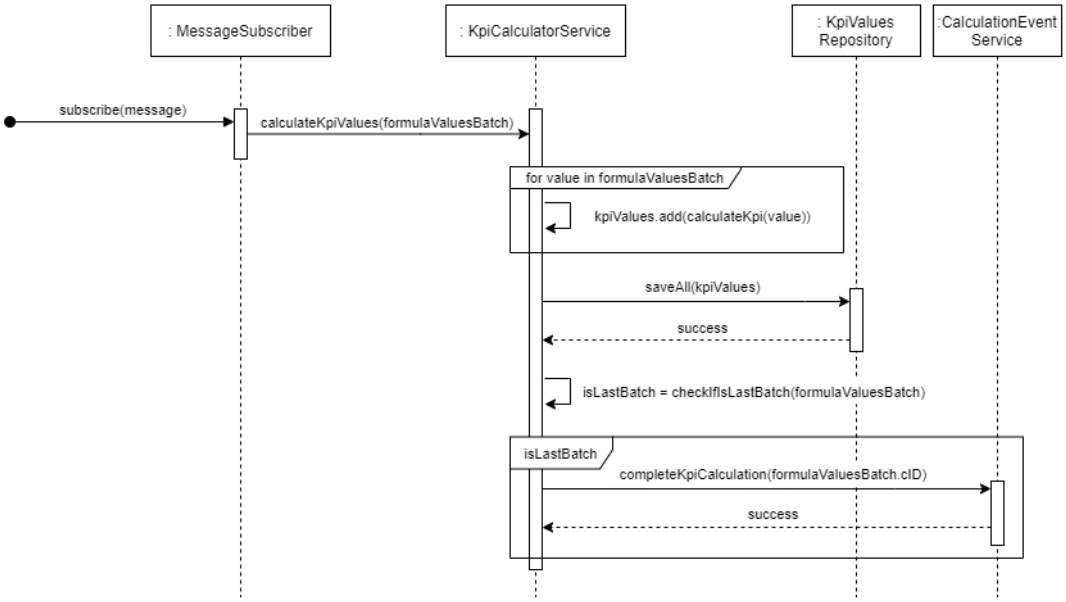


Figure 5.17: Sequence diagram for calculating a KPI

With these values, is performed the calculation of the formula and the temporal integrity. When completed, the values are persisted in the database. Each value is associated with the calculation identification. If the message received indicates that is the last batch, it is inserted an end calculation event in the database.

UC-8: Visualize calculated KPIs

Figure 5.18 represents the process view of visualizing the calculated KPIs by calculation identification.

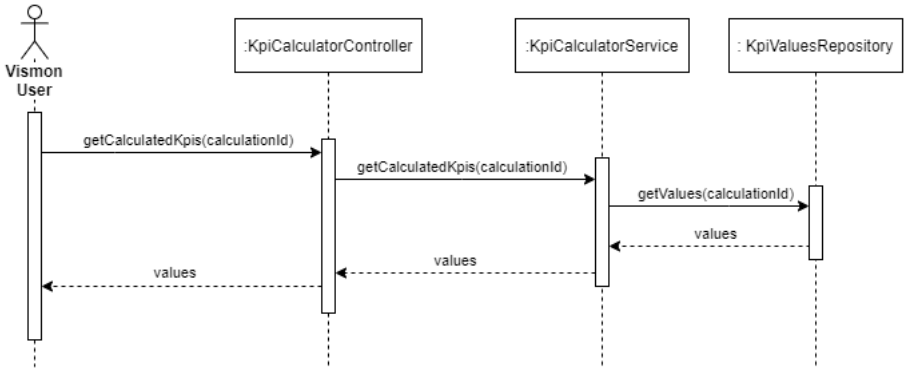


Figure 5.18: Sequence diagram to visualize the calculated KPIs

Database

`PerformanceDataAggregator` component needs to persist the configurations for the pre-defined aggregations and the data pre-aggregated. Figure 5.19 represents the data model for this service in its private schema. Table *metaEntity_meag_gp_resourceSpec* represents a base table for the executed aggregations.

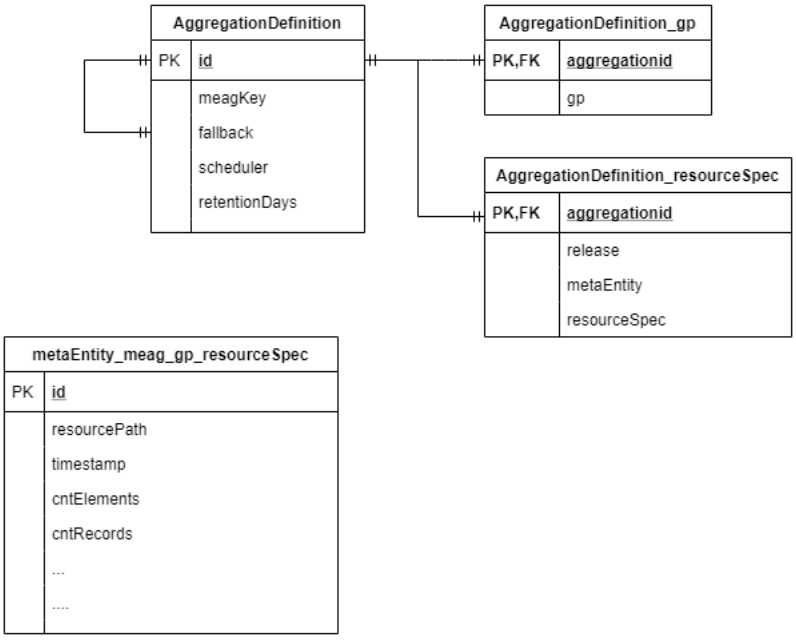


Figure 5.19: Data model for PerformanceDataAggregator

`KPICalculator` has its private database to store KPI calculations and its data model is represented in figure 5.20.

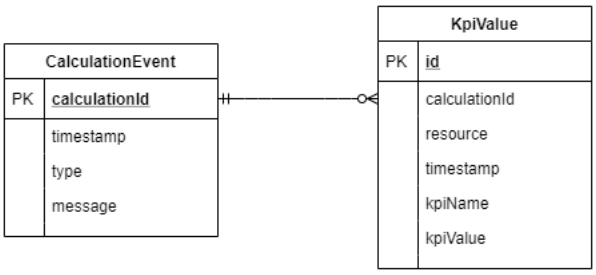


Figure 5.20: Data model for KPICalculator

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.4, presented in Annex B represents a Kanban board that sums the design process. The remaining drivers were completely addressed in this iteration.

Chapter 6

Solution

The purpose of this chapter is to present the construction of the solution for the respective problem. The details of the PoC are explained throughout this chapter, including some aspects of the PoC implementation and adopted practices.

6.1 Development Approach

The Agile Methodologies "are a group of software development methods that are based on iterative and incremental development" (Kumar and Bhatia 2012). In software development, teams start with simple requirements and then continue to increment the details of the solution. Figure 6.1 illustrates this methodology.

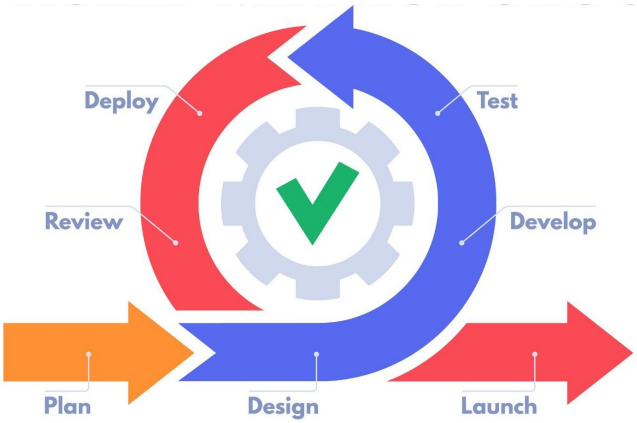


Figure 6.1: Agile methodology. Source (Aditi 2021)

The development of the PoC was based on the agile methodology, more specifically in Scrum. There were two iterations in the development of the project. The first iteration included the development of the KPIs and formulas inventory and the second one, the KPI calculation.

6.2 Implementation

After the design of the system architecture, the components were implemented to create the PoC of the framework. The designation and responsibilities of the components are presented in section 5.3. Thereby, three microservices were created:

- **Inventory Microservice** - works as an inventory of KPIs and formulas.
- **Aggregations Microservice** - performs aggregations with PM data.
- **Calculator Microservice** - calculates KPIs when a user of Vismon requests.

The remaining components that are part of this system were already developed by Celfinet.

Software patterns and principles were used to enhance the robustness of the solution. A design pattern signifies "a general reusable solution to a commonly occurring problem in software design" (Jiang and Mu 2011). It can be a description or template that guides the problem solving and can be utilized in several situations.

The following sections (6.2.1 to 6.2.3) discuss the three microservices, introducing some patterns and techniques used through the development. Section 6.2.4 provides complementary details about what was also relevant for the solution.

6.2.1 Inventory Microservice

The `KPIFormulaInventory` microservice represents the KPI-Formula Inventory Service which belongs to the KPI creation subdomain (see figure 4.3). This microservice is responsible for managing KPIs and their formulas and categories. So, to calculate a KPI, it needs to be present in the inventory with an enable formula.

Figure 6.2 represents the implementation view of this component. The following list sums the responsibilities of each package:

- Web package: this package provides an abstraction of the system to external clients. This access to methods presented in the Service layer.
- Service package: this package contains the business logic necessary to perform the use cases related to the management of KPIs, formulas and tags.
- Domain package: represents all the entities of the business domain as domain objects. Figure 6.3 represents the class diagram that describes the system's classes and the relationships among objects.
- Repository package: this package encapsulates the set of objects persisted in the database.
- Config, exception and infrastructure: these packages support the application with classes to configure the component, exceptions and the necessary clients to perform HTTP requests.

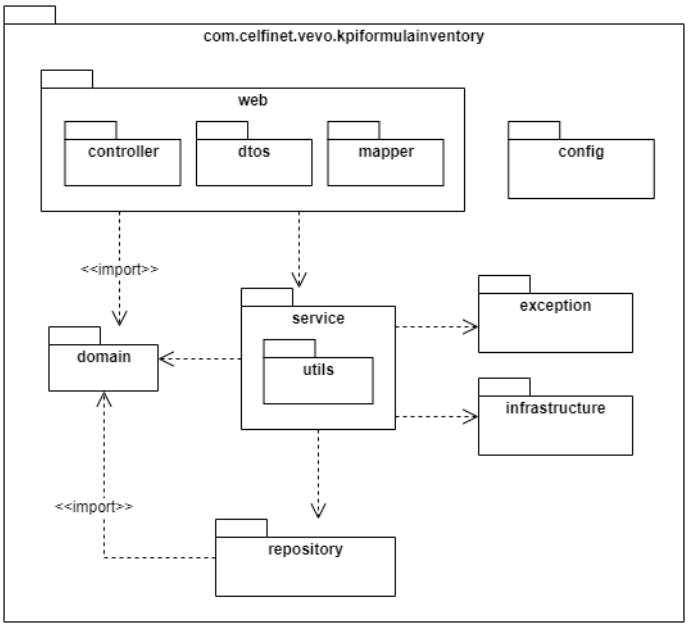


Figure 6.2: Package diagram of KPIFormulaInventory

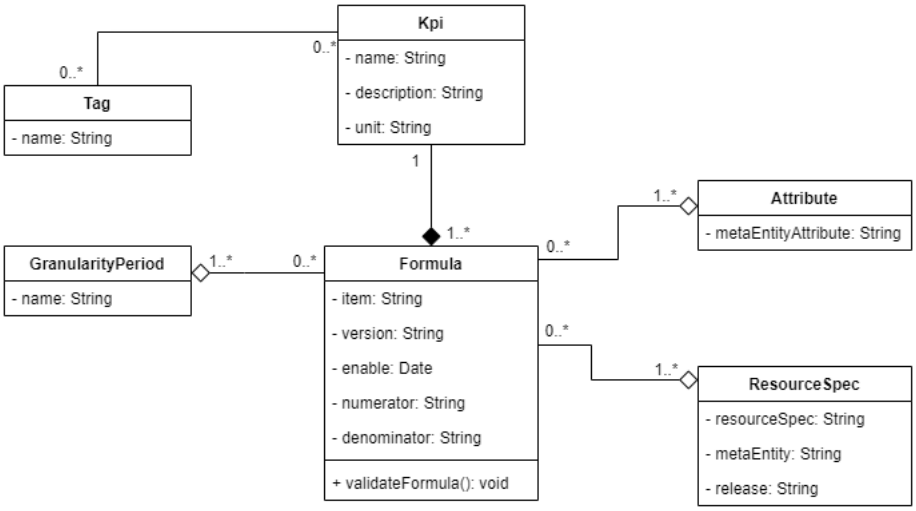


Figure 6.3: Class diagram of KPIFormulaInventory

To control the creation of the objects, the Builder design pattern was applied to separate "the construction of a complex object from its representation so that the same construction process can create different representations" (SourceMaking 2021a). Aggregations and Calculator microservice also used this pattern.

The *@Builder* annotation introduced by Project Lombok[1] is a useful mechanism to implement this pattern without writing any code. This annotation produces the builder APIs for the classes. Listing 6.1 presents an example of its usage.

[1] https://projectlombok.org/

```
1 import lombok.Builder;
2
3 @Builder
4 public class Kpi{
5     private int id;
6     private String name;
7     private String unit;
8 }
```

Listing 6.1: Example of the usage of @Builder annotation

As shown in figure 6.3, the Formula class has a `validateFormula()` method to validate the numerator and denominator. This method must validate the syntax of the mathematical expression:

$$formula = \frac{numerator}{denominator}$$

When the expression is incorrect, the system throws an exception of the InvalidFormulaExpressionException type.

To carry out the validation, research has been performed to find libraries that can validate the syntax of a mathematical expression. The library also needs to support the defined pattern of the formula's variable:

<metaEntity>_<metaEntityAttributeGroup>.<metaEntityAttribute>

The libraries selected to verify if they fulfil the requirements were Exp4j[2], Java Expression Language (JEXL)[3] and MVFLEX Expression Language (MVEL)[4]. A set of simple tests was defined and used to evaluate their adequacy:

1. **Valid expressions:** the results of the tests are present in table 6.1. The expected result is **Valid**.
   - $x + y$ - a simple mathematical expression.
   - $100 + (x + y)/(x/y)$ - a mathematical expression with several operators and parenthesis.
   - $100 + log(x) * (x/y)$ - a mathematical expression with logarithm.
   - $100 + 2^x/(x/y)$ - a mathematical expression with a potency.
   - $cell\_pm\_lte\_eri.x + cell.pm\_lte\_eri.y$ - a mathematical expression with the syntax of the variables in the PoC.

[2] https://www.objecthunter.net/exp4j/
[3] https://commons.apache.org/proper/commons-jexl/
[4] http://mvel.documentnode.com/#language-guide-for-2.0

Table 6.1: Tests for valid mathematical expressions

Valid expression	Exp4j	JEXL	MVEL
$x + y$	Valid	Valid	Valid
$100 + (x + y) * (\frac{x}{y})$	Valid	Valid	Valid
$100 + log(x) * (x/y)$	Valid	Valid	Invalid
$100 + 2^x/(x/y)$	Valid	Valid	Invalid
$cell.pm\_lte\_eri.x + cell.pm\_lte\_eri.y$	Valid	Valid	Invalid

2. **Invalid expressions:** the results of the tests are present in table 6.2. The expected result is **Invalid**.
   - $100xy$ - a mathematical expression with missing signals.
   - $100 + (x + y$ - a mathematical expression with a missing parenthesis.
   - $100 + (x/)$ - a mathematical expression with a missing number or variable.

Table 6.2: Tests for invalid mathematical expressions

Invalid expression	Exp4j	JEXL	MVEL
$100xy$	Valid	Invalid	Invalid
$100 + (x + y$	Invalid	Invalid	Invalid
$100 + (x/)$	Invalid	Invalid	Invalid

To conclude, the JEXL library was the one used in the solution to validate the KPI's formula insofar as the only one that has passed all the defined tests.

Data Transfer Object (DTO) pattern stands for "an object that carries data between processes to reduce the number of method calls" (Fowler 2021a). DTOs are used in microservices to transfer data between methods and between microservices. Figure 6.4 presents an example of this pattern implementation when transferring data between this service and external clients.

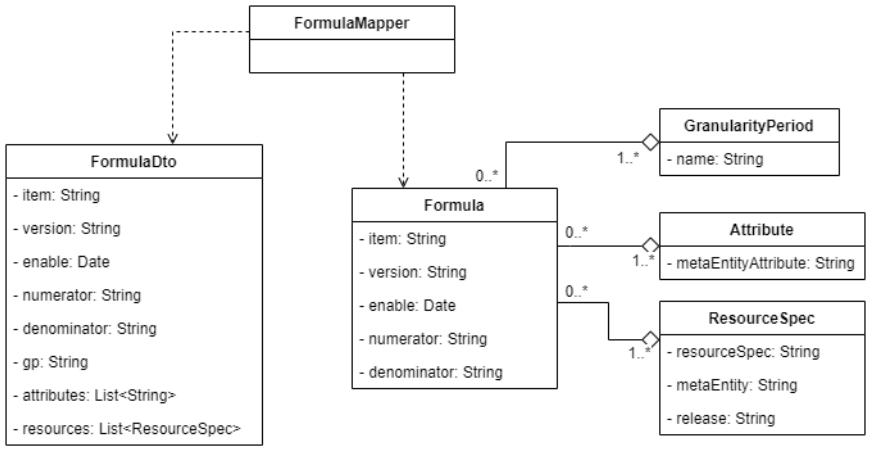


Figure 6.4: Usage of DTO in KPIFormulaInventory

The Repository pattern was used to save the domain objects. A Repository "encapsulates the set of objects persisted in a data store and the operations performed over them, providing a more object-oriented view of the persistence layer" (Fowler 2021b). This pattern helps to keep domain objects ignorant of their persistence and achieve loose coupling.

To configure the persistence layer, Spring Data provides an interface with the most relevant methods in a standard way that simplifies and remove the implementation of this layer. With this, a Data Access Object (DAO) interface must implement the JpaRepository interface to have access to all provided methods (Baeldung 2021). Listing 6.2 represents an example of the usage of this interface.

```
1 public interface FormulaRepository  extends
      JpaRepository<Formula, Long> {
2
3   Optional<Formula> findByItemAndVersionIgnoreCase(String item,
      String version);
4
5   List<Formula> findByKpiNameIgnoreCase(String kpiName);
6
7   List<Formula> findByEnable(Boolean enabled);
8
9 }
```

Listing 6.2: Example of the JpaRepository interface

6.2.2 Aggregations Microservice

The `PerformanceDataAggregator` microservice represents the Performance Aggregator Service which belongs to the Performance data subdomain (see figure 4.3). This microservice is responsible for executing pre-defined aggregations of PM data. Thereby, it is also responsible for providing data to calculate the KPIs.

Figure 6.5 represents the implementation view of this component. The following list sums the responsibilities of each package:

- Web package: this package provides an abstraction of the system to external clients. This access to methods presented in the Service layer.
- Service package: this package contains the business logic necessary to perform the aggregations with PM data and provide data to calculate the KPIs.
- Domain package: represents all the entities of the business domain as domain objects.
- Repository package: this package encapsulates the set of objects persisted in the database.
- Messaging package: this package contains all the support classes to provide the subscription and production of messages to the message broker.
- Config and exception: these packages support the application with classes to configure the component, exceptions and the necessary clients to perform HTTP requests.

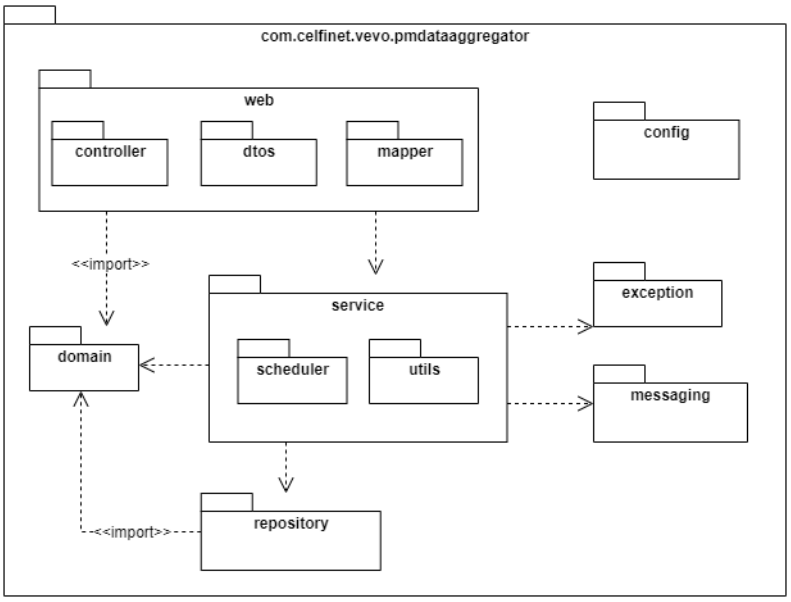


Figure 6.5: Package diagram of PerformanceDataAggregator

For the system to execute aggregations of PMs data they must be configured. Each aggregation definition has a nodal and temporal aggregation, the fall-back for the aggregation, the scheduler for the execution and the retention period. The fall-back attribute defines which aggregation gave rise to the current one. Having as an example in figure 6.6, the fall-back for the Node-Day aggregation is the Cell-day.

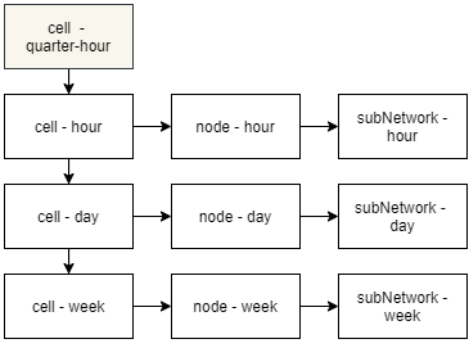


Figure 6.6: Example of aggregation fall-back of PM data

The scheduler is configured as a Cron Expression. A Cron Expression "is a string consisting of six or seven subexpressions that describe individual details of the schedule" (Oracle 2009). There multiple combinations to configure this expression however in this case it is very simple as "0 0 12 * * ?" (starts every day at 12h) or "0 0 * * * ?" (starts every day at every hour).

Cron Expression is used to configure trigger instances of the Quartz Job Scheduling Library[5]. This library was chosen due to its familiarity and simplicity for configuring jobs and triggers. With this library, it is possible to configure simple schedules for any job. Listing 6.3 presents the logic implemented in this component to configure the aggregations to be executed periodically.

[5]http://www.quartz-scheduler.org/

```
public void init() throws SchedulerException {
    List<AggregationDefinition> aggregations =
    aggregationDefinitionRepository
            .findBySchedulerNotNull();

    for (AggregationDefinition definition : aggregations) {
        JobKey jobKey = new JobKey("job_" + definition.getId(),
    "aggregations");

        Map<String, Object> map = new HashMap<String, Object>();
        map.put("aggregation", definition);

        JobDetail job = JobBuilder
                .newJob(AggregatorJob.class)
                .withIdentity(jobKey)
                .setJobData(new JobDataMap(map)).build();

        Trigger trigger = TriggerBuilder.newTrigger()
                .withIdentity("trigger_" + definition.getId(),
    "aggregations")
                .withSchedule(CronScheduleBuilder
                    .cronSchedule(definition.getScheduler()))
                .build();

        scheduler.scheduleJob(job, trigger);
    }
    scheduler.start();
}
```

Listing 6.3: Configuration of the aggregation's triggers

The pre-defined aggregations are constructed by executing an insert statement in the database. This statement is generated considering the attributes of a group and the type of aggregation. The same method is adopted when a request is received to calculate KPIs, in other words, a query is built to be executed in the database and retrieve the data.

To build the insert statements and queries the jOOQ[6] library was elect to use in this component. This library embeds the SQL language to help to construct type-safe statements.

```
public static String buildQuery() {
    return select(field("attr1"))
        .from(table("table_x"))
        .where(and(field("timestamp")
            .between("2021-05-02","2021-05-03")))
        .groupBy(field("resourcePath"))
        .orderBy(field("resourcePath").asc()).getSQL();
}
```

Listing 6.4: Example of the usage of jOOQ

[6]https://www.jooq.org

6.2.3 Calculator Microservice

The `KPICalculator` microservice represents the KPI Calculator Service which belongs to the KPI Calculation subdomain (see figure 4.3). This microservice is responsible for calculating KPIs.

Figure 6.7 represents the implementation view of this component. The following list sums the responsibilities of each package:

- Web package: this package provides an abstraction of the system to external clients. This access to methods presented in the Service layer.
- Service package: this package contains the business logic necessary to perform KPIs calculation.
- Domain package: represents all the entities of the business domain as domain objects.
- Repository package: this package encapsulates the set of objects persisted in the database.
- Messaging package: this package contains all the support classes to provide the subscription and production of messages to the message broker.
- Config, exception and infrastructure: these packages support the application with classes to configure the component, exceptions and the necessary clients to perform HTTP requests.

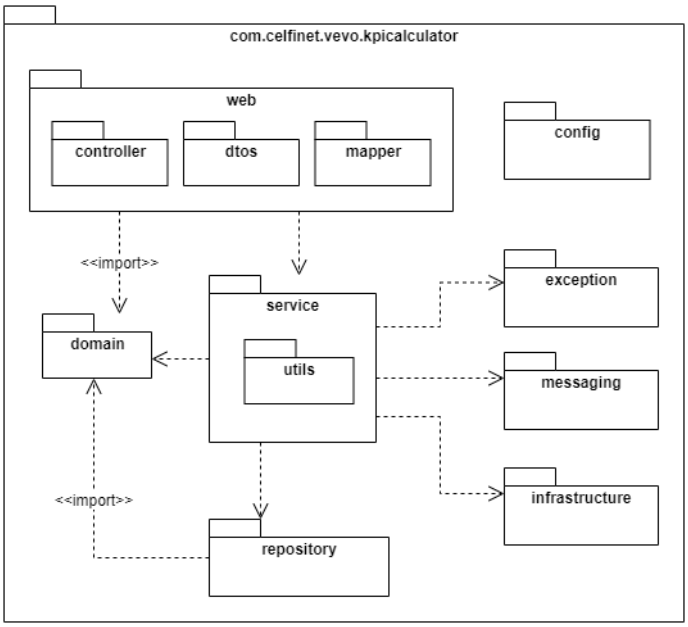


Figure 6.7: Package diagram of KPICalculator

The Dependency Injection technique was applied in this microservice and in the Inventory and Calculator microservice to break the dependencies between the classes. Thereby, they become dependent on interfaces and no longer on themselves. Dependency injection is a design pattern that is used to implement the Inversion of Control principle. Figure 6.8 presents an example of the implementation of this pattern in the Calculator microservice.

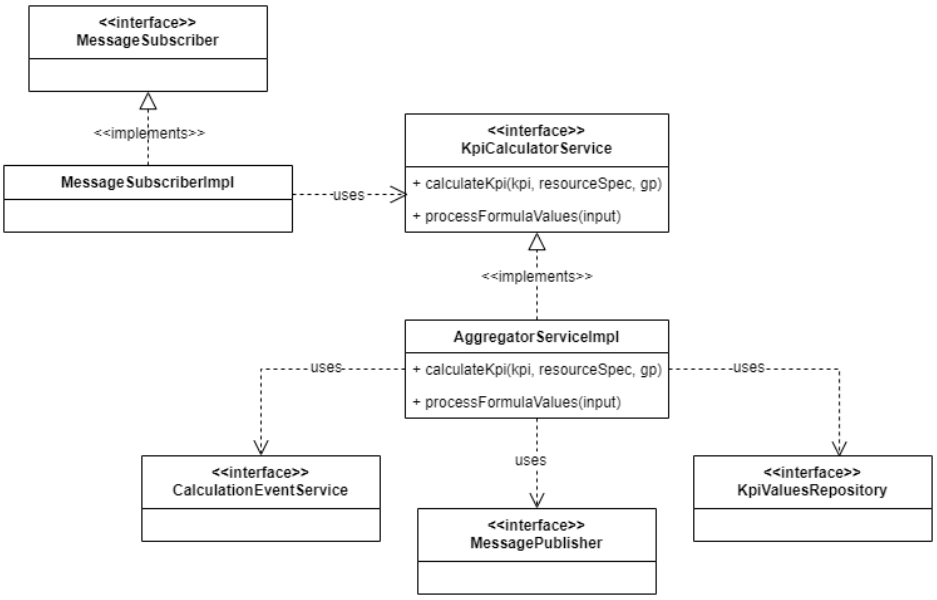


Figure 6.8: Example of the usage of dependency injection in KPICalculator component

Facade design pattern was implemented in the three developed microservices to provide a layer of abstraction, wrapping the complexity of the modules in a unified interface. The Facade pattern "defines a higher-level interface that makes the subsystem easier to use" (SourceMaking 2021b). The implementation of an API encapsulates the system and makes it accessible by other systems or clients.

To have good documentation of the developed APIs of the created microservices was used the Swagger tool. Swagger "is a set of open-source tools built around the OpenAPI Specification" (Swagger 2021) that helps developers design, build, document and consume REST APIs. In addition, it allows easy access to the endpoints implemented by the system.

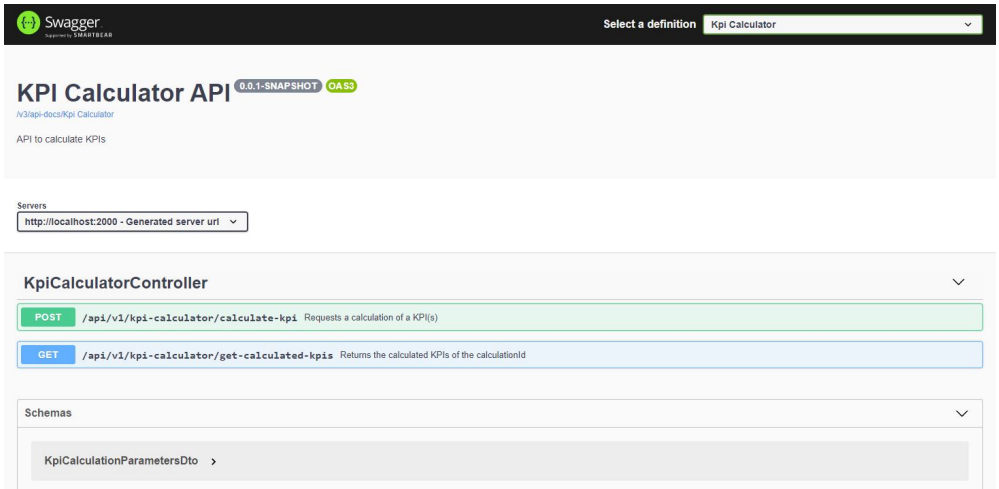


Figure 6.9: Swagger UI of the KPICalculator component

6.2.4 Particular Situations

This section describes the Apache Kafka technology and the management of its topics. In addition, it also addresses the use of continuous practices, such as Continuous Integration (CI) and Continuous Delivery (CDE).

Apache Kafka

Apache Kafka is an open-source event streaming platform to implement data pipelines. Event streaming "is the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events" (Apache Foundation 2017). Apache Kafka allows to publish and subscribe to a stream of events. In addition, it stores and processes streams of events.

To implement the Kafka producer and subscriber in the microservices was followed a tutorial[7]. For this, was used the spring-kafka dependency in the projects.

To manage data inside the Apache Kafka is used AKHQ[8]. This platform is an open-source Kafka GUI to administer topics, data, consumer groups, among others. Figure 6.10 presents two topics created in the message broker to support the calculation of KPIs: calculateKPIs and calculateKPIsResponse. This UI helps to see the messages flow through the topics.

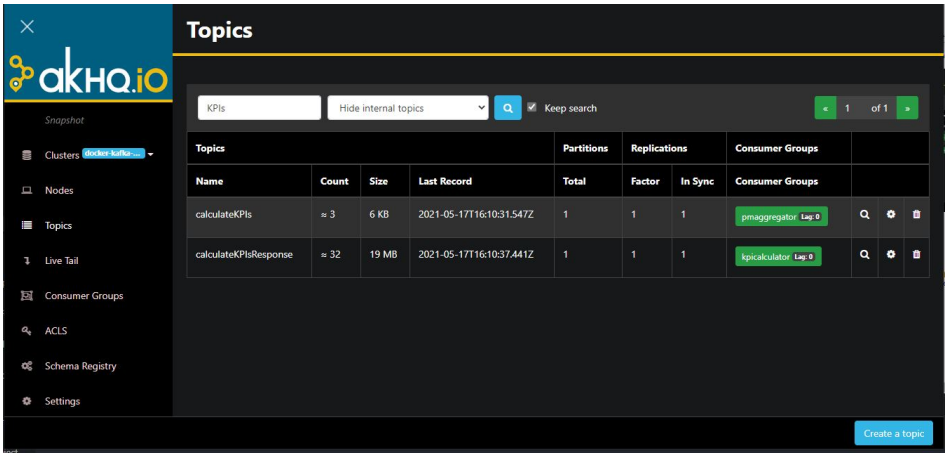


Figure 6.10: Manage topics view

This platform also provides a detailed view of the data that is present in a determined topic. Figure 6.11 gives an example of the data that is present in the calculateKPIs topic.

[7] https://www.baeldung.com/spring-kafka
[8] https://akhq.io/

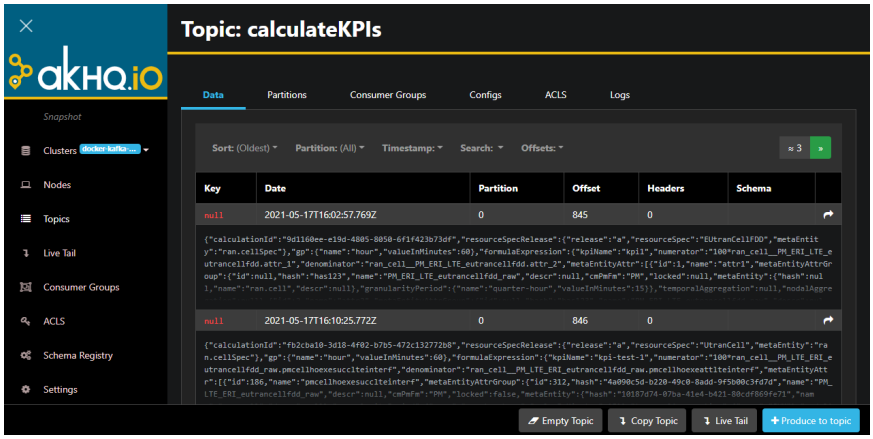


Figure 6.11: Manage calculateKPIs view

Continuous Integration and Delivery

To help companies accelerate the development and delivery of the software, have been applied some continuous practices to greenfield and maintenance projects. Continuous software engineering refers to "develop, deploy and get quick feedback from software and customer in a very rapid cycle" (Shahin, Ali Babar, and Zhu 2017).

CI is a widely "established development practice in software development industry, in which members of a team integrate and merge development work" (Shahin, Ali Babar, and Zhu 2017). CDE ensures that an application "is always at production-ready state after successfully passing automated tests and quality checks" (Shahin, Ali Babar, and Zhu 2017).

These practices are important to the project because it improves software delivery process. Although this project is a PoC, it is relevant to have these practices since the beginning.

Azure DevOps[9] is a Microsoft Software as a Service (SaaS) platform that provides a set of tools, standing out for the project, Azure Repos and Azure Pipelines. Azure Repos provides Git repositories for source control. Figure 6.12 represents the view of the projects contained in the PoC.

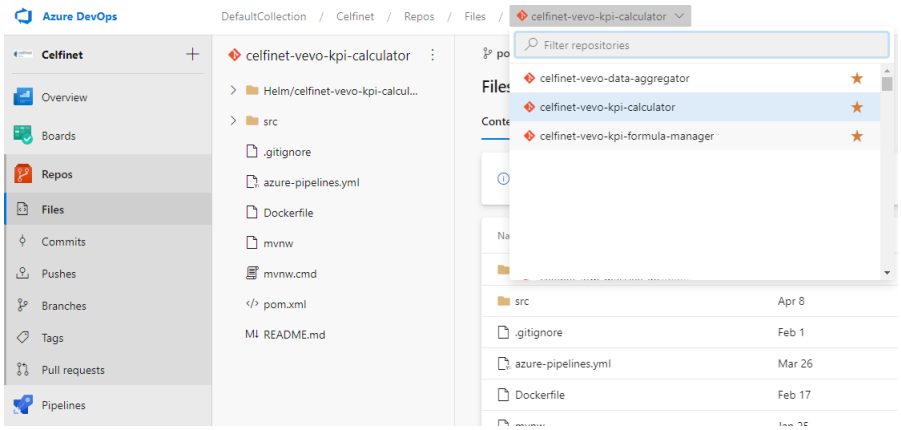


Figure 6.12: View of the repositories in Azure Repos

[9]https://azure.microsoft.com/en-us/services/devops/

Azure Pipelines provides services to support CI/CDE (Microsoft 2021a). Figure 6.13 represents a fragment of the pipeline.

```yaml
steps:
  Settings
- task: DownloadSecureFile@1
  name: FeedAuth
  displayName: 'Configure Azure DevOps Server Maven Feed Authentication'
  inputs:
    secureFile: 'settings.xml'
  Settings
- task: Maven@3
  name: CompileJava
  displayName: 'Maven Build'
  inputs:
    mavenPomFile: 'pom.xml'
    goals: 'verify'
    publishJUnitResults: true
    options: '-s $(FeedAuth.secureFilePath)'
    testResultsFiles: |
      **/surefire-reports/TEST-*.xml
      **/failsafe-reports/TEST-*.xml
    codeCoverageToolOption: 'jaCoCo'
    sonarQubeRunAnalysis: false
  Settings
- task: Docker@2
  condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/poc'))
  displayName: 'Build & Publish Docker images to Celfinet Harbor'
  inputs:
    containerRegistry: 'Harbor'
    repository: 'vismon/vevo/celfinet-vevo-kpi-calculator'
    command: 'buildAndPush'
    Dockerfile: '**/Dockerfile'
    tags: $(MajorVersion).$(MinorVersion).$(Build.BuildId)
```

Figure 6.13: Build pipeline of the KPICalculator service

6.3 Code Validation and Analysis

This section presents some details about the validation and analysis of the code developed for the PoC.

6.3.1 Unit, Integration and Acceptance Tests

Testing is a crucial step and described as "the process of making sure that a software system meets its requirements, that brings the necessary functionality [...] to its user community" (Bass, P. C. Clements, and Kazman 2013).

Tests have classifications according to their scopes, such as unit, integration, system, acceptance tests, among others. Within the context of the PoC were developed unit, integration and acceptance tests.

A unit test "exercises a unit of code in isolation and compares actual with expected results" (Olan 2003). It tests the software "in a standalone fashion" and sometimes needs to use stubs to replace other units with which the test interacts (Bass, P. C. Clements, and Kazman 2013).

The unit tests were written after the creation of functionality and applied to the components of the solution. Listing 6.5 presents a unit test that checks if was thrown an InvalidFormulaExpressionException when the mathematical expression has incorrect syntax.

```
@Test
public void formulaIsInvalid() {
  Formula formula = Formula.builder()
      .numerator("cell__pm_lte_eri.x+cell__pm_lte_eri.y-(")
      .denominator("cell_pm_lte_eri.x").build();

  List<MetaEntityAttrRefDTO> attrs = List.of(
      new MetaEntityAttrRefDTO("x", "pm_lte_eri","cell"),
      new MetaEntityAttrRefDTO("y", "pm_lte_eri","cell"));

  assertThrows(InvalidFormulaExpressionException.class,
      () -> formula.validateFormula(attrs));
}
```

Listing 6.5: Example of a unit test

Integration tests concentrate "on finding problems related to the interfaces between elements in a design" (Bass, P. C. Clements, and Kazman 2013) and happens when independent units of software work together.

To assure that the elements of a component work properly together, the inclusion of the integration tests adopted. In the context of the project, integration tests only cover relations between the elements of a component.

```
@Test
public void testCalculateKpi_integration() throws Exception {

  KpiCalculationParameters request = getKpiCalculationRequest();

  UUID calculationId =
    kpiCalculatorService.calculateKpis(request);
  assertNotNull(calculationId);

  CalculateKPIsResponse calculationResponse =
    getCalculationResponse(calculationId);
  String message =
    objectMapper.writeValueAsString(calculationResponse);
  messageSubscriber.subscibeKpiCalculation(message);

  KpiCalculationResult kpiValues =
    kpiCalculatorService.getKpiValues(calculationId.toString());

  assertNotNull(kpiValues);
  assertEquals(2, kpiValues.getPayload().size());
}
```

Listing 6.6: Example of a integration test

The acceptance test "verifies that the application delivers the business value expected by the customer" (Humble and Farley 2011).

In the context of the project, acceptance tests cover relations between the components of the system. Consider that acceptance tests are carried out based on the functional requirements they were executed at the end of every iteration. Table 6.3 presents the acceptance tests.

Table 6.3: Acceptance tests

Functional Requirement	Result
Manages a KPI	Fulfil
Manages a KPIs category	Fulfil
Categorizes KPIs	Fulfil
Visualizes KPIs	Fulfil
Manages a formula for a KPI	Fulfil
Visualize active formulas for a KPI	Fulfil
Request the calculation of KPIs by resource or time	Fulfil
Visualize calculated KPIs	Fulfil

6.3.2 Code Coverage

Code coverage "is a measure of the degree to which a test suite exercises a software system" (Ivankovi et al. 2019). It was measured for the three microservices. Celfinet defined a minimum value for the code coverage of its projects at 80%.

The obtained results include unit and integration tests present in the components. Figures 6.14, 6.15 and 6.16 presents the code coverage measured in the `KPIFormulaInventory`, `PerformanceDataAggregator` and `KPICalculator` services respectively.

celfinet-vevo-kpi-formula-manager

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.celfinet.vevo.kpiformulamanager.service		81%		57%	24	77	31	168	8	51	0	7
com.celfinet.vevo.kpiformulamanager.domain		68%		38%	27	40	30	73	3	14	0	4
com.celfinet.vevo.kpiformulamanager.web.mapper		96%		50%	4	16	4	54	0	12	0	5
com.celfinet.vevo.kpiformulamanager		37%		n/a	1	2	2	3	1	2	0	1
com.celfinet.vevo.kpiformulamanager.service.aggregation.utils		91%		n/a	1	3	1	3	1	3	0	1
com.celfinet.vevo.kpiformulamanager.service.utils		99%		87%	1	10	0	53	0	6	0	3
com.celfinet.vevo.kpiformulamanager.web.controller		100%		n/a	0	17	0	19	0	17	0	6
com.celfinet.vevo.kpiformulamanager.exception		100%		n/a	0	10	0	18	0	10	0	5
com.celfinet.vevo.kpiformulamanager.infrastruture.parameter.model		100%		n/a	0	2	0	5	0	2	0	1
com.celfinet.vevo.kpiformulamanager.config		100%		n/a	0	2	0	5	0	2	0	1
Total	298 of 2,019	85%	59 of 120	50%	58	179	68	401	13	119	0	34

Figure 6.14: Code coverage of KPIFormulaInventory service

celfinet-vevo-data-aggregator

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.celfinet.vevo.dataaggregator.service		90%		77%	41	137	31	383	2	42	0	4
com.celfinet.vevo.dataaggregator.service.job		4%		0%	6	7	18	19	4	5	0	1
com.celfinet.vevo.dataaggregator.domain		50%		32%	14	18	16	36	1	4	0	2
com.celfinet.vevo.dataaggregator.config		79%		n/a	2	16	14	57	2	16	0	4
com.celfinet.vevo.dataaggregator.service.utils		94%		83%	16	55	7	70	5	19	0	4
com.celfinet.vevo.dataaggregator.web.controller		0%		n/a	3	3	6	6	3	3	2	2
com.celfinet.vevo.dataaggregator.messaging		82%		50%	2	6	6	21	1	5	0	2
com.celfinet.vevo.dataaggregator.service.scheduler		61%		n/a	1	5	3	9	1	5	0	2
com.celfinet.vevo.dataaggregator.web.mapper		92%		50%	3	7	3	24	0	4	0	1
com.celfinet.vevo.dataaggregator		37%		n/a	1	2	2	3	1	2	0	1
com.celfinet.vevo.dataaggregator.model.exception		82%		n/a	1	2	2	4	1	2	0	1
com.celfinet.vevo.dataaggregator.infrastruture.parameter.model		97%		n/a	1	5	1	11	1	5	0	2
com.celfinet.vevo.dataaggregator.domain.enums		100%		n/a	0	3	0	8	0	3	0	2
com.celfinet.vevo.dataaggregator.domain.sql		100%		100%	0	5	0	16	0	2	0	1
com.celfinet.vevo.dataaggregator.repository		100%		n/a	0	2	0	9	0	2	0	1
Total	486 of 3,482	86%	82 of 306	73%	91	273	109	676	22	119	2	30

Figure 6.15: Code coverage of PerformanceDataAggregator service

celfinet-vevo-kpi-calculator

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.celfinet.vevo.kpicalculator.service		81%		51%	42	90	34	234	5	38	0	3
com.celfinet.vevo.kpicalculator.repository		56%		0%	3	8	13	33	1	6	0	2
com.celfinet.vevo.kpicalculator.service.utils		86%		75%	7	24	7	66	2	8	0	3
com.celfinet.vevo.kpicalculator.web.mapper		11%		0%	3	5	16	18	2	4	1	3
com.celfinet.vevo.kpicalculator.domain		0%		0%	2	2	3	3	1	1	1	1
com.celfinet.vevo.kpicalculator.service.cache		85%		50%	3	16	5	31	2	15	0	2
com.celfinet.vevo.kpicalculator.messaging		89%		n/a	0	4	4	18	0	4	0	2
com.celfinet.vevo.kpicalculator		37%		n/a	1	2	2	3	1	2	0	1
com.celfinet.vevo.kpicalculator.util		0%		n/a	1	1	1	1	1	1	1	1
com.celfinet.vevo.kpicalculator.exception		100%		n/a	0	23	0	79	0	23	0	8
com.celfinet.vevo.kpicalculator.config		100%		n/a	0	14	0	30	0	14	0	5
com.celfinet.vevo.kpicalculator.domain.enums		100%		n/a	0	2	0	5	0	2	0	2
com.celfinet.vevo.kpicalculator.web.controller		100%		n/a	0	2	0	4	0	2	0	1
Total	430 of 2,427	82%	65 of 138	52%	62	193	85	525	15	120	3	34

Figure 6.16: Code coverage of KPICalculator service

6.3.3 Code Analysis

This section presents the analysis tools used in this project. They helped in the validation of the code and its structure, detecting defects that may exist. For that, were used Checkstyle and SpotBugs.

Checkstyle[10] helps writing code for Java that "adheres to a coding standard" (Checkstyle 2021) by automating the code analysis process. It is possible to configure any coding standard in the Checkstyle tool. Figure 6.17 presents a report of the static analysis performed on the `KPICalculator` component.

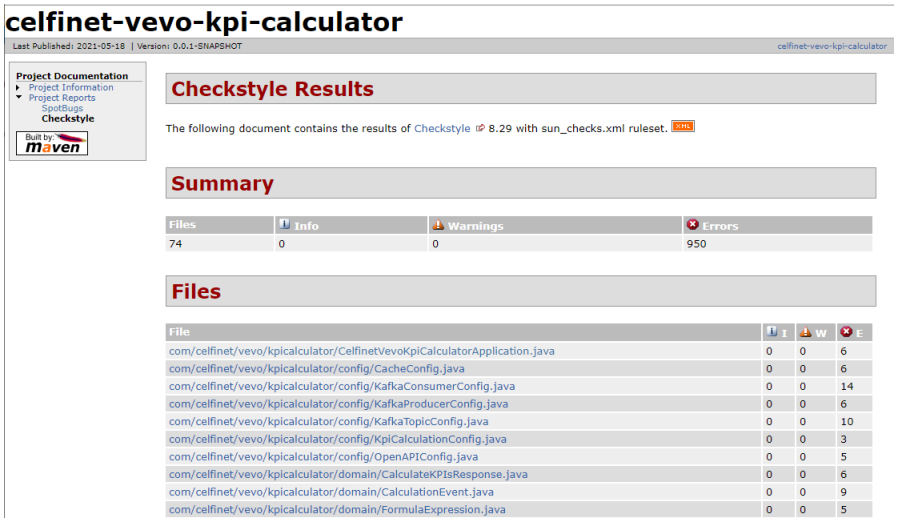
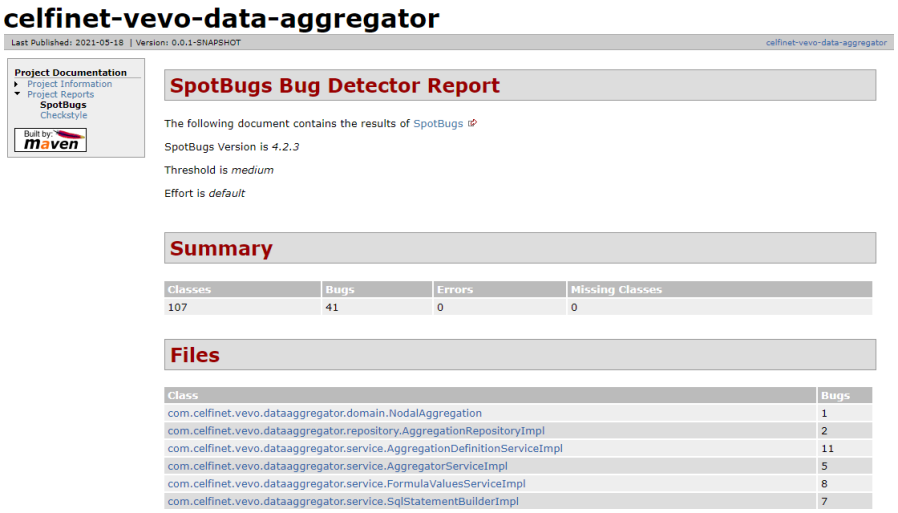


Figure 6.17: Checkstyle report from KPICalculator component

SpotBugs[11] is an application that "uses static analysis to look for bugs in Java code" (SpotBugs 2021). Figure 6.18 presents a report of the static analysis performed on the `Performance DataAggregator` component.

To generate the code analysis reports were added the plugins to the reporting section of the *pom.xml* file. Therefore, running the *mvn site* generates the analysis report.

[10] https://checkstyle.sourceforge.io/
[11] https://spotbugs.github.io/



celfinet-vevo-data-aggregator

Last Published: 2021-05-18 | Version: 0.0.1-SNAPSHOT celfinet-vevo-data-aggregator

Project Documentation
- Project Information
- Project Reports
  - SpotBugs
  - Checkstyle

Built by: maven

SpotBugs Bug Detector Report

The following document contains the results of SpotBugs

SpotBugs Version is *4.2.3*

Threshold is *medium*

Effort is *default*

Summary

Classes	Bugs	Errors	Missing Classes
107	41	0	0

Files

Class	Bugs
com.celfinet.vevo.dataaggregator.domain.NodalAggregation	1
com.celfinet.vevo.dataaggregator.repository.AggregationRepositoryImpl	2
com.celfinet.vevo.dataaggregator.service.AggregationDefinitionServiceImpl	11
com.celfinet.vevo.dataaggregator.service.AggregatorServiceImpl	5
com.celfinet.vevo.dataaggregator.service.FormulaValuesServiceImpl	8
com.celfinet.vevo.dataaggregator.service.SqlStatementBuilderImpl	7

Figure 6.18: SpotBugs report from PerformanceDataAggregator component

Chapter 7

Evaluation

This chapter focuses on the definition, execution and analysis of the results of the solution evaluation.

7.1 Introduction

One of the goals of this project was to develop a system that can create formulas for KPIs and apply them whenever it is necessary to calculate a KPI. The solution needs to process PM data to aggregate the measured counters and get the KPI result.

The objective of this evaluation is to measure the result of the work developed and contribute with data to answer the research question:

What is the effect of aggregation of performance counters to calculate a KPI in a shorter period?

Thus, it was necessary to assess whether all functional requirements have been implemented and correspond to what was requested.

To evaluate the work developed, support the initially defined objectives and the research question it was defined the following hypothesis.This hypothesis specifies the range of network elements to include in the test.

With the processed and aggregated data, the developed system can calculate KPIs in a shorter period than if the data was not aggregated, with a minimum of 10000 and a maximum of 50000 records.

7.2 Evaluation Methodology

The evaluation of the project was carried out when the features of the application were developed. To evaluate the solution from different perspectives, four indicators were defined. These indicators are defined and briefly described in the following list:

1. Accuracy to evaluate the correctness and accuracy of the KPI calculation.
2. Processing time to measure the mean duration of the KPI calculation.
3. Scaling to check the solution components can be scaled-up and scaled-down.

4. Different retention times to access if it is possible to have different retention times for distinct data aggregations.

Sections 7.3 to 7.6 presents the results of the defined indicators.

7.3 Accuracy

This indicator measures the accuracy of the solution when calculating a value for a KPI. This indicator is crucial to guarantee the validity of the solution since it will work as a base for any system that has the necessity of knowing KPI values. If the calculation is wrong, every process that consumes its value will be operating with an incorrect result.

Besides the unit and integration tests included in the components of the system, to test this indicator, it was necessary to create a couple of tests with random PM data. The values generated by the system were compared with the expected values for the KPIs. So, the sources of information are the calculated KPI values.

Thereby, the test cases were created based on the complexity of the formula expression. There were defined three levels of complexity based on the mathematical operators and functions that an expression may contain. The following list presents the defined levels of complexity:

1. Formula with mathematical operators: $+$, $-$, $\times$, $\div$
2. Formula with power and/or square root functions: y^2, $\sqrt{y}$
3. Formula with logarithm function: $\log_y$

With the defined levels of complexity, three KPIs and formulas were created to test the accuracy of the calculated values. The KPIs and measures were defined only for this test case, thus not representing any real performance measure. These formulas were defined for the resource specification **UtranCell** and granularity period **Hour**. In addition, was defined that the measures present in the formulas have a nodal and temporal aggregation of **SUM**. Table 7.1 presents the KPIs and formulas defined for this test case.

Table 7.1: Definition of the formula's expression

KPI	Formula	Expression
kpi-test-1	numerator	$100 * ran\_cell.pm\_eri\_lte\_cell.attr1$
	denominator	$ran\_cell.pm\_eri\_lte\_cell.attr2$
kpi-test-2	numerator	$100 * \sqrt{(ran\_cell.pm\_eri\_lte\_cell.attr3)}$
	denominator	$ran\_cell.pm\_eri\_lte\_cell.attr2$
kpi-test-3	numerator	$log(ran\_cell.pm\_eri\_lte\_cell.attr4)^2$
	denominator	1

Lastly, were generated random values for the variables present in the formulas and a random resource to the test case. The values were created for the resource **UtranCell=CLL1229** and for May 2, 2021, during hour 3. Assuming the data was generated at the UtranCell level with a granularity period of 15 minutes, it must be aggregated by the hour. Table 7.2 represents the generated PM data and the values aggregated by the hour.

Table 7.2: Generated data for the test case

timestamp	attr1	attr2	attr3	attr4
2021-05-02 3:00	1	288	2000	100
2021-05-02 3:15	2	288	1999	100
2021-05-02 3:30	3	288	1998	100
2021-05-02 3:45	4	288	1997	100
2021-05-02 3:00	**10**	**1152**	**7994**	**400**

The KPI was calculated acknowledging the values presented in the last row of table 7.2, followed by the calculation of the KPIs by the PoC.

Table 7.3: Accuracy test results

KPI	Formula	Expected Value	Result Value
kpi-test-1	$\frac{100*ran\_cell.pm\_eri\_lte\_cell.attr1}{ran\_cell.pm\_eri\_lte\_cell.attr2}$	82.25	82.25
kpi-test-2	$\frac{100*\sqrt{(ran\_cell.pm\_eri\_lte\_cell.attr3)}}{ran\_cell.pm\_eri\_lte\_cell.attr2}$	1.83	1.83
kpi-test-3	$\frac{log(ran\_cell.pm\_eri\_lte\_cell.attr4)^2}{1}$	6.77	6.77

Comparing the expected values with the result values from the system, it is possible to conclude that the calculations were done correctly.

7.4 Processing time

This indicator measures the average processing time for a KPI calculation, involving different levels in the network resources and granularity periods. The calculation was performed for the kpi-test-1, defined in section 7.3.

The results from this indicator are important to support the hypothesis defined in the objectives of the evaluation.

To collect the processing times, a set of tests was defined containing two levels of nodal aggregation and one level of temporal aggregation. Besides this, the amount of data to calculate the KPIs was defined having in account the interval of records defined in the hypothesis: 10000 to 50000 records.

Table C.1 presents the NEs and the number of records of each NE involved in the tests. As it is possible to observe, the measured elements belong to the UtranCell resource specification. Thus, regarding nodal aggregations, the tests were defined for three levels: UtranCell, NodeFunction and SubNetwork because are the levels present in the NEs path, as shown in table C.1.

The data involved in these tests represents the period from 2021-10-02 00:00:00 to 2021-05-30 05:00:00, with a granularity period of 15 minutes. Thereby, the temporal aggregation defined for the tests was Hour.

Since the PoC was developed having in account the pre-defined aggregations in the system, it was implemented a simple mechanism that when a flag is active, deactivates those aggregations and forces the system to perform the aggregations on-demand,. With this, it was possible to collect the times for the calculations with pre-defined and on-demand aggregations.

Table C.2 presents the defined test executions to measure the meantime of the calculation of KPIs when the pre-defined aggregations are present and when they are not present. The number of records present in the last column presents the number of unique pair NE/timestamp that the system uses to calculate the KPIs.

To have objectives results and conclusions, statistical tests were carried out using hypothesis testing. Hypothesis testing "allows the comparison of the two formulations to be made on objective terms, with knowledge of the risks associated with reaching the wrong conclusion" (Montgomery 2013).

The R language and RStudio were used to execute the hypothesis tests. Thereby, the test parametric t-test or the Wilcoxon test were chosen when comparing two samples, having into account the data distribution. In the scope of the test of the two independent variables, were followed the succeeding steps:

1. Obtain a sample of 50 executions of a given independent variable, being transmitted the same data and in each attempt for each aggregation and noted the execution time, in seconds. To automatize the tests defined above JMeter, an "application designed to load test functional behaviour and measure performance" (JMeter 2019) was used. Figure 7.1 presents a test plan defined in JMeter.

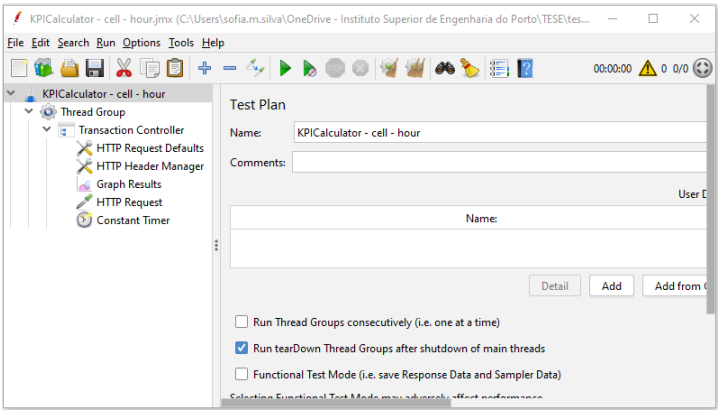


Figure 7.1: JMeter test plan

2. Generate box plots to visualize and compare the data between aggregations (pre-defined and on-demand)
3. Carry out Lilliefors (Kolmogorov-Smirnov) test to verify that the samples follow a distribution normal.
4. If there is no normal distribution, the Wilcoxon will be used, otherwise the t-test test.
5. Perform the t-test / Wilcoxon.

UtranCell-Hour

The times collected for the calculation of a KPI at UtranCell-Hour aggregation level are present in table C.3, in Appendix C. Figure 7.2 presents a box plot to visualize the data in a graphical representation.

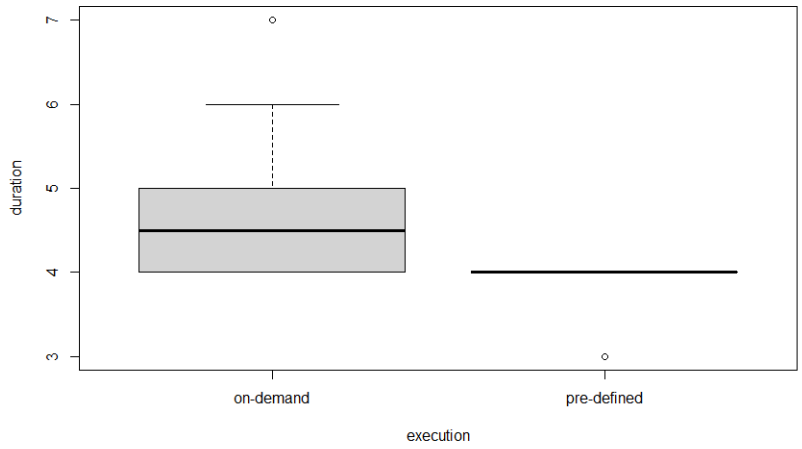


Figure 7.2: Box plot of UtranCell-Hour calculations

Analysing the box plot it is possible to observe that the calculations with pre-defined aggregation are almost constant, at 4 seconds. Regarding the calculations with the aggregation on-demand, it is possible to conclude that 50% of the time the duration of the execution was bigger than 4.5 seconds. Thus, the calculation performed with the pre-defined aggregation is faster than the calculation performed with the aggregation on-demand. To prove this, it was defined the following hypothesis:

$$H_0 : \mu_1 = \mu_2 \tag{7.1a}$$
$$H_1 : \mu_1 > \mu_2 \tag{7.1b}$$

where μ_1 is the mean duration of the calculation with on-demand aggregation and μ_2 is the mean duration of the calculation with pre-defined aggregation.

Since the data does not follow a normal distribution (see section C.1.1) it was carried out the Wilcox test to test hypothesis 7.1.

```
> wilcox.test(duration~execution,  alt="greater", mu=0, conf.level=0.95, paired=F)

        Wilcoxon rank sum test with continuity correction

data:  duration by execution
W = 1887.5, p-value = 4.768e-09
alternative hypothesis: true location shift is greater than 0
```

Figure 7.3: Wilcox test for UtranCell-Hour

Observing the results in figure 7.5, p-value = 4.768e-09 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, the mean duration of the calculation with on-demand aggregations is greater than the mean duration of the calculation with pre-defined aggregations.

NodeFunction-Hour

The times collected for the calculation of a KPI at NodeFunction-Hour aggregation level are present in table C.4, in Appendix C. Figure 7.4 presents a box plot to visualize the data in a graphical representation.

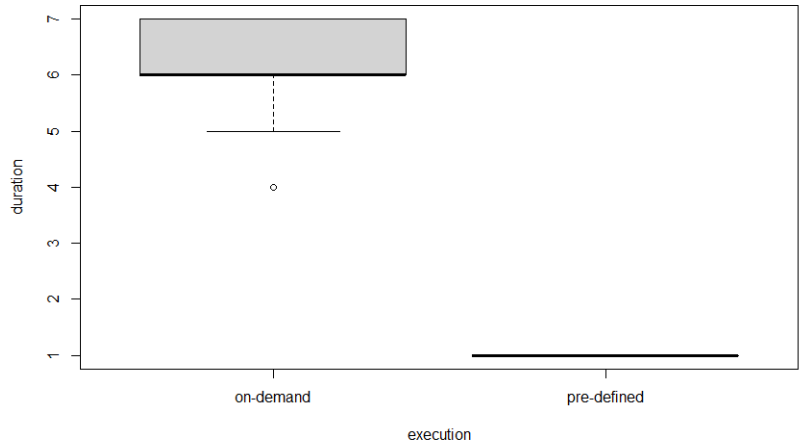


Figure 7.4: Box plot of NodeFunction-Hour calculations

Analysing the box plot it is possible to observe that the calculations with pre-defined aggregation are constant, at 1 second. On the other hand, the calculations with the aggregation on-demand have more dispersed data with a minimum execution time of 5 seconds. Thus, the calculation performed with the pre-defined aggregation is faster than the calculation performed with the aggregation on-demand. To prove this, it was defined the following hypothesis:

$$H_0 : \mu_1 = \mu_2 \tag{7.2a}$$
$$H_1 : \mu_1 > \mu_2 \tag{7.2b}$$

where μ_1 is the mean duration of the calculation with on-demand aggregation and μ_2 is the mean duration of the calculation with pre-defined aggregation for the NodeFunction-Hour level.

Since the data does not follow a normal distribution ((see section C.1.2)) it was carried out the Wilcox test to test hypothesis 7.2.

```
> wilcox.test(duration~ï..execution,  alt="greater", mu=0, conf.level=0.95, paired=F)

        Wilcoxon rank sum test with continuity correction

data:  duration by ï..execution
W = 2500, p-value < 2.2e-16
alternative hypothesis: true location shift is greater than 0
```

Figure 7.5: Wilcox test for NodeFunction-Hour

Observing the results in figure 7.5, p-value = 2.2e-16 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, the mean duration of the calculation with on-demand aggregation is greater than the mean duration of the calculation with pre-defined aggregation.

SubNetwork-Hour

The times collected for the calculation of a KPI at SubNetwork-Hour aggregation level are present in table C.5, in Appendix C. Figure 7.6 presents a box plot to visualize the data in a graphical representation.

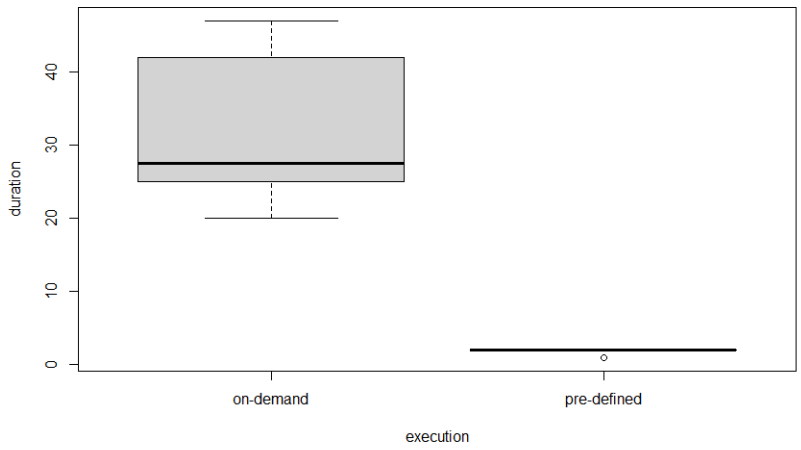


Figure 7.6: Box plot of SubNetwork-Hour calculations

Analysing the box plot it is possible to observe that the calculations with pre-defined aggregation do not pass the 5 seconds. The processing times for the calculations with the aggregation on-demand, are very dispersed and have a minimum value of approximately 20 seconds and a maximum of approximately 50 seconds. Thus, the calculation performed with the pre-defined aggregation is faster than the calculation performed with the aggregation on-demand. To prove this, it was defined the following hypothesis:

$$H_0 : \mu_1 = \mu_2 \tag{7.3a}$$
$$H_1 : \mu_1 > \mu_2 \tag{7.3b}$$

where μ_1 is the mean duration of the calculation with on-demand aggregation and μ_2 is the mean duration of the calculation with pre-defined aggregation for the SubNetwork-Hour level.

Since the data does not follow a normal distribution (C.1.3) it was carried out the Wilcox test to test hypothesis 7.3.

```
> wilcox.test(duration~i..execution,  alt="greater", mu=0, conf.level=0.95, paired=F)

        Wilcoxon rank sum test with continuity correction

data:  duration by i..execution
W = 2500, p-value < 2.2e-16
alternative hypothesis: true location shift is greater than 0
```

Figure 7.7: Wilcox test for SubNetwork-Hour

Observing the results in figure 7.7, p-value = 2.2e-16 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, the mean duration of the calculation with on-demand aggregation is greater than the mean duration of the calculation with pre-defined aggregation.

Conclusion

The execution times were used to prove that the average duration of the execution with and without pre-defined aggregation is not the same. The results indicate that the mean calculation processing time with pre-defined aggregations is shorter than the mean processing time for the calculation with aggregations on-demand.

These results also support the hypothesis defined in section 7.1. Besides this, the results of the hypothesis tests contribute to the answer to the research question present in section 7.1.

7.5 Scaling

Since the scalability quality attribute is significant for the developed application, this indicator assesses whether it is possible to scale any service when necessary.

Celfinet uses Kubernetes to automate the "deployment, scaling and management of containerized applications" (Kubernetes 2021b). This platform provides service discovery, in other words, it "can expose a container using the DNS name or using their IP address" (Kubernetes 2021c) and can "load balance and distribute the network traffic" (Kubernetes 2021c) to have a stable deployment.

As defined in section 5.3, the system deployment is based on the pattern service instance per container. Thus, Kubernetes is also used to manage the developed PoC.

Kubernetes can automatically scale the number of pods based on CPU utilization or other metrics. A pod "is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers" (Kubernetes 2021d). It is also the smallest unit of computing that can be deployed in Kubernetes.

The Horizontal Pod Autoscaler controller mechanism provided by Kubernetes is responsible for adjusting the number of replicas by observing the metrics collected within 15 seconds. During this period, it "queries the resource utilization against the metrics specified" (Kubernetes 2021a). It is based on a single algorithm to find the number of replicas that is present below (Kubernetes 2021a):

$$desiredReplicas = ceil[currentReplicas*(currentMetricValue/desiredMetricValue)]$$

To Horizontal Pod Autoscaler determines the desiredReplicas, it must be specified threshold value for the resource usage. This configuration is simple, as shown in figure 7.8. In this example, the Horizontal Pod Autoscaler will retain the average utilization of the pods in the scaling mark at 60 per cent of the CPU resource usage.

```yaml
type: Resource
resource:
  name: cpu
  target:
    type: Utilization
    averageUtilization: 60
```

Figure 7.8: Horizontal Pod Autoscaler. Source (Kubernetes 2021a)

To manage Kubernetes clusters exists a platform called Rancher[1]. This "provides an intuitive user interface for DevOps engineers to manage their application workload" (Rancher 2021).

Rancher also provides features to manage the Horizontal Pod Autoscaler provided by Kubernetes. Nonetheless, Rancher supplies a graphical interface to manage the workloads that give the possibility to scale up and scale down the components manually.

Figure 7.9 shows a snippet of the management view for the KPICalculator workload. In this view, it is possible to see an option *Config Scale* that allows to scale up the application when clicking in the + icon and scale down by clicking on the - icon. In this case, there are two running instances of the KPICalculator application.

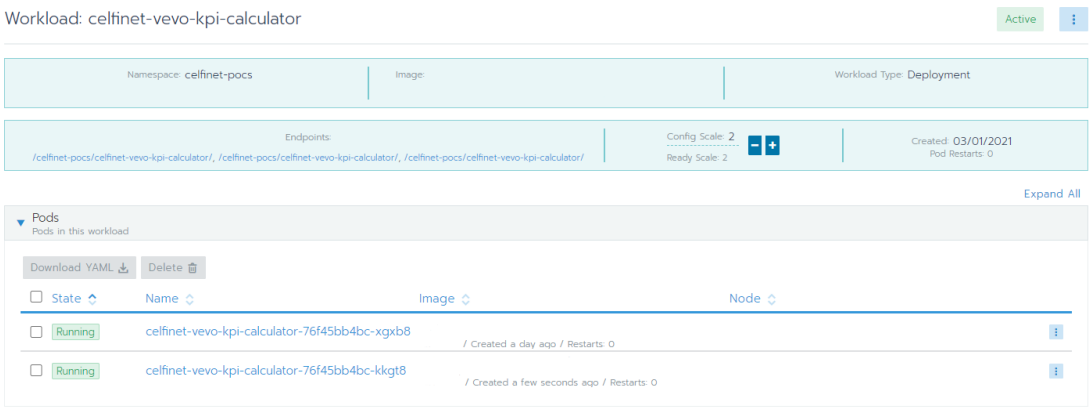


Figure 7.9: Management view of the workload KPICalculator in Rancher

[1]https://rancher.com/

To conclude, it is possible to say that the solution is scalable, given its importance for the PoC (see section 5.2). Using Kubernetes allow to scale up and scale down the applications that compose the PoC by configuring the desired resources and thresholds of usage. In addition, as applications need to use databases and a message broker, they may also need to be scaled according to the number of instances of the applications.

7.6 Retention times

This indicator evaluates if it is possible to have different retention times for processed PM.

The PoC was developed by having in account Approach 2 - Pre-defined Aggregations, described in section 4.3.2. Thereby, besides the existing raw data, there are tables with the aggregated data. Figure 7.10 presents possible aggregations of the raw data.

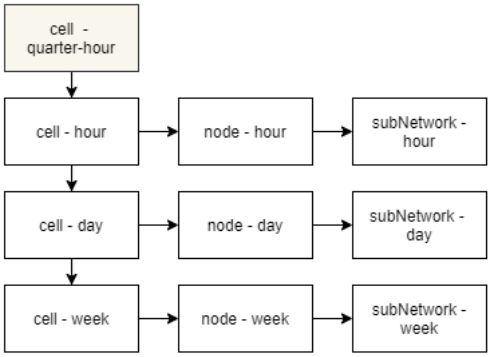


Figure 7.10: Example of aggregation fall-back of PM data

As there can be multiple pre-defined aggregations involving distinct network levels and granularity periods, there is a possibility to configure the retention period for each one. As present in section 5.5, the configurations of the pre-defined aggregations are in the AggregationDefinition table. Each one has a `retentionDays` attribute associated. With this, it is possible to have distinguished retention days for the pre-defined aggregations.

According to the example present in figure 7.10, the aggregation cell-day can have a retention period of 90 days and the cell-week a retention period of 365 days.

A mechanism could be implemented that removes records that classifies as outdated to ensure that PM data is within the retention period. It can be a task scheduled to run every day to remove the oldest data from the aggregation tables.

Since there can be multiple pre-defined aggregation definitions for the PM data, the date from which the data is rejected needs to be calculated. The calculation is performed by subtracting the current date with the number of days configured.

7.7 Overall Evaluation

The results of the evaluation of the defined indicators are generally positive. The system correctly calculates KPIs and allows its components to be scaled. Furthermore, when performing aggregations with the PM data, the calculation time of the KPIs decreased and supports having data sets with different time intervals.

The Quantitative Evaluation Framework (QEF) tool was utilised to complement the evaluation of the indicators. The QEF has been used to evaluate the completeness and fulfilment of the requirements of the solution. This framework measures the percentage of compliance with each of the requirements, defined in chapter 4, intending to validate and guarantee the QoS (Escudeiro 2003).

The quality benchmark is composed of dimensions, which are constituted by factors and made up of requirements. To evaluate the QoS of the PoC it was defined two dimensions, with the following factors, in the QEF:

- **Functionality**: Use cases, System functionalities and Content quality.
- **Quality**: Scalability, Manageability, Maintainability, Technical Documentation and Reusability.

The quality benchmark was created regarding the requirements in chapter 4 and its importance for the PoC. The QEF representation of the project is described in detail in appendix C.2, with the calculated quality of the system. Thus, observing figure C.4, it is possible to conclude that the PoC was an overall quality of 94%.

Furthermore, it was possible to use the developed system in another PoC regarding anomaly detection and collect the feedback. Several KPIs and formulas used by telecommunication operators were configured in the KPI Inventory microservice. With this, it is possible to understand that this framework manages formulas for KPIs with PM raw counters or calculated KPIs. In addition, to increase the precision of the KPI value, the number of decimal places could be configurable.

However, in this integration, the formulas were all of the granularity period of the PM data, so it was not necessary to perform aggregations. Consequently, was not collected the feedback about the PM data aggregations.

Chapter 8

Conclusions

This chapter presents the project conclusions, including the objectives achieved, limitations and future work and final assessment.

In the last phase of the adopted research methodology, ADR, the researchers outline the achievements in the IT artefact and describe the outcomes to formalize the learning (Sein et al. 2011).

8.1 Work Summary

All in all, a PoC was developed to demonstrate the creation of KPIs and formulas and the calculation of the values for the KPIs. Likewise, the PoC also showed that it is possible to have an operator, vendor and technology agnostic system for PM data. Looking at the solution presented in chapters 5 and 6, the result is considered positive. The objectives proposed in section 1.3 were accomplished with the development of the system, having an overall quality of 94% (see section C.2).

At this stage, there is sufficient information to answer the research question defined in chapter 1. It aims to investigate the effect of the aggregation of the performance counters on the calculation time of one KPI:

What is the effect of aggregation of performance counters to calculate a KPI in a shorter period?

From the results of the hypothesis tests (see section 7.4), it is possible to confirm with 95% confidence that the calculations executed with pre-defined aggregations are faster than the calculations done with on-demand ones. As a result, the response to the research question is: by applying pre-aggregations to the performance counters the KPI calculation becomes faster.

As this solution is generic, Celfinet has a new base for a KPI calculation system that could be part of multiple projects. As mentioned in section 7.7, this framework was incorporated in another PoC. Thus is possible to conclude that any environment that needs to calculate KPIs can include this KPI calculation system.

8.2 Project Contributions

The research methodology, ADR, deals with two challenges: "(1) addressing a problem situation encountered in a specific organizational setting by intervening and evaluating; and (2) constructing and evaluating an IT artefact that addresses the class of problems typified by the encountered situation" (Sein et al. 2011).

Celfinet has been addressing some specific use cases related to the PM and KPI calculation. This project addresses a class of problems as the solution is vendor and technology agnostic and can adapt to different contexts. Moreover, as it allows the creation of KPIs and formulas, it addresses several problems related to standard and operator-specific formulas.

This project will allow to Celfinet address a class of problems and include a PM feature in Vismon Manager. As mentioned in chapter 3, a generic base for KPI management and calculation provides Celfinet with the possibility to capture a competitive advantage.

8.3 Limitations and Future Work

This section sets out the limitations assessed throughout the project and approaches future work for the solution. Every solution needs continuous evolution and maintenance, and the KPI Calculation framework is not different. Regardless of the effort put into the development of this project, not everything has been accomplished with complete success, as is possible to see in QEF (see appendix C.2).

The PoC only supports mathematical formulas, and for that, this is considered a limitation since there exist logical expressions to calculate a KPI. Besides this, to configure a formula it is only possible to add attributes configured in the system. Thus, as future work, the KPI Formula inventory could be modified to allow the inclusion of other KPIs as formula variables.

Also, the KPI calculation framework can be extended to correlate PM and CM data. This correlation will allow the calculation of the nodal integrity for the KPI value because CM data provides the configuration of all the NEs. Furthermore, associating these two data types it is possible to insert CM parameters in a formula of a KPI. In addition, could be added to the inventory component the possibility of manage KQI.

Regarding the pre-defined aggregations, the system presents some limitations. As this solution is generic, there is a possibility to configure any attribute of PM files. Consequentially, the identification of the resource may not contain its resource path. This situation could be considered a limitation of the system since it was designed with the standard PM file type of 3GPP.

Besides this, QEF proves that the aggregation requirements were not completely fulfilled. Therefore, as future work for the company, the product team needs to refine the component responsible for the aggregations.

Regarding quality attributes, the solution evaluation presents limitations in Scalability and Manageability. The scalability attribute was not evaluated in a simulation to verify if the applications can be scaled-up or scaled-down when it is necessary. Despite this, it is possible to scale any component with the adoption of the microservices architecture style, the containers and Kubernetes.

The manageability requirement requires the logs to be in a centralized database. The POC only generate logs locally, so as consequence, this attribute was not completely fulfilled.

Finally, the KPI calculation system could be refined to contemplate approach 3 - Pre-defined and on-demand Aggregations (see section 4.3.3). This approach calculates the KPI value with volatile and aggregated data and does not discard data that arrives late, duplicates or versioned. Consequently, the system will become more robust and trustable.

8.4 Concluding Remarks

The project was a great way to deepen knowledge related to the architectural style based on microservices. This application of this style has been growing in the development of computer systems software. In addition, it also provided learning of new concepts related to telecommunications.

The documentation executed in this project provides the foundation for a system that imports PM data and calculates KPIs. The construction of the system considered the customer requirements to ensure its quality and potential. Indeed, the world is increasingly dependent on telecommunications infrastructures and their proper functioning.

Therefore, it is expected that the research and work developed will be beneficial and included in new systems that the company needs to evolve.

Bibliography

(ITU), International Telecommunication Union (2008). "ITU-T: E.800 - Definitions of terms related to quality of service". In: *The American Mathematical Monthly*, p. 21. issn: 00029890. doi: 10.2307/2304469.

3GPP Specification, Technical and Group Services (2020a). *3GPP TS 32.401: 3rd Generation Partnership Project - Telecommunication management - Performance Management (PM) - Concept and requirements*. Tech. rep. Release 16.

3GPP Specification, Technical and Group Services (2020b). *3GPP TS 32.432: 3rd Generation Partnership Project - Telecommunication management - Performance measurement - File format definition*. Tech. rep. Release 16.

3GPP Specification, Technical and Group Services (2020c). "3GPP TS 32.450: 3rd Generation Partnership Project - Telecommunication management - Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN): Definitions". In: 0.Release 16.

Aditi (2021). *Agile Methodology based Services*. url: http://www.aditicorp.com/services/agile-methodology-based-services/ (visited on 06/14/2021).

Apache Foundation (2017). "Apache Kafka Documentation". In: pp. 449–456. doi: 10.3139/9783446456013.012. url: https://kafka.apache.org/documentation/.

Baeldung (2021). *Introduction to Spring Data Cassandra | Baeldung*. url: https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa%20https://www.baeldung.com/spring-data-cassandra-tutorial (visited on 05/17/2021).

Bass, L, P C Clements, and R Kazman (2013). *Software Architecture in Practice*. isbn: 9780321711502.

Cervantes, Humberto and Rick Kazman (2016). *Designing Software Architecture: A Practical Approach*. Pearson Education, Inc., p. 388. isbn: 9780134390789.

Checkstyle (2021). *Checkstyle*. url: https://checkstyle.sourceforge.io/index.html (visited on 05/23/2021).

Clements, Paul et al. (2011). *Documenting Software Architectures Views and Beyond*. Pearson Education, Inc. isbn: 9783642253874.

Comarch (2021). *Performance Management*. url: https://www.comarch.com/telecommunications/service-assurance/performance-management/.

Escudeiro, Paula (2003). "Quantitative Evaluation Framework". In: January 2008.

Fowler, Martin (2021a). *P of EAA: Data Transfer Object*. url: https://martinfowler.com/eaaCatalog/dataTransferObject.html (visited on 05/17/2021).

Fowler, Martin (2021b). *P of EAA: Repository*. url: https://martinfowler.com/eaaCatalog/repository.html (visited on 05/17/2021).

GSM Association (2020). "The Mobile Economy 2020". In: *Gsma*, pp. 2–62. url: https://www.gsma.com/.

HELIX PERFORMANCE Gain deep network insights (n.d.).

Humble, Jezz and David Farley (2011). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, Inc, p. 512. isbn: 978-0-321-60191-9. url: https://books.google.pt/books?hl=en&lr=&id=6ADDuzere-

YC&oi=fnd&pg=PT30&dq=+Continuous+Delivery:+Reliable+Software+Releases+through+Build,+Test,+and+Deployment+Automation&ots=-wqoSRH5kc&sig=pUSkQP-POyu8OnsaAS2w55Z-mUw&redir_esc=y#v=onepage&q=acceptance&f=false.

International, Quality-One (2019). *QFD | Quality Function Deployment | Quality-One*. url: https://quality-one.com/qfd/ (visited on 01/24/2021).

ITU-T (2000). "TMN Management Functions". In: *ITU-T Recommendation M.3400* 3400. url: https://www.itu.int/rec/T-REC-M.3400/en.

Ivankovi, Marko et al. (2019). "Code coverage at Google". In: *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 955–963. isbn: 9781450355728. doi: 10.1145/3338906.3340459. url: https://doi.org/10.1145/3338906.3340459.

Jiang, Shuai and Huaxin Mu (2011). "Design patterns in object oriented analysis and design". In: *ICSESS 2011 - Proceedings: 2011 IEEE 2nd International Conference on Software Engineering and Service Science*, pp. 326–329. doi: 10.1109/ICSESS.2011.5982229.

JMeter, Apache (2019). *Apache JMeter*. url: https://jmeter.apache.org/%20https://jmeter.apache.org/%0Ahttps://jmeter.apache.org/%0Ahttp://jmeter.apache.org/ (visited on 05/20/2021).

Koen, P A et al. (1996). "Fuzzy Front End : and Techniques". In: *Industrial Research* pp, pp. 5–35. url: http://www.stevens.edu/cce/NEW/PDFs/FuzzyFrontEnd%7B%5C_%7DOld.pdfNEW/PDFs/FuzzyFrontEnd%7B%5C_%7DOld.pdf.

Koen, Peter (2013). *Front End Innovation*. url: http://frontendinnovation.com/fei%20http://www.slideshare.net/BrandGenetics/koen-fei (visited on 02/22/2021).

Kreher, R (2006). *UMTS Performance Measurement: A Practical Guide to KPIs for the UTRAN Environment*. Germany: Published in 2006 by John Wiley & Sons Ltd. isbn: 9780470034873. url: http://books.google.com/books?id=eapzGALS%7B%5C_%7DN0C.

Kruchten, Philippe B. (1995). *The 4+1 View Model of Architecture*. Vol. 12. 6, pp. 42–50. doi: 10.1109/52.469759.

Kubernetes (2021a). *Horizontal Pod Autoscaler*. url: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (visited on 05/23/2021).

Kubernetes (2021b). *Production-Grade Container Orchestration*. url: https://kubernetes.io/ (visited on 05/23/2021).

Kubernetes (2021c). *What is Kubernetes?* url: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (visited on 05/23/2021).

Kubernetes (2021d). *What is Kubernetes?* url: https://kubernetes.io/docs/concepts/workloads/pods/ (visited on 05/23/2021).

Kuklinski, Slawomir and Lechoslaw Tomaszewski (2019). "Key performance indicators for 5g network slicing". In: *Proceedings of the 2019 IEEE Conference on Network Softwarization: Unleashing the Power of Network Softwarization, NetSoft 2019*, pp. 464–471. doi: 10.1109/NETSOFT.2019.8806692.

Kumar, Gaurav and Pradeep Bhatia (2012). "Impact of Agile methodology on software development process". In: *International Journal of Computer Technology and Electronics Engineering (IJCTEE)* 2.4, pp. 46–50.

Leal, José Eugenio (2020). "AHP-express: A simplified version of the analytical hierarchy process method". In: *MethodsX* 7. issn: 22150161. doi: 10.1016/j.mex.2019.11.021.

Lee, Paul et al. (2019). "Cycling's technological transformation". In: *Technology,Media, and Telecommunications Predictions 2020*, pp. 118–126. url: https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/technology-media-telecommunications/deloitte-cn-tmt-prediction-full-report-2020-en-200217.pdf.

Li, Li and Subin Shen (2014). "End-to-End QoS Performance Management Across LTE Networks". In: 2011, pp. 45–58. doi: 10.1007/978-3-319-07389-7_5.

Martinez-Mosquera, Diana and Rosa Navarrete (2020). "Development and Evaluation of a Big Data Framework for Performance Management in Mobile Networks". In: pp. 1–18. doi: 10.1109/ACCESS.2020.3045175.

Microsoft (2021a). *Plan, code, collaborate, ship applications - Azure DevOps | Microsoft Docs*. url: https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops (visited on 04/27/2021).

Microsoft (2020). *Saga distributed transactions pattern*. url: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/saga/saga (visited on 01/27/2021).

Microsoft (2021b). "Choose whether to use messages or events - Learn | Microsoft Docs". In: url: https://docs.microsoft.com/en-us/learn/modules/choose-a-messaging-model-in-azure-to-connect-your-services/2-choose-whether-to-use-message-queues-or-events.

Microsoft Corporation (2009). *Microsoft Application Architecture Guide*, p. 560. isbn: 9780735627109. url: http://www.microsoft.com/en-us/download/details.aspx?id=16236.

Montgomery, Douglas C (2013). *Design and Analysis of Experiments*. Eighth Edi. John Wiley & Sons, Inc. isbn: 9781118146927. doi: 10.1002/9783527809080.cataz11063.

Newman, Sam (2015). *Building Microservices*. O'Reilly Media, Inc, p. 280. isbn: 978-1-491-95035-7. url: https://learning.oreilly.com/library/view/building-microservices-2nd/9781492034018/ch01.html%7B%5C#%7Dintro-chapter%20https://www.google.hr/books?hl=en%7B%5C&%7Dlr=%7B%5C&%7Did=jjl4BgAAQBAJ%7B%5C&%7Dpgis=1%7B%5C%%7D5Cnhttp://%20oreilly.com/catalog/errata.csp?isbn=9781491950357.

Nicola, Susana, Eduarda Pinto Ferreira, and J.J. Pinto Ferreira (2012). "International Journal of Information Technology & Decision Making". In: *International Journal of Information Technology & Decision Making*, pp. 661–703.

NOKIA (2016). *Operating Documentation , Issue 06 WCDMA RAN Key*.

Olan, Michael (2003). "Unit testing: test early, test often". In: *Journal of Computing Sciences in Colleges* 19.2, pp. 319–328. issn: 1937-4771.

Oracle (2009). *Cron Expressions*. url: https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm (visited on 05/16/2021).

Osterwalder, Alexander and Yves Pigneur (2003). "Modeling value propositions in e-business". In: *ACM International Conference Proceeding Series* 50, pp. 429–436. doi: 10.1145/948005.948061.

Ramalingam, Chandra (July 2020). *Building Domain Driven Microservices*. url: https://medium.com/walmartglobaltech/building-domain-driven-microservices-af688aa1b1b8 (visited on 01/26/2021).

Rancher (2021). *Overview*. url: https://rancher.com/docs/rancher/v2.x/en/overview/ (visited on 05/23/2021).

Rich, Nick and Matthias Holweg (2000). "VALUE ANALYSIS - Value Engineering". In: *INNOREGIO: dissemination of innovation and knowledge management techniques*, pp. 0–31.

Richards, Mark (2015). *Software Architecture Patterns*. O'Reilly Media, Inc. isbn: 9781491924242. url: https://learning.oreilly.com/library/view/software-architecture-patterns/9781491971437/%20http://www.ncbi.nlm.nih.gov/pubmed/24798474.

Richardson, Chris (2019a). *Command Query Responsibility Segregation (CQRS)*. url: https://microservices.io/patterns/data/cqrs.html (visited on 01/28/2021).

Richardson, Chris (2019b). *Database per Service*. url: https://microservices.io/patterns/data/database-per-service.html (visited on 01/27/2021).

Richardson, Chris (2020a). *Deploying microservices*. url: https://microservices.io/articles/deployment.html (visited on 02/20/2021).

Richardson, Chris (2020b). "Pattern: Service instance per container". In: *Microservice architecture*. url: https://microservices.io/patterns/deployment/service-per-container.html.

Sathyan, Jithesh (2016). *Fundamentals of EMS, NMS and OSS/BSS*. Auerbach Publications. doi: 10.1201/b15748. url: https://learning.oreilly.com/library/view/fundamentals-of-ems/9781420085747/.

Sein, Maung K. et al. (2011). "ACTION DESIGN RESEARCH". In: *MIS Quarterly* 35.1, pp. 37–56.

Shahin, Mojtaba, Muhammad Ali Babar, and Liming Zhu (2017). "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices". In: *IEEE Access* 5.Ci, pp. 3909–3943. issn: 21693536. doi: 10.1109/ACCESS.2017.2685629.

Sinche, Soraya et al. (2020). "A Survey of IoT Management Protocols and Frameworks". In: *IEEE Communications Surveys and Tutorials* 22.2, pp. 1168–1190. issn: 1553877X. doi: 10.1109/COMST.2019.2943087.

SourceMaking (2021a). *Builder Design Pattern*. url: https://sourcemaking.com/design_patterns/builder (visited on 04/25/2021).

SourceMaking (2021b). *Facade Design Pattern*. url: https://sourcemaking.com/design_patterns/facade (visited on 04/25/2021).

SpotBugs (2021). *SpotBugs*. url: https://spotbugs.github.io/ (visited on 05/23/2021).

Swagger (2021). *About Swagger Specification | Documentation*. url: https://swagger.io/docs/specification/about/ (visited on 04/27/2021).

Ulaga, Wolfgang and Andreas Eggert (Jan. 2006). *Value-based differentiation in business relationships: Gaining and sustaining key supplier status*. doi: 10.1509/jmkg.2006.70.1.119. url: http://journals.ama.org/doi/abs/10.1509/jmkg.2006.70.1.119.

Zhang, Xincheng (2017). *LTE Optimization Engineering Handbook*. isbn: 9781119159001. doi: 10.1002/9781119158981.

Appendix A

AHP Method

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Figure A.1: IR values for square n matrices of order n

A.1 Phase 3

For phase 3, the standardized matrix of criteria was constructed and the relative priorities for the criteria were calculated. Results are presented in table A.1.

Table A.1: Standardized matrix of the second level with relative priority

	Estimated Processing Time	**Different retention times**	**Need for re-processing**	**Development time**	**Relative priority**
Estimated Processing Time	0.27	0.27	0.31	0.26	0.28
Different retention times	0.13	0.13	0.15	0.13	0.14
Need for reprocessing	0.07	0.07	0.08	0.09	0.07
Development time	0.53	0.53	0.46	0.52	0.51

A.2 Phase 4

The priority vector (x) corresponds to the column referring to the relative priorities in table A.1, so:

$$x = (0.28; 0.14; 0.07; 0.51)$$

To calculate the eigenvalue (λ_{max}) it is considered that:

$$Ax = \lambda_{max} x$$

So A, corresponds to the comparison matrix represented in the table 4.6:

$$\begin{bmatrix} 1 & 2 & 4 & 1/2 \\ 1/2 & 1 & 2 & 1/4 \\ 1/4 & 1/2 & 1 & 1/6 \\ 2 & 4 & 6/1 & \end{bmatrix} x \begin{bmatrix} 0.28 \\ 0.14 \\ 0.07 \\ 0.51 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.28 \\ 0.14 \\ 0.07 \\ 0.51 \end{bmatrix} \Leftrightarrow \tag{A.1}$$

$$\Leftrightarrow \begin{bmatrix} 1.10 \\ 0.55 \\ 0.30 \\ 2.06 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.28 \\ 0.14 \\ 0.07 \\ 0.51 \end{bmatrix} \Leftrightarrow \tag{A.2}$$

$$\lambda_{max} = average\left\{\frac{1.10}{0.28}, \frac{0.55}{0.14}, \frac{0.30}{0.07}, \frac{2.06}{0.51}\right\} \simeq 4.01 \tag{A.3}$$

With the eigenvalue (λ_{max}) calculated, and bearing in mind that n corresponds to the number of criteria, the Consistency Index (CI) is:

$$CI = \frac{\lambda_{max} - n}{n-1} = \frac{4.01 - 4}{4-1} \simeq 0.0035 \tag{A.4}$$

To calculate the CR it is necessary to know the value of IR. This value is presented in figure A.1, and for the current situation, using three criteria, IR value is 0.90. Considering the formula for CR, then:

$$RC = \frac{IC}{IR} = \frac{0.0035}{0.9} \simeq 0.004 < 0.1 \tag{A.5}$$

A.3 Phase 5

Criteria - Estimated Processing Time

Table A.2: Second level estimated processing time comparison matrix

Estimated Processing Time	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**
Approach 1	1	1/4	1/3	1/5
Approach 2	4	1	2	2/3
Approach 3	3	1/2	1	1/3
Approach 4	5	2	3	1
Total	13	3.75	6.33	2.20

Criteria - Different retention times

Table A.3: Standardized matrix of estimated processing time of the second level

Estimated Processing Time	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**	**Relative priority**
Approach 1	0.07	0.07	0.05	0.09	0.07
Approach 2	0.31	0.27	0.32	0.30	0.30
Approach 3	0.23	0.13	0.16	0.15	0.17
Approach 4	0.38	0.53	0.47	0.45	0.46

Table A.4: Second level different retention times comparison matrix

Different retention times	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**
Approach 1	1	1/3	1/4	1/3
Approach 2	3	1	1/3	2
Approach 3	4	3	1	3
Approach 4	7/2	2/3	1/2	1
Total	11.5	5.00	2.08	6.33

Table A.5: Standardized matrix of different retention times of the second level

Different retention times	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**	**Relative priority**
Approach 1	0.09	0.07	0.12	0.05	0.08
Approach 2	0.26	0.20	0.16	0.32	0.23
Approach 3	0.35	0.60	0.48	0.47	0.48
Approach 4	0.30	0.13	0.24	0.16	0.21

Criteria - Need for reprocessing

Table A.6: Second level need for reprocessing comparison matrix

Need for reprocessing	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**
Approach 1	1	1/2	2	1/3
Approach 2	2	1	4	2/3
Approach 3	1/2	1/4	1	1/6
Approach 4	3	2	6	1
Total	6.50	3.75	13.00	2.17

Criteria - Development time

Table A.7: Standardized matrix of need for reprocessing of the second level

Need for reprocessing	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**	**Relative priority**
Approach 1	0.15	0.13	0.15	0.15	0.15
Approach 2	0.31	0.27	0.31	0.31	0.30
Approach 3	0.08	0.07	0.08	0.08	0.07
Approach 4	0.05	0.53	0.46	0.46	0.38

Table A.8: Second level development time comparison matrix

Development time	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**
Approach 1	1	1/2	5	3
Approach 2	2	1	4	2
Approach 3	1/5	4	1	1/2
Approach 4	1/2	2	2	1
Total	3.70	7.50	12.00	6.50

Table A.9: Standardized matrix of development time of the second level

Development time	**Approach 1**	**Approach 2**	**Approach 3**	**Approach 4**	**Relative priority**
Approach 1	0.27	0.07	0.42	0.46	0.30
Approach 2	0.54	0.13	0.33	0.31	0.33
Approach 3	0.05	0.53	0.08	0.08	0.19
Approach 4	0.14	0.27	0.17	0.15	0.18

Appendix B

Attribute Driven Design

In this chapter, some key concepts about microservices architecture pattern, layered architecture pattern and microkernel architecture pattern are presented.

B.1 Iteration 1: Establishing an Overall System Structure

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

Microservices Architecture Pattern

Microservice architecture was emerged as a trend or a pattern as consequence of years of experience in building systems. We have been adopting new technologies and better ways to deliver software faster.

"Microservices are independently releasable services that are modelled around a business domain" (Newman 2015). These are small, autonomous and work together to build a more complex system. This architecture style enables the continuous delivery and deployment, improves maintainability and allows the development to be organized around multiple teams. On the other hand, it improves fault isolation since if an error occurs in a service, only that service is affected and not the complete system. Using a monolith architecture, where the logic is in one large component, an error can lead to a failure of the entire system. Figure B.1 shows the comparison between a monolithic application and an application based in microservices architecture.

Besides this, microservices architecture presents more advantages as:

1. **Technology Heterogeneity** - rather than have a system with a standard technology, the services can choose the right tool that fulfils all the necessities.
2. **Scaling** - smaller and independent services allow the scale of those services that need to be scaling and not the whole system
3. **Ease of Deployment** - with this architecture, services are deployed independently of the rest of the system, resulting in a faster deploy and a faster delivery to the customer.

Microservices embrace the information hiding principle to guarantee a clear separation between what can and can not be changed. A component choose what to expose via external interfaces and tries to hide as much information as possible. To guarantee a system with loose coupling and high cohesion, services need to have a stable network interface. All

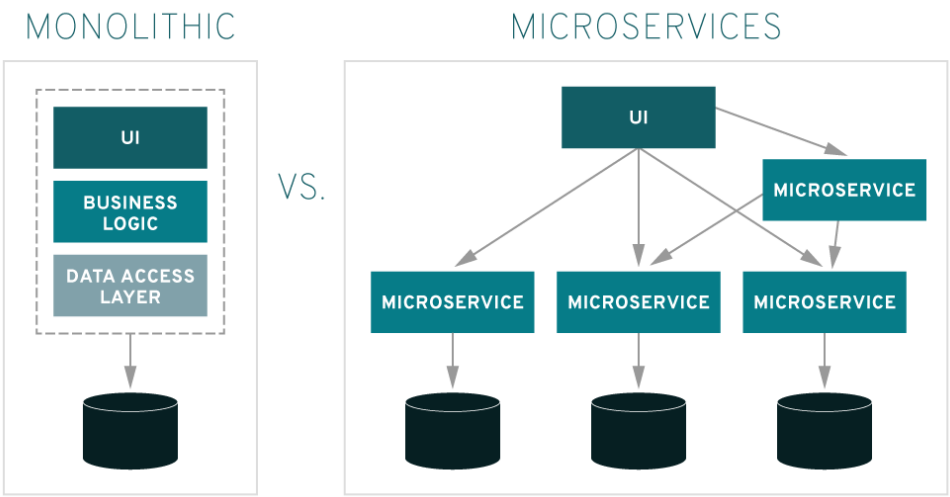


Figure B.1: Monolithic versus microservices(Source RedHat)

the changes inside a microservice must be backward compatible to not affect the upstream consumers.

Domain-Driven design is used in microservice architecture to define service boundaries. This helps to "design software systems based on the underlying model of the business domain" (Ramalingam 2020). This approach helps building modular systems that can change and evolve independently.

Domain and sub-domain are related to the problem space, that is, the business problem. A Domain represents the business concept or what an organization does and can be divided into several subdomains. Each subdomain may have multiple bounded contexts, however it is preferred to have one bounded context per subdomain, since it is related to the solution space. This means that, a bounded context explains how the solution will be implemented originating one or more microservices.

Usually, a context map is created to help to identify and define relationships between bounded contexts and aggregates. The relationships are defined according direction, that is, one will be upstream (supplier) and other downstream (consumer). The upstream end influences the downstream context, since the downstream end depends on data of the upstream end.

Database Management in Microservices Architecture

In a microservices architecture, the approach of a database per service is usually preferred. Each service has his own persisted and private data, making it unable to be accessed by other services. There are some approaches to keeping a service's data private, when using relational databases, such as a database per service, schema per service or private tables per service. Although this two approaches uses the same database, the boundaries are clear to enforce the modularity in the system (Richardson 2019b).

However, using a database per service has some disadvantages as the implementation of business transactions that involve multiple services or implementing queries that need to join multiple databases. To help resolve this constraints when implementing transactions and queries that spans multiple services, there are several patterns, like Saga pattern or Command Query Responsibility Segregation (CQRS).

Saga design pattern is a way to manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of local transactions that updates each service and publishes a message/event to trigger the next transaction. If a step fails, this pattern performs compensatory transactions that go against previous transactions. Each local transaction represents the atomic work performed by a service.

To implement saga pattern there are two common approaches: choreography, where each local transaction publishes domain events; orchestration, where exists an orchestrator that tells services what local transactions to execute. "Each approach has its own set of challenges and technologies to coordinate the workflow" (Microsoft 2020).

CQRS enables an application to work with distinct models: one model to write the data, that is altered by commands; one or more models used to read data from. Commands must be based in tasks or events and published in a asynchronous processing queue. Reads must never modify the domain and returns a data transfer object representing the domain entity (Richardson 2019a).

This pattern supports multiple denormalized views, improves the separation of concerns and it is often used in an event sourced architecture. However, it increases complexity and could potentiate code duplications.

Distributed Control Styles

In a microservice architecture, services need to work together to reach a common goal. In an higher level, there are two approaches to getting microservices to work and collaborate: orchestration or choreography.

Orchestration
Service orchestration represents a single centralized executable business process, that is, the orchestrator, which coordinates the interaction among different services. The orchestrator is responsible for invoking and combining the services, which includes the management of transactions between individual services. This follows a paradigm of type request/response.

The orchestrator specifies an executable process that involves message exchanges with other systems, such that the message exchange sequences are controlled by the orchestration design.

This approach is easy to maintain and manage, however it creates dependencies due to highly coupled services. Since the orchestrator has the sole responsibility, if it goes down, the processing stops and it fails.

Choreography
Service choreography is a global description of the participating services, which is defined by exchange of messages, rules of interaction and agreements between two or more endpoints. Choreography employs a decentralized approach for service composition, describing the interactions between multiple services.

In choreography, a service broadcast an event and, those interested, consume it and perform actions on it. Each service knows what to do and how to react and does not need instructions, it works as an orchestrator. Choreography differs from an orchestration with respect to where the logic that controls the interactions between the services involved should reside. This approach is also known as event-driven architecture or reactive architecture.

In choreography, exists a faster processing, as there are no central controller, there are no single point failure and it is easy to add and update services.

Communication between Services

In a microservices-based application, services must interact using an inter-process communication protocols. These can be divided into two types: synchronous and asynchronous.

In synchronous communication, the client sends a request and waits for a response from the service. HTTP is the most used protocol in this type of communication.

In asynchronous communication, the message sender usually doesn't wait for a response. In this situation, lightweight asynchronous messaging protocol is used, like AMQP. Asynchronous communication make microservices more robust since the sender does not have to block while waiting for receiver to respond. The components included in this pattern, can be distinguished as:

- Sender / Publisher - packages and sends messages or sends events;
- Receiver / Subscriber - listening component that processes the content sent by a publisher in a message or subscribe events that want to communicate with.

Message brokers or event-bus interfaces are used to propagate messages and events, as RabbitMQ, Apache Kafka or Azure Service Bus.

Messages and events are not the same, as an event is a lightweight notification that indicates that something happened. Messages contains raw data itself, not just a reference to that data (Microsoft 2021b).

Layered Architecture

The layered architecture pattern is one of the most used to develop software, ease of development and reflects the division of the software into layers. "Each layer represents a grouping of modules that offers a cohesive set of services" (P. Clements et al. 2011), allowing a clear separation of concerns. A system is organized into horizontal layers, each one performing a different role and responsibility in the application.

This pattern "focuses on the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other" (Microsoft Corporation 2009) and makes easier "to create a design that supports the reusability of components" (Microsoft Corporation 2009).

This pattern does not specify how many layers and what types of layers to include, however there are four defined standard layers: presentation, business, persistence, and database (Richards 2015), as shown in figure B.2. Sometimes, "the business layer and persistence layer are combined into a single business layer, particularly when the persistence" (Richards 2015).

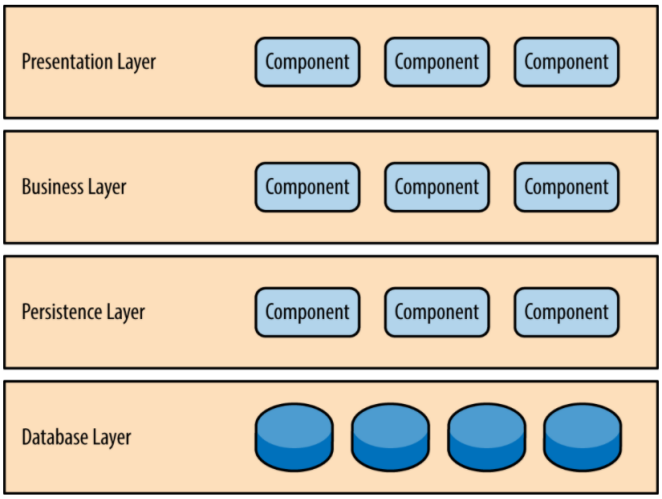


Figure B.2: Layered architecture pattern (Richards 2015)

According to figure B.2, a request coming from the presentation layer that needs to access the data in database, must pass trough the business layer, persistence layer and reach the database layer. A layer can not be bypassed, unless it is open. Layers can be opened or closed, depending on the case.

In addition, the layered architecture pattern could lead to a monolithic application, even if the presentation and business layers were deployed into different units. Because of this, implementations of this pattern are difficult to scale and has a lower performance, since one request needs to pass through multiple layers.

Microkernel Architecture

Microkernel architecture pattern is a common pattern for the implementation of product-based applications, sometimes it is referred as the plug-in architecture pattern. It is composed by two components: a core system and plug-in modules. As the application logic is divided between the core system and the plug-ins, this pattern provides " extensibility, flexibility, and isolation of application features and custom processing logic" (Richards 2015), as the core system does not needs to know which plug-in modules are available. Figure B.3 represents this pattern.

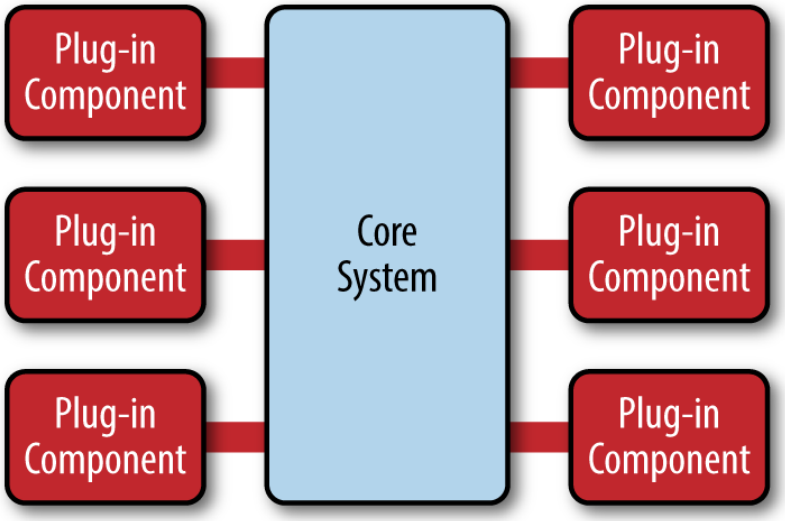


Figure B.3: Microkernel architecture pattern (Richards 2015)

- **Core system** - usually contains only the minimum functionalities for the system to be operational. Also, from a business-application perspective, the core of the system is defined as the general business logic without "custom code for special cases, special rules, or complex conditional processing" (Richards 2015).
- **Plug-in modules**- this modules are stand-alone and independent components. They usually contains "specialized processing, additional features, and custom code that is meant to enhance or extend the core system to produce additional business capabilities" (Richards 2015).

To make plug-in modules available, usually it is used a plug-in registry, that contains information about each module. There are several ways to connect the plug-in modules to the core system, as messaging, web services, among others. This pattern does not specify which type of connection should be used since it depends on the project needs.

This pattern makes it very easy to deploy plug-in modules, as they can be added to the core system at runtime. In addition, since they can be tested in isolation and can respond quickly to a environment that is constantly changing.

However, microkernel architecture pattern is complex to implement due to the "contract versioning, internal plug-in registries, plug-in granularity, and the wide choices available for plug-in connectivity" (Richards 2015). Furthermore, this pattern has a low scalability rate, since these applications are product based and are implemented as single units.

Summary of patterns

Table B.1 presents a summary of the analysis performed previously on the microservices, layered and microkernel architecture patterns. There are present six characteristics with an associated rating: ↑ (high) and ↓ (low).

Table B.1: Summary of patterns, adapted from (Richards 2015)

	Microservices	**Layered**	**Microkernel**
Overall Agility	↑	↓	↑
Deployment	↑	↓	↑
Testability	↑	↑	↑
Performance	↓	↓	↑
Scalability	↑	↓	↓
Development	↑	↑	↓

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.2 represents a Kanban board, that summarizes the design process.

Table B.2: Kanban board of iteration 1

Not Addressed	**Partially Addressed**	**Completely Addressed**	**Decisions Made**
	UC-1 to 8		Selected architecture pattern establishes the service that will support this functionality
		QA-1	Selected architecture pattern is based in a set of services, each one related to a specific subdomain. With this, each service has its responsibilities well defined
		QA-2	Selected architecture and deployment pattern allows the services to be scaled up and down when it is necessary
		QA-3	Defined logging model and destination database
		C-1	Selected technologies take into consideration the costs
		C-2 and C-3	The logical structure of the system includes this component
		AC-1	Selected architecture pattern follows this concern
	AC-2		Selected architecture pattern establishes approaches for database management
		AC-3	Selected architecture pattern establishes approaches for communication between services and the design made presents asynchronous communication between two components

B.2 Iteration 2: Support creation of KPIs and formulas

Step 6: Sketch Views and Record Design Decisions

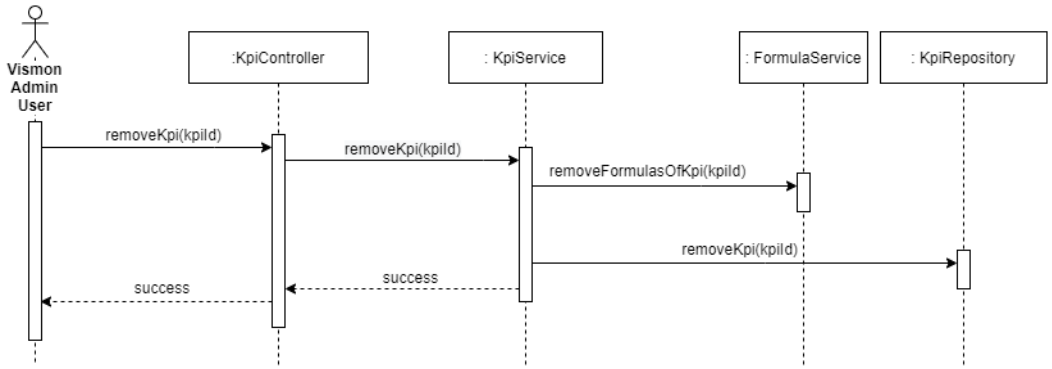


Figure B.4: Sequence diagram for remove a KPI

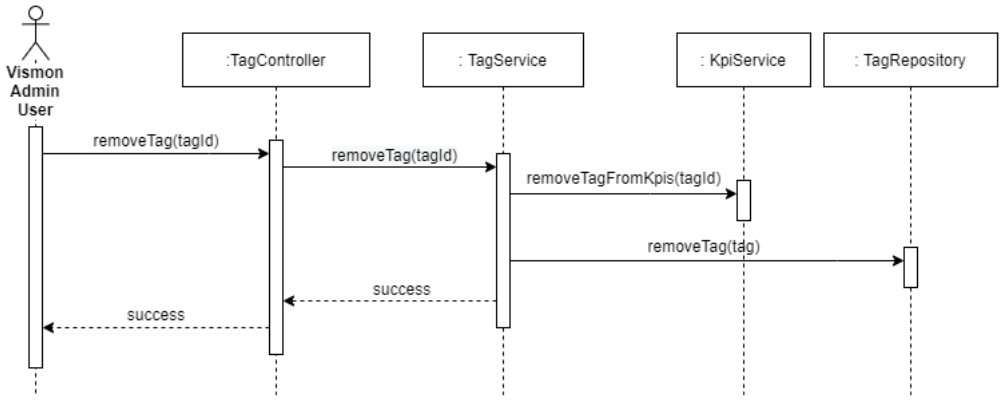


Figure B.5: Sequence diagram for removing a tag

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.3 represents a Kanban board, that summarizes the design process.

Table B.3: Kanban board of iteration 2

Not Addressed	Partially Addressed	Completely Addressed	Decisions Made
		UC-1 to 6	The designed processes support these use cases
	UC-7, 8		No relevant decisions made
	AC-2		The designed processes specifies the database for the KPIFormulaInventory

B.3 Iteration 3: Support the calculation of KPIs

Step 6: Sketch Views and Record Design Decisions

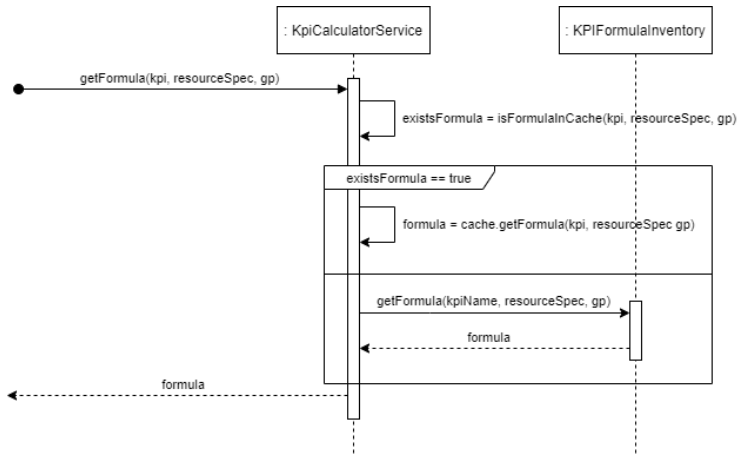


Figure B.6: Sequence diagram for getting a formula

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Table B.4 represents a Kanban board, that summarizes the design process.

Table B.4: Kanban board of iteration 3

Not Addressed	Partially Addressed	Completely Addressed	Decisions Made
		UC-7, 8	The designed processes completely addresses these use cases
		AC-2	The designed processes specifies the database for the PerformanceDataAggregator and KPICalculator

Appendix C

Experimentation and Evaluation

C.1 Processing times

Table C.1: NEs involved in the tests

Network Element	**N° of records in raw data**
SubNetwork=CountryNM,NodeFunction=3,UtranCell=CLL1227	23061
SubNetwork=CountryNM,NodeFunction=3,UtranCell=CLL1229	23060
SubNetwork=CountryNN,NodeFunction=1,UtranCell=CLL1234	23060
SubNetwork=CountryNN,NodeFunction=1,UtranCell=CLL1235	23060
SubNetwork=CountryNN,NodeFunction=1,UtranCell=CLL1236	23060
SubNetwork=CountryNN,NodeFunction=2,UtranCell=CLL1230	23060
SubNetwork=CountryNN,NodeFunction=2,UtranCell=CLL1231	23060
SubNetwork=CountryNN,NodeFunction=2,UtranCell=CLL1232	23060

Table C.2: Test definitions to register calculation times

Test Number	**Aggregation Execution**	**Aggregation Type**	**Number of Records**
1	Pre-defined	UtranCell-Hour	46120
2	On-demand	UtranCell-Hour	46120
3	Pre-defined	NodeFunction-Hour	17295
4	On-demand	NodeFunction-Hour	17295
5	Pre-defined	SubNetwork-Hour	11530
6	On-demand	SubNetwork-Hour	11530

Table C.3: Collected times for UtranCell-Hour aggregation

Execution Number	UtranCell-Hour			
1	on-demand	4	pre-defined	3
2	on-demand	5	pre-defined	4
3	on-demand	4	pre-defined	4
4	on-demand	4	pre-defined	4
5	on-demand	4	pre-defined	4

Table C.3: Collected times for UtranCell-Hour aggregation

6	on-demand	4	pre-defined	4
7	on-demand	4	pre-defined	4
8	on-demand	4	pre-defined	4
9	on-demand	4	pre-defined	4
10	on-demand	4	pre-defined	4
11	on-demand	4	pre-defined	4
12	on-demand	5	pre-defined	4
13	on-demand	7	pre-defined	4
14	on-demand	6	pre-defined	4
15	on-demand	6	pre-defined	4
16	on-demand	6	pre-defined	4
17	on-demand	5	pre-defined	4
18	on-demand	4	pre-defined	4
19	on-demand	4	pre-defined	4
20	on-demand	4	pre-defined	4
21	on-demand	5	pre-defined	4
22	on-demand	5	pre-defined	4
23	on-demand	5	pre-defined	4
24	on-demand	4	pre-defined	4
25	on-demand	4	pre-defined	4
26	on-demand	4	pre-defined	4
27	on-demand	4	pre-defined	4
28	on-demand	4	pre-defined	4
29	on-demand	4	pre-defined	4
30	on-demand	5	pre-defined	4
31	on-demand	5	pre-defined	4
32	on-demand	5	pre-defined	4
33	on-demand	5	pre-defined	4
34	on-demand	5	pre-defined	4
35	on-demand	5	pre-defined	4
36	on-demand	4	pre-defined	4
37	on-demand	4	pre-defined	4
38	on-demand	4	pre-defined	4
39	on-demand	5	pre-defined	4
40	on-demand	5	pre-defined	4
41	on-demand	5	pre-defined	4
42	on-demand	4	pre-defined	4
43	on-demand	5	pre-defined	4
44	on-demand	5	pre-defined	4
45	on-demand	5	pre-defined	4
46	on-demand	4	pre-defined	4
47	on-demand	4	pre-defined	4
48	on-demand	5	pre-defined	4
49	on-demand	5	pre-defined	4
50	on-demand	5	pre-defined	4

Table C.4: Collected times for NodeFunction-Hour aggregation

Execution Number	NodeFunction-Hour			
1	on-demand	5	pre-defined	1
2	on-demand	4	pre-defined	1
3	on-demand	4	pre-defined	1
4	on-demand	5	pre-defined	1
5	on-demand	5	pre-defined	1
6	on-demand	5	pre-defined	1
7	on-demand	6	pre-defined	1
8	on-demand	6	pre-defined	1
9	on-demand	6	pre-defined	1
10	on-demand	6	pre-defined	1
11	on-demand	6	pre-defined	1
12	on-demand	6	pre-defined	1
13	on-demand	6	pre-defined	1
14	on-demand	5	pre-defined	1
15	on-demand	5	pre-defined	1
16	on-demand	5	pre-defined	1
17	on-demand	6	pre-defined	1
18	on-demand	6	pre-defined	1
19	on-demand	5	pre-defined	1
20	on-demand	5	pre-defined	1
21	on-demand	6	pre-defined	1
22	on-demand	6	pre-defined	1
23	on-demand	6	pre-defined	1
24	on-demand	6	pre-defined	1
25	on-demand	6	pre-defined	1
26	on-demand	6	pre-defined	1
27	on-demand	6	pre-defined	1
28	on-demand	6	pre-defined	1
29	on-demand	6	pre-defined	1
30	on-demand	6	pre-defined	1
31	on-demand	6	pre-defined	1
32	on-demand	6	pre-defined	1
33	on-demand	6	pre-defined	1
34	on-demand	7	pre-defined	1
35	on-demand	7	pre-defined	1
36	on-demand	7	pre-defined	1
37	on-demand	7	pre-defined	1
38	on-demand	7	pre-defined	1
39	on-demand	7	pre-defined	1
40	on-demand	7	pre-defined	1
41	on-demand	7	pre-defined	1
42	on-demand	7	pre-defined	1
43	on-demand	7	pre-defined	1
44	on-demand	7	pre-defined	1

Table C.4: Collected times for NodeFunction-Hour aggregation

45	on-demand	7	pre-defined	1
46	on-demand	7	pre-defined	1
47	on-demand	7	pre-defined	1
48	on-demand	7	pre-defined	1
49	on-demand	7	pre-defined	1
50	on-demand	7	pre-defined	1

Table C.5: Collected times for SubNetwork-Hour aggregation

Execution Number	SubNetwork-Hour			
1	on-demand	45	pre-defined	2
2	on-demand	46	pre-defined	1
3	on-demand	43	pre-defined	2
4	on-demand	42	pre-defined	2
5	on-demand	43	pre-defined	2
6	on-demand	42	pre-defined	2
7	on-demand	42	pre-defined	2
8	on-demand	46	pre-defined	2
9	on-demand	46	pre-defined	2
10	on-demand	41	pre-defined	2
11	on-demand	42	pre-defined	2
12	on-demand	38	pre-defined	2
13	on-demand	41	pre-defined	2
14	on-demand	43	pre-defined	2
15	on-demand	42	pre-defined	2
16	on-demand	44	pre-defined	2
17	on-demand	44	pre-defined	2
18	on-demand	43	pre-defined	2
19	on-demand	42	pre-defined	2
20	on-demand	43	pre-defined	2
21	on-demand	47	pre-defined	2
22	on-demand	35	pre-defined	2
23	on-demand	23	pre-defined	2
24	on-demand	20	pre-defined	2
25	on-demand	21	pre-defined	2
26	on-demand	25	pre-defined	2
27	on-demand	24	pre-defined	2
28	on-demand	26	pre-defined	2
29	on-demand	26	pre-defined	2
30	on-demand	26	pre-defined	2
31	on-demand	25	pre-defined	2
32	on-demand	24	pre-defined	2
33	on-demand	25	pre-defined	2
34	on-demand	25	pre-defined	2
35	on-demand	25	pre-defined	2

Table C.5: Collected times for SubNetwork-Hour aggregation

36	on-demand	25	pre-defined	2
37	on-demand	30	pre-defined	2
38	on-demand	27	pre-defined	2
39	on-demand	34	pre-defined	2
40	on-demand	28	pre-defined	2
41	on-demand	26	pre-defined	2
42	on-demand	25	pre-defined	2
43	on-demand	22	pre-defined	2
44	on-demand	23	pre-defined	2
45	on-demand	24	pre-defined	1
46	on-demand	27	pre-defined	2
47	on-demand	25	pre-defined	2
48	on-demand	25	pre-defined	2
49	on-demand	26	pre-defined	2
50	on-demand	26	pre-defined	2

C.1.1 UtranCell-Hour

To verify if the data follows a normal distribution, a Lilliefors (Kolmogorov-Smirnov) test was executed. For that, a test of hypothesis was performed: H_0 = data follows a normal distribution, H_1 = data not follows a normal distribution.

```
> pre_defined <- utranCell_hour[utranCell_hour$execution=="pre-defined", "duration"]
> on_demand <- utranCell_hour[utranCell_hour$execution=="on-demand", "duration"]
> lillie.test(on_demand-pre_defined)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  on_demand - pre_defined
D = 0.29326, p-value = 9.311e-12
```

Figure C.1: Lilliefors test for UtranCell-Hour

Observing the results in figure C.1, p-value = 9.31e-12 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, that the data does not follows a normal distribution.

C.1.2 NodeFunction-Hour

To verify if the data follows a normal distribution, a Lilliefors (Kolmogorov-Smirnov) test was executed. For that, a test of hypothesis was performed: H_0 = data follows a normal distribution, H_1 = data not follows a normal distribution.

```
> pre_defined <- nodeFunction_hour[nodeFunction_hour$ï..execution=="pre-defined", "duration"]
> on_demand <- nodeFunction_hour[nodeFunction_hour$ï..execution=="on-demand", "duration"]
> lillie.test(on_demand-pre_defined)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  on_demand - pre_defined
D = 0.24156, p-value = 8.412e-08
```

Figure C.2: Lilliefors test for NodeFunction-Hour

Observing the results in figure C.2, p-value = 9.311e-12 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, that the data does not follows a normal distribution.

C.1.3 SubNetwork-Hour

Firstly, to verify if the data follows a normal distribution, a Lilliefors (Kolmogorov-Smirnov) test was executed. For that, a test of hypothesis was performed: H_0 = data follows a normal distribution, H_1 = data not follows a normal distribution.

```
> pre_defined <- subNetwork_hour[subNetwork_hour$ï..execution=="pre-defined", "duration"]
> on_demand <- subNetwork_hour[subNetwork_hour$ï..execution=="on-demand", "duration"]
> lillie.test(on_demand-pre_defined)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  on_demand - pre_defined
D = 0.24417, p-value = 5.574e-08
```

Figure C.3: Lilliefors test for SubNetwork-Hour

Observing the results in figure C.3, p-value = 5.574e-08 < 0.05, so the H_0 is rejected. There is statistical evidence that allows us to conclude that, with a level of significance of 5%, that the data does not follows a normal distribution.

C.2 Quantitative Evaluation Framework

q	D	Q_i	Dimension	Q_j	W_{ij} (Factor Weight *j* in Dim *i*) [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*) [0,100]
94%	0,13	97,5	Functionality	100	0,500	Functional (Refering Use Cases)	8	FF01 - Manages a KPI	100
							4	FF02 - Manages a KPIs Category	100
							2	FF03 - Categorize KPIs	100
							6	FF04 - Visualize KPIs	100
							10	FF05 - Manages a formula for a KPI	100
							8	FF06 - Visualize active formulas for a KPI	100
							10	FF07 - Request the calculation of KPI by resource or time	100
							8	FF08 - Visualize calculated KPIs	100
				80	0,125	System	10	FS01 - The system configure the triggers for the aggregations	80
							10	FS02 - The system aggregates the counters	80
				100	0,375	Content Quality	4	FCQ01 - The calculated KPI include the KPI unit.	100
							6	FCQ02 - The KPI value is presented with two decimal cases.	100
							10	FCQ03 - The calculated KPI include the temporal integrity.	100
							6	FCQ04 - The calculated KPI include the granularity period	100
							10	FCQ05 - The calculated KPI includes the timestamp and network resource	100
							8	FCQ06 - The system limits the calculated KPI values	100
		87,5	Quality	50	0,125	Scalability	4	QS01 - The system can scale-up and scale-down when it is necessary	50
				50	0,125	Manegeability	2	QME01 - The system must produce logs	50
				100	0,250	Maintainability	8	QM01 - Architectural and software patterns should be properly applied	100
							8	QM02 - Documentation and code must be syncrhonized	100
				100	0,250	Technical documentation	8	QT01 - Analysis and design documentation should be produced	100
							8	QT02 - Solution must include tests	100
				100	0,250	Reusability	10	QR01 - The system can be used in distinct operators without chaching any code	100
							10	QR02 - The system is vendor and technology agnostic.	100

Figure C.4: Quantitative evaluation framework- overview

Figure C.5: Quantitative evaluation framework- dimensions

Dimension	Q_j	W_{ij} (Factor Weight j in Dim i) [0,1]	Factor	rw_{jk} (requirement weight k in Factor j) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment k) [0,100]
Functionality	100	0,500	Functional (Refering Use Cases)	8	FF01 - Manages a KPI	100
				4	FF02 - Manages a KPIs Category	100
				2	FF03 - Categorize KPIs	100
				6	FF04 - Visualize KPIs	100
				10	FF05 - Manages a formula for a KPI	100
				8	FF06 - Visualize active formulas for a KPI	100
				10	FF07 - Request the calculation of KPI by resource or time	100
				8	FF08 - Visualize calculated KPIs	100
	80	0,125	System	10	FS01 - The system configure the triggers for the aggregations	80
				10	FS02 - The system aggregates the counters	80
	100	0,375	Content Quality	4	FCQ01 - The calculated KPI include the KPI unit.	100
				6	FCQ02 - The KPI value is presented with two decimal cases.	100
				10	FCQ03 - The calculated KPI include the temporal integrity.	100
				6	FCQ04 - The calculated KPI include the granularity period	100
				10	FCQ05 - The calculated KPI includes the timestamp and network resource	100
				8	FCQ06 - The system limits the calculated KPI values	100

Figure C.6: Quantitative evaluation framework - functionality dimension overview

Quality	50	0,125	Scalability	4	QS01 - The system can scale-up and scale-down when it is necessary	50
	50	0,125	Manegeability	2	QME01 - The system must produce logs	50
	100	0,250	Maintainability	8	QM01 - Architectural and software patterns should be properly applied	100
				8	QM02 - Documentation and code must be syncrhonized	100
	100	0,250	Technical documentation	8	QT01 - Analysis and design documentation should be produced	100
				8	QT02 - Solution must include tests	100
	100	0,250	Reusability	10	QR01 - The system can be used in distinct operators without chaching any code	100
				10	QR02 - The system is vendor and technology agnostic.	100

Figure C.7: Quantitative evaluation framework - quality dimension overview

Dimension	Functionality
Factor	Functional

		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FF01 - Manages a KPI	The Vismon admin user adds or removes a KPI	No access to functionality	Partial access to the functionality	Full access to the functionality
FF02 - Manages a KPIs Category	The Vismon admin user adds or removes a category.	No access to functionality	Partial access to the functionality	Full access to the functionality
FF03 - Categorize KPIs	The Vismon admin user adds or remove a category from a KPI	No access to functionality	Partial access to the functionality	Full access to the functionality
FF04 - Visualize KPIs	The Vismon admin user visualize the created KPIs	No access to functionality	Partial access to the functionality	Full access to the functionality
FF05 - Manages a formula for a KPI	The Vismon admin user visualize the created formulas of a KPIs. The system returns 0 or 1 formula for a KPI, per resource and granularity period.	No access to functionality	Partial access to the functionality	Full access to the functionality
FF06 - Visualize active formulas for a KPI	Content Specialist can edit details of a challenge and add extra informations an	No access to functionality	Partial access to the functionality	Full access to the functionality
FF07 - Request the calculation of KPI by resource or time	Vismon users can request the calculation of KPIs by resource and granularity period, with the possibility to apply filters and group the results.	No access to functionality	Partial access to the functionality	Full access to the functionality
FF08 - Visualize calculated KPIs	Vismon users can visualize calculated KPIs.	No access to functionality	Partial access to the functionality	Full access to the functionality

Figure C.8: Quantitative evaluation framework - use cases

Dimension	Functionality
Factor	System

		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**80**	**100**
FS01 - The system configure the triggers for the aggregations	The system is able to configure the aggregations to be executed that has a defined scheduler.	The system does not configure the executions.	The system configures with errors executions.	The system configures the executions.
FS02 - The system aggregates the counters	The system is able to aggregate correctly and without errors PM data with the defined nodal and temporal aggregation	The system does not aggregate PM data	The system aggregates PM data with some errors and incorrections.	The system aggregates PM data correctly and without errors

Figure C.9: Quantitative evaluation framework - content quality

Dimension	Functionality
Factor	Content Quality

		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FCQ01 - The calculated KPI include the KPI unit.	All the results related with the calculation of the KPI values must contain the KPI unit (example: %, dBm)	Contains the kpi unit	-	Does not contain the kpi unit
FCQ02 - The KPI value is presented with two decimal cases.	All the results related with the calculation of the KPI values must be rounded to two decimal cases.	No	-	Yes
FCQ03 - The calculated KPI include the temporal integrity.	All the results related with the calculation of the KPI values must be contain the temporal integrity associated with the value.	No	-	Yes
FCQ04 - The calculated KPI include the granularity period	All the results related with the calculation of the KPI values must be contain the granularity period associated (example: hour, day, week)	No	-	Yes
FCQ05 - The calculated KPI includes the timestamp and network resource	All the results related with the calculation of the KPI values must be contain the timestamp (example: 2021/02/22 10:00) and the network resource (example: resourceXPTO)	No	-	Yes
FCQ06 - The system limits the calculated KPI values	The system must have a default limit value for the data to be returned (example: 10000 records)	Does not limit the values	-	Limit the values

Figure C.10: Quantitative evaluation framework - content quality

Dimension	Quality
Factor	Scalibility, Manegeability,Maintanability, Technical documentation, Reusability

Requirement	Metric Evaluation	Wfk - Fullfilment (%)		
		0	50	100
QS01 - The system can scale-up and scale-down when it is necessary	The system is capable to scale-up when the demand for the resources is high and scale-down when the demand decreases.	Not tested	Partially tested	Fully tested
QME01 - The system must produce logs	The system logs its activity to a centralized database.	The system does not produce logs.	The system produces logs in a decentralized way.	The system produces centralized log.
QM01 - Architectural and software patterns should be properly applied	To build the system, architectural and software patterns must be properly applied, whenever is necessary	Patterns were not applied	-	Patterns were applied
QM02 - Documentation and code must be syncrhonized	The code must be aligned with the produced documentation	Documentation does not exist	Few synchronized	All synchronized
QT01 - Analysis and design documentation should be produced	The documentation must be produced for the use cases, as sequence diagrams.	There are no sequence diagrams available for the main use cases	There are some sequence diagrams available for the main use cases	The sequence diagrams are available for the main use cases
QT02 - Solution must include tests	The developed code must be covered by unit tests to ensure the quality of the procuded code.	No includes unit tests	At least 50% of the code is covered by tests	At least 80% of the code is covered by tests
QR01 - The system can be used in distinct operators without chaching any code.	The system must be operator agnostic to be utilized in distinct environments without changing any code.	The system does not support multi operators.	-	The system supports multi operators.
QR02 - The system is vendor and technology agnostic.	The system must be configurable to support several equipment vendors and technologies.	The system does not support multi vendors and operators.	-	The system supports multi vendors and operators.

Figure C.11: Quantitative evaluation framework - quality