The design and engineering of a unified data access layer : bridging data consumers and diverse data sources

Citation for published version (APA):
Safari, F. (2017). *The design and engineering of a unified data access layer : bridging data consumers and diverse data sources*. Technische Universiteit Eindhoven.

Document status and date:
Published: 28/09/2017

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 08. Jul. 2024

Technische Universiteit
Eindhoven
University of Technology

/ Department of Mathematics and Computer Science / PDEng Software Technology

The Design and Engineering of a Unified Data Access Layer:

Bridging Data Consumers and diverse Data Sources

Fariba Safari

Where innovation starts

The Design and Engineering of a Unified Data Access Layer

Bridging Data Consumers and diverse Data Sources

Fariba Safari

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

Partners	Océ	Eindhoven University of Technology
Steering Group	Tim Paffen & Jeroen Janssen & George Fletcher (Project Supervisors)	
Date	18 September 2017	
Document Status	Public	
PDEng report no.	2017/046	

The design described in this report has been carried out in accordance with TU/e Code of Scientific Conduct.

Contact Address	Eindhoven University of Technology Department of Mathematics and Computer Science MF 5.080A, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands +31402474334
Published by	Eindhoven University of Technology Stan Ackermans Institute
Printed by	Eindhoven University of Technology *UniversiteitsDrukkerij*
PDEng Report No.	2017/046
Abstract	In the research and development process at Océ, considerable amounts of data are being generated by either machines or services. This data is used by different data consumers at Océ for various purposes such as analysis and reporting. However, since the data is generated by different machines and for different purposes, the data sources are incompatible in the sense of the format and the access technology. When a data consumer requires data from different data sources, it has to write its own integration code. This leads to effort and time for each data consumer. Therefore, there is a need to have a unified Data Access Layer between data sources and data consumers to reduce the effort and time for each data consumer. In addition, for solving the incompatibility of data formats, the goal is to provide a schema and type system, in which data structures are mapped with each other. Design and implementation of the data access layer is an important goal to show the validation of the chosen technology. Design challenges are: 1) mapping the schema of the data sources to the foreign tables. 2) automatic process of creating foreign tables when the schema evolves, and 3) designing reusable foreign data wrappers. These challenges are addressed in the design. The system quality is validated based on the user feedback. The report also lists several recommendations for future improvements.
Keywords	Data access layer, data integration, foreign data wrappers, join, data source, data consumer
Preferred Reference	THE DESIGN AND ENGINEERING OF A UNIFIED DATA ACCESS LAYER: BRIDGING DATA CONSUMERS AND DIVERSE DATA SOURCES, SAI Technical Report, September 2017. (2017/046)
Partnership	This project was supported by Eindhoven University of Technology and Océ.
Disclaimer Endorsement	Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology or Océ. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or Océ, and shall not be used for advertising or product endorsement purposes.

Disclaimer
Liability

While every effort will be made to ensure that information contained within this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation, or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or services marks of their respective owners. We use these name without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright

Copyright © 2017. Eindhoven University of Technology. All rights reserved. No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the Eindhoven University of Technology and Océ.

Foreword

Océ is one of the world leading manufacturers of print systems for the professional market. These systems generate an increasing amount of different data. Within our organization there are also other sources of data, e.g. related to contracts, service activities and products that have evolved separately from each other, resulting in different systems. We see an increasing need to be able to relate the data from new and existing sources to successfully support the different Business Entities within Océ. New big data related projects and developments are on the horizon. Thus a clear need to have a platform was identified. This platform must be able to integrate and provide access to a multitude of existing and new data sources.

As part of this assignment, Fariba took good initiative and independently consulted stakeholders and colleagues. As the project progressed, she shaped the concept of the Data Access Layer. Analysis on different technical routes for this DAL were done and, in the process, she built several prototypes to evaluate these options. She built a prototype implementation to showcase the integration of different data sources. This prototype has been used by actual end-users for verification and we are very happy how the end result turned out. The work Fariba has done and the insight she has provided will become part of our Data Architecture Roadmap. As part of this roadmap we will take into considerations the future work and ideas she has provided.

As supervisors it is very rewarding to see progress and personal growth in a student and we hope we contributed in a positive way to that. We see that Fariba has shown great eagerness to learn and develop herself and we also believe she will continue to develop herself even further. She has quickly become part of our teams, both in professional and social context.

We are very happy she has accepted a job position at Océ and that she will actually start in one of our teams.

Tim Paffen and Jeroen Janssen
18 September 2017

Preface

This report summarizes the project entitled: "*The Design and Engineering of a Unified Data Access Layer: Bridging Data Consumers and diverse Data Sources*" executed by Fariba Safari, as part of her graduation assignment for the PDEng (Professional Doctorate in Engineering) program under supervision of Océ and Eindhoven University of Technology (TU/e). The goal of the project is to design and implement a Data Access Layer to reduce the effort and time for accessing data within Océ. This report confirms the successful design and engineering of the project. It also provides detailed information about the software development and the project management process.

The audiences of this report are both technical as well as non-technical readers. Readers that are interested in the domain, the problem and its challenges within Océ, can read Chapters 1,2, and 3. Readers who are interested in the technical details, such as system requirements, system architecture, and system design, are referred to Chapters 3, 5, and 6. Readers that are interested to know the rationality behind the chosen technologies are invited to read Chapter 4. Readers mainly interested in the results or the future continuation of the project can read Chapter 7 and 8. Readers interested to know about the project plan and retrospective are referred to Chapter 9 and 10.

Fariba Safari
18 September 2017

Acknowledgments

I would like to express my sincere gratitude to my supervisors from Océ, Tim Paffen and Jeroen Janssen for providing me the opportunity to carry out the project in Océ. Your critical thinking and knowledge encouraged me to explore different aspects and approaches on the project. I enjoyed working with you and I am looking forward to working with you in the future. In addition, I would like to thank helpful people at Océ who were interested in my project and helped me in gathering required information: Stefan Sanders, Rob Kersemakers, Johan Hoogendoorn, Peter Kruizinga, Gé Kessels, Jacques Bergmans, Jan Saris, Jos Jans, Pieter Verduin, Jeroen Dopp, Peter Cornelissen, William Howard, Hristina Moneva, Luc de Smet, Rob Gaal and Jos Derix.

I owe the success of this project also to my university supervisor, George Fletcher. Thank you for the continuous support, motivation and feedback. In addition, my gratitude goes to everybody involved in the Software Technology PDEng program from TU/e, especially Ad Aerts, Yanja Dajsuren, and Desiree van Oorschot for giving me the opportunity to be part of the PDEng program and their support and guidance during the past two years. I also want to thank the coaches from TU/e involved with my project. In particular, Harold Weffers for good conversations about project management and risk assessment. I would also like to thank my colleagues from the PDEng program for the good moments we shared together.

Finally, my deep and sincere gratitude to my family for their love, help, and support. I am grateful to my sisters for always being there for me as friends. I am forever indebted to my parents for encouraging and supporting me to take new opportunities and experiences in my life.

Fariba Safari
18 September 2017

Executive Summary

Océ, as one of the leading manufacturers in printing industry, deals with various machines and services that produce data. In addition, it deals with various business systems that aim to gather insight from the data for the purpose of analysis and reporting. To improve the data access process for accessing and combining the data sources, Océ decided to take advantage of the data integration technologies. This project was established to achieve this.

In the current situation in Océ, the data access process is not straightforward. Each business system needs to take care of the data access and handle the incompatibilities of the data by itself. Since data is gathered from different machines and services, the chances of incompatibility of data are high. This makes the process of data access complicated when data from multiple sources are needed. By bringing data together, users gain comprehensive insights for analysis. For example, for predicting a malfunction in a machine, combining data of the service visits and the functional logging is helpful. Hence, the goal of this project is to develop a Data Access Layer (DAL) for combining diverse data sources with different formats of data. Having a Data Access Layer strengthens the business from its very core by incorporating the information from multiple sources. It helps to create a comprehensive view from the data of the customers and the products.

The main delivery of this project is a prototype DAL that provides the data access and integration. The Foreign Data Wrappers technology is the choice for implementation based on the results of a performance test and requirement analysis. The system design is modeled in a way to cover the challenging aspects of the design, such as mapping the schema of the data sources to the foreign tables, as well as automatic processes of creating foreign tables. The system is extensible for new data sources. Multiple data source formats and data types are supported. The system is tested by users, using real data, in their working environment. The results are verified based on a user feedback survey showing that the use of a DAL is beneficial for data analysis and reporting within Océ.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the project and its context including the PDEng program, Océ, and the stakeholders. The outline section gives a brief overview of what is discussed in the following chapters.

1.1 PDEng Program Introduction

This project was conducted as part of the Professional Doctorate in Engineering (PDEng) program. The PDEng degree program in Software Technology is provided by the Department of Mathematics and Computer Science of Eindhoven University of Technology in the context of the 3TU School for Technological Design, Stan Ackermans Institute. The program is a two-year, third-cycle engineering degree program during which the trainees focus on strengthening their technical and non-technical competencies related to the design and development of software for software-intensive systems in an industrial setting.

1.2 Project Context

This project is initiated by Océ, a Canon company, within the Research and Development (R&D) department. Océ is a top 10 R&D investor in the Netherlands and its main products are printing technologies: large format, continuous feed, cut-sheet printing, and sheet-fed printing. In addition, they offer work flow software solutions and business services. Océ's key growth areas are graphic arts, business services, and industrial printing. Graphic arts is the printing of text and graphics for applications such as newspapers, books, magazines, banners and signage. Océ products offer customers a number of advantages, such as shorter runs, individualization, customization, and high-quality specialized products. Arizona is a graphic arts production machine (See Figure 1.1).

Figure 1.1: Océ Arizona 6170 XTS printer: Signature for up to 200 signs per day [1]

An example for the cut sheet printing is the VarioPrint i300, also known as the Niagara (Figure 1.2).

Figure 1.2: Océ VarioPrint I300: Cutsheet High Production System [1]

Niagara produces a large amount (gigabytes per day) of log data useful for analysis. This log data flows from the machines in the field towards Océ. This data is the input for a number of data analysis tools and eventually results in valuable information for both Océ and their customers. However, since the data is generated by different machines and for different purposes, the data sources are incompatible in the sense of the format and the access technologies. This brings up difficulties in accessing data from both data sources and combining them. The goal is to design and implement a data integration and access layer to hide the technical details of data sources and provide an easy way to access data by handling the incompatibilities. The goal aligns with the vision of the organization to provide data to various stakeholders in an easy and transparent manner.

1.3 Project Stakeholders

This section introduces the involved stakeholders in the project. Océ is the owner and initiator of the project. This makes it the most important stakeholder in the project. Océ stakeholders have several interests as their key drivers for the project (See Table 1.1).

Table 1.1: Océ stakeholders

Role	Interests
Project owners	• Having a Data Access Layer (DAL) to provide easy, fast, and transparent access to different data sources. • Having DAL as the main data access platform within Océ.
System users	• Access the data without the need to know about the technical aspects of data sources such as the storage technology, the location, the access mechanism, and the API. • Access the real-time data by considering the performance. • Access the data by writing queries in a language that they are familiar with .
Data providers	• Maintain the ownership of data and provide data to the authenticated and authorized users

The Eindhoven University of Technology (TU/e) is responsible for the educational aspect of the project and fulfilling the requirements. That means certain standards need to be met. It is concerned with the design process, project management, and implementation. Table 1.2 lists the stakeholders, their role and interests.

Table 1.2: TU/e stakeholders

Role	Interests
TU/e supervisor	• The design and documentation meet the standard of a PDEng project.
TU/e PDEng trainee	• Fulfill the responsibility for the design and implementation of the DAL.

1.4 Outline

Chapter 2 explains the domain of the project. Problem analysis including the requirements, use cases are specified in Chapter 3. They provide a basis for designing the system and choosing the suitable technology. Solution Analysis chapter (Chapter 4) explains the decisions taken for choosing the solution technology. Moving on, Chapter 5,6 presents the architecture and design of the system, respectively. Chapter 7 explains the validation and verification of the system. The results and future work for the project are presented in Chapter 8. This chapter also gives a conclusion about the project. The last two chapters of this report reflect on the management part of the project. Chapter 9 addresses the management process during the nine months of the project as well as techniques used to manage it. In Chapter 10, the author gives her retrospective and reflection on the project.

Chapter 2

Domain Analysis

Understanding the domain, its related elements and the dependencies between them is an important step to establish a good understanding of the project. This chapter focuses on the domain analysis. The main objective in the domain analysis is to identify the data sources, data consumers, and the data flow between them. The information is gathered from existing technical documents and interviews with stakeholders and experts. Five different layers are explained.

2.1 Domain Model

In the current situation, data flows through five layers. These layers and the data flow between them are shown in the Domain Model in Figure 2.1. This model contains blocks that describe the composition of the system by describing users as actors and the other participating entities as System Modeling Language (SysML) blocks. The dashed lines with "flow" stereotype show the information flow between the different layers and blocks. In the following section, layers and their contained blocks are described. These layers are:

- **Initial data providers**: The machines and the services that produce data.
- **Data gathering layer**: The process of collecting data from the initial data providers.
- **Data sources**: Any source of data, for example, an SQL database or a CSV file.
- **Data consumers**: Any application that retrieves and manipulates data.
- **Users**: People who work with the data consumers and intend to analyze data.

2.1.1 Initial data providers

Initial data providers are the machines and the services that produce diagnostic and usage data. These providers produce data in both automatic way, such as customer devices and services, as well manual way, such as operational data providers. Below the data providers are explained.

1. **Customer devices**: Océ printers which provide diagnostic data and Canon printers which provide the usage data for analysis and customer reporting.
2. **Customer services**: The following services which are provided to the customer devices.

- Managed Document Services (MDS) cloud and Site Audit System: Systems from which total fleet information is retrieved.
- Universal Gateway(UGW): An proprietary interface for retrieving meter reads from Canon devices.
- UniFlow: A software package that provides accounting information.

3. **Operational data providers**: The Canon back office systems, which are manually supplied by an actor called operational data provider.

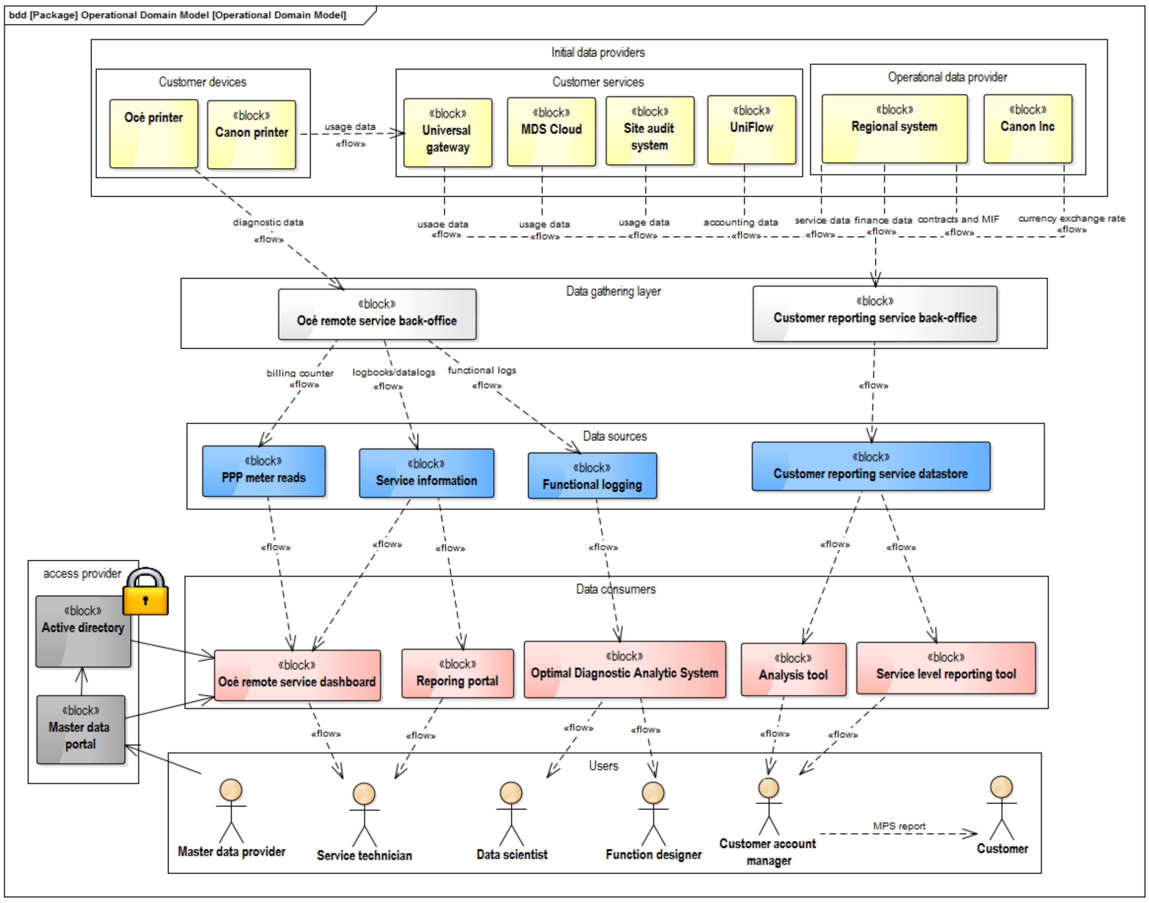


Figure 2.1: Domain Model

2.1.2 Data gathering layer

Data from different machines are generated and in the data gathering layer an Extract, Transform, and Load (ETL) technique is used to gather this data. Océ Remote Service (ORS) and Customer Reporting Service (CRS) back office gather data from the initial data providers by implementing ETL processes. CRS system connects to Canon Back office in order to collect data including contracts and service information. Figure 2.2 shows that the ETL process employed to collect data into the CRS data sources.

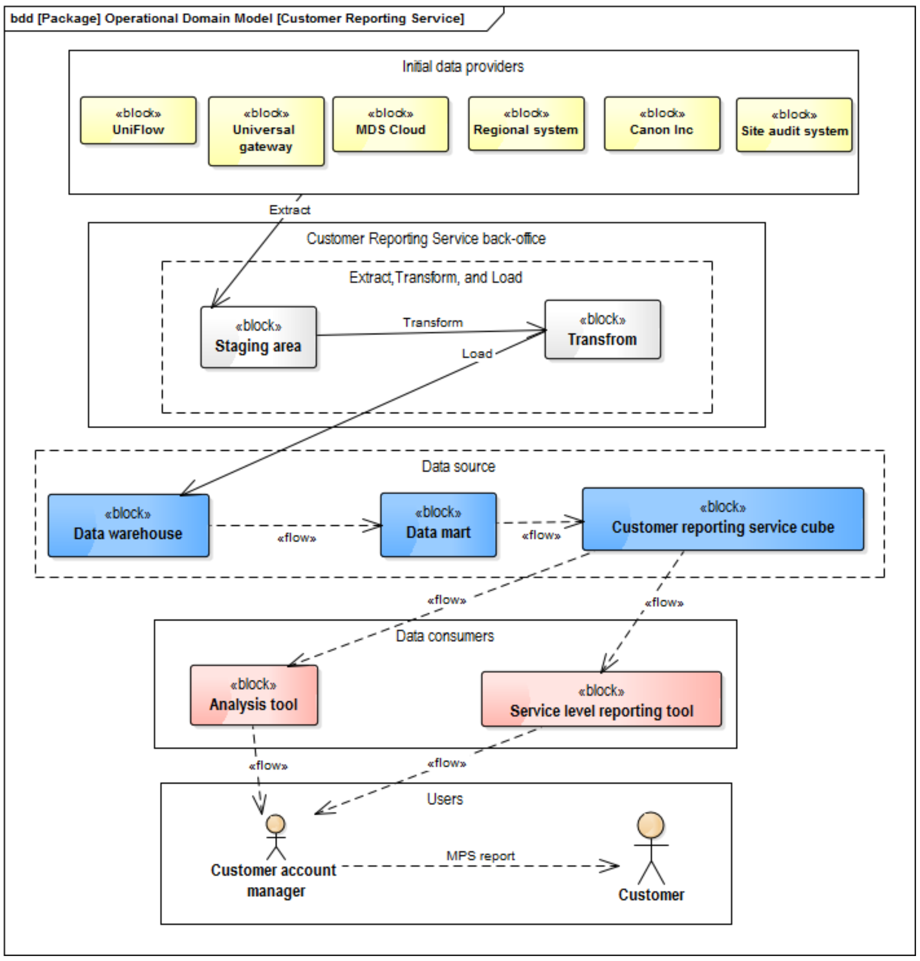


Figure 2.2: Customer Reporting Service

ORS system connects to the Océ printers in order to collect data including diagnostic data and meter reads. An ETL process in employed to store this data in meter reads and service information data sources.

2.1.3 Data sources

The term data source is used to refer to any source of data, for example, an SQL database, a cube, an HDF5 store, or a web service. Current data sources are: PPP meter reads, service information, functional logging, and CRS data store. In some cases, a data source is a collection of passive files (for example: functional logging) and in other cases it is a data source including software to access the data, such as a database server with a web service (for example: meter reads and service information). Here, each of the data sources is explained.

PPP meter read

It is a structured database storing the meter reads from Océ printers. Figure 2.3 depicts this data source. Accessing the data is through an internal web service. A client can retrieve the meter reads for a specific machine using the web service. Clients can be either a user or a system.

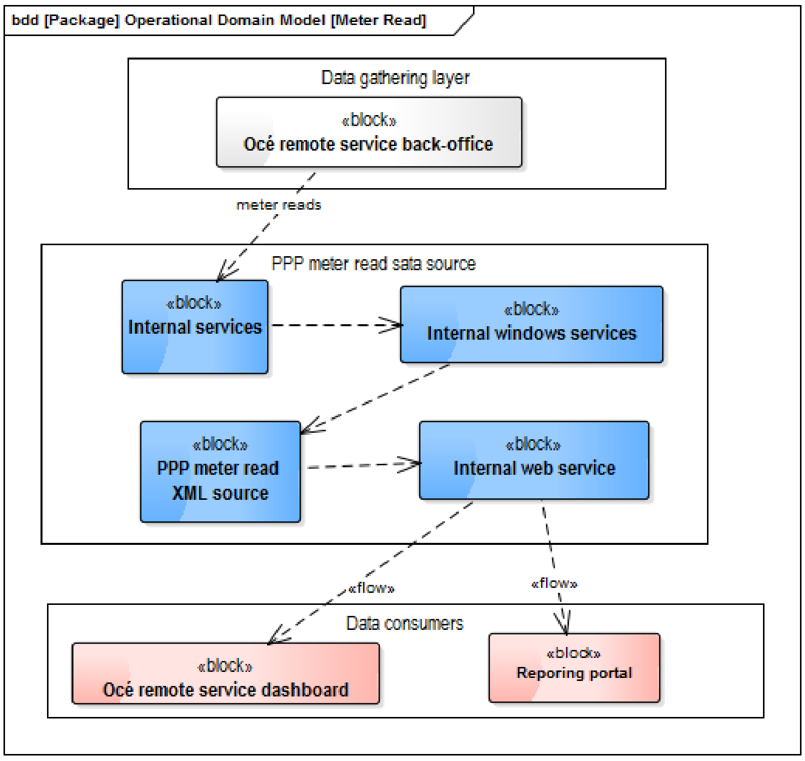

bdd [Package] Operational Domain Model [Meter Read]
Data gathering layer
«block» Océ remote service back-office
meter reads
PPP meter read sata source
«block» Internal services
«block» Internal windows services
«block» PPP meter read XML source
«block» Internal web service
«flow»
«flow»
Data consumers
«block» Océ remote service dashboard
«block» Reporing portal

Figure 2.3: PPP meter reads

Service Information

Service Information is stored as raw files, which include logbooks and data logs. Logbooks contain data from the service technicians. The contents of these logbooks are the same in 80-90% of the time (see Figure 2.4). Followings are the two different technical formats for logbooks from which the STAR logbooks have the historic format.

1. **STAR logbooks**: The logbooks are stored as text files in the Archive database. They can be downloaded by using one of the STAR modules named STAR Archive as *Microsoft Access* format.

2. **Product logbooks**: A new generation of *Extensible Markup language(XML)* logbooks for new products. Similar to STAR logbooks , there is a download tool in order to access product logbooks.

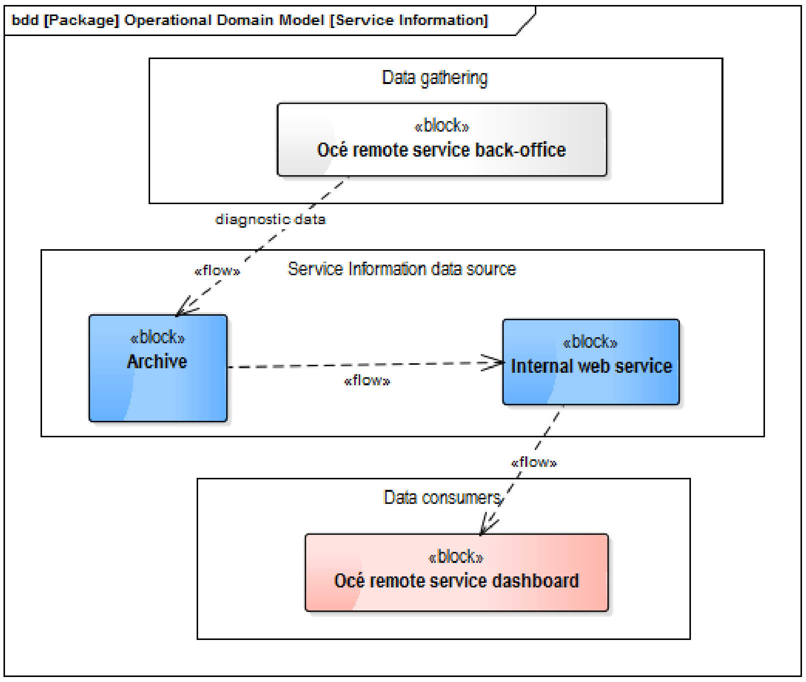

bdd [Package] Operational Domain Model [Service Information]
Data gathering
«block»
Océ remote service back-office
diagnostic data
«flow»
Service Information data source
«block»
Archive
«flow»
«block»
Internal web service
«flow»
Data consumers
«block»
Océ remote service dashboard

Figure 2.4: Service information

Functional Logging

It is a collection of large structured data from many printers, situated all over the world. Functional logging is stored in a Comma-separated-value format (CSV files) initially. For increasing the performance, the files are converted to the Hierarchical Data Format (HDF5). HDF5 is a container of data with hierarchical ordered contents. The functional logging flow is shown in Figure 2.5.

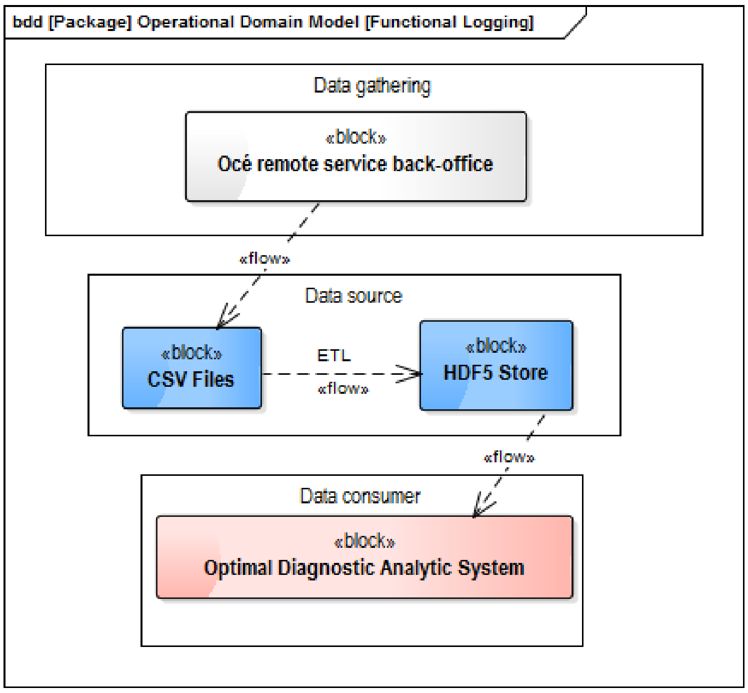

bdd [Package] Operational Domain Model [Functional Logging]
Data gathering
«block»
Océ remote service back-office
«flow»
Data source
«block»
CSV Files
ETL
«flow»
«block»
HDF5 Store
«flow»
Data consumer
«block»
Optimal Diagnostic Analytic System

Figure 2.5: Functional Logging

CRS data store

The CRS data store includes the service level reporting data in the SQL multidimensional cubes shown as *Customer reporting service cube* in Figure 2.2. This data store is available to different reporting systems: Service Level Reporting and Analysis tools.

2.1.4 Data Consumers

The term data consumer refers to any application that retrieves and manipulates data. For example, CRS is a data consumer which itself is a reporting service. Data consumers use data from the existing data sources in order to achieve their specific goals, such as analyzing data and reporting to the customers.

- Océ Remote Service (ORS) dashboard: a back-office system, which connects to the products in order to collect data and provide remote assistance to the customer. The dashboard provides access to this collected data.
- Reporting Portal: a portal for reporting from the Service Information data source.
- Service Level Reporting (SLR) and Analysis Tool (AT): the Excel workbooks that can be used for analyzing data and generating reports for customers (See Figure 2.2).
- Optimal Diagnostic Analysis System (ODAS): a Jupyter Notebook environment for users to perform data analysis.

2.1.5 Users

Users are master data providers, service technicians, data scientists, function designers, customer account managers, and customers. The users work with data consumers for their specific tasks. For example, a customer account manager uses SLR to generate reports for customers.

- **Master data providers**: People who are responsible for configuration of the master data.
- **Service technicians**: People whose aim it is to troubleshoot the printer problems and issues, on-site and remotely.
- **Function designers**: The designers who are in control of the design of a printer module.
- **Data scientists**: People who are responsible for analyzing and interpreting data for the purpose of improving products and processes.
- **Customer Account Managers**: Users whose aim it is to build reports for the customers.
- **Customer**: Organizations/people who consume a service.

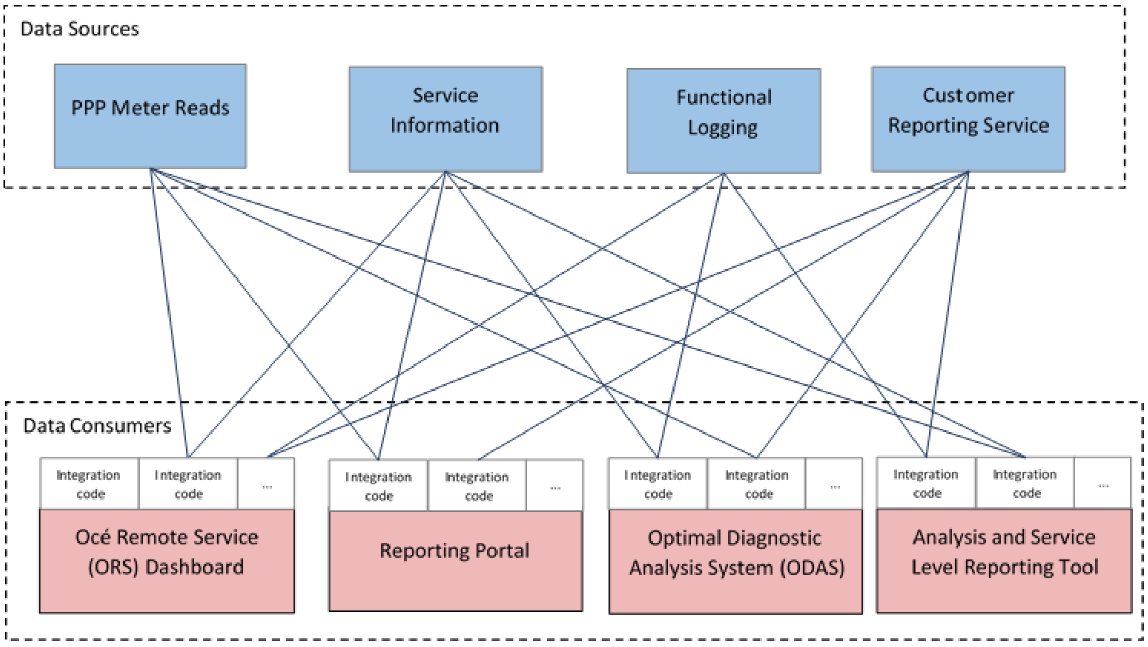
Chapter 3

Problem Analysis

After analysis of the domain, the problems is analyzed. The problem is discussed by explaining the existing situation difficulties and the characteristics of the desired situation. A set of requirements, functional and non-functional, are extracted and formulated to be satisfied for this project. In addition, the use cases and their key concerns are represented. The problem definition and requirements formulation process are conducted by discussions with the stakeholders and thorough analysis of the problem.

3.1 Problem Statement

In the existing situation, each system has access to a group of specific data sources and there is no transparent access between these different data sources and data consumers. The current situation has several disadvantages as shown in Figure 3.1.



Data Sources
PPP Meter Reads
Service Information
Functional Logging
Customer Reporting Service
Data Consumers
Integration code
Integration code
...
Océ Remote Service (ORS) Dashboard
Reporting Portal
Optimal Diagnostic Analysis System (ODAS)
Analysis and Service Level Reporting Tool

Figure 3.1: Data sources and data consumers in the current situation

In order to access the data, each data consumer needs to know: where the data sources are stored physically; where the database servers run; what the required application programming interface (API) is; and which storage technologies to use. In addition, the data consumer needs to take care of the process of joining data (based on a common data elements such as z1.serialNumber=z2.printerId), when data from multiple data sources are needed. In this situation, the incompatibility of data sources increases the difficulties. Thus, one action is to identify the internal structure of the data.

In order to overcome the mentioned disadvantages, a solution should be devised that abstracts the data sources to the data consumers. The focus of the solution should be on integrating data sources in a way that the technical details are hidden, and the data consumers can work with a unified API for accessing data (See Figure 3.2). In the next chapter, the requirements and the approaches for solving the problem is identified.

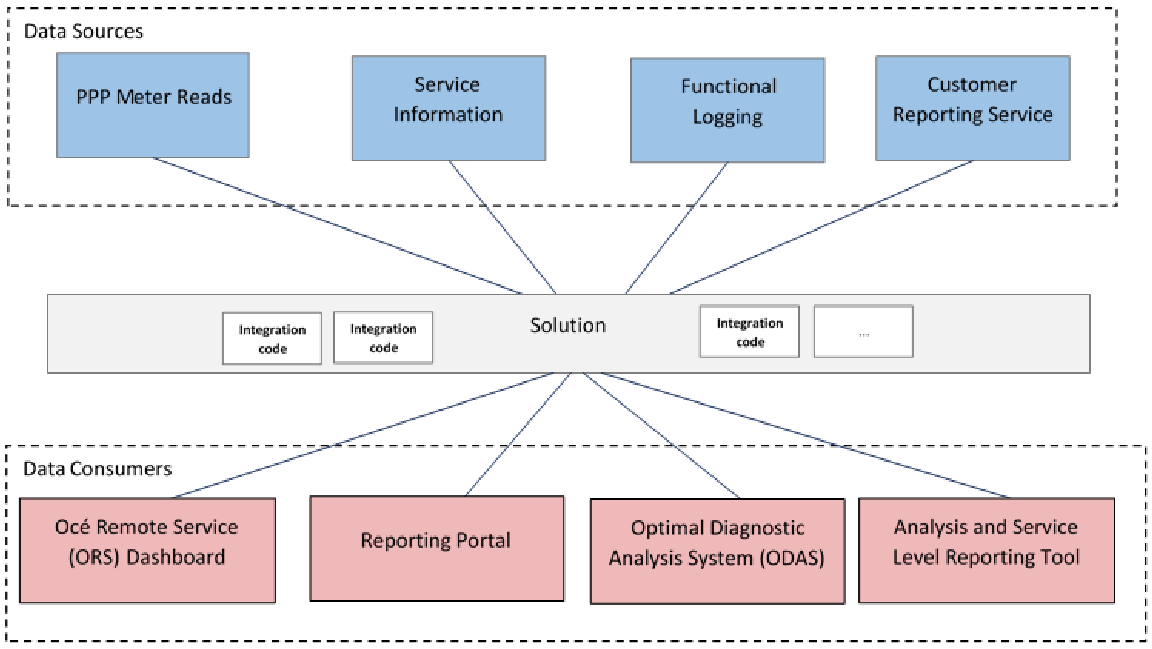


Figure 3.2: Data sources and data consumers in the desired situation

3.2 Use cases and key concerns

The main use cases are identified in the process of refining the requirements. Table 3.1 lists the identified use cases and their main key concerns. Use cases are categorized based on the common actors and their key concerns into three groups. The identified actors are: Customer account managers, Service technicians, Function designers, and Data scientists. Each of these actors are explained in the Problem Analysis chapter.

Table 3.1: Main use cases and key concerns

Title	Actors	Key concerns
Combine the Functional Logging and the Service Information in ODAS.	Function Designers Data Scientists	• Access to data sources through a compatible Python interface. • Write Blaze Queries to access data. • Perform filtering on the data sources to get only the interesting parts of the data: data of specific period of time, data of specific machines. • Perform column filtering on the data sources to get only the specified columns. • Access to combination of the two different data sources based on common columns: machine Id. • Get data in the format which is compatible with ODAS data formats: Data Frames, Lists, and Series, as well as dictionaries.
Include CRS data in Océ Remote Service (ORS) dashboard	Service Technicians	• Access to the data sources through compatible SQL interface.
Include the Océ Meter Reads in a CRS Report to extend the Canon Meter Reads	Customer Account Manager	• Access to the data sources through compatible SQL interface. • Get data in the format which is compatible with CRS data format: SQL Table

3.3 Functional Requirements Specification

This section lists the functional requirements identified after communication with the stakeholders. The requirements are categorized based on their priority level: *Must have* requirements are critical to the success of the project; *Should have* requirements are important but not necessary for the scope of the project; *Nice to have* requirements are desirable but not necessary. Table 3.2 summarizes the functional requirements.

Table 3.2: Functional Requirements Specification

Num	**Requirements**	***Priority***
FR-01	Full functionality is provided to data consumers without the need to know about the technical aspects of data sources which are: the storage technologies, location, access mechanism, and API.	*Must*
FR-02	Data consumers need to use one unified API in order to access the data sources.	*Must*
FR-02-01	Standard tools such as MATLAB and R can access the data sources.	*Should*
FR-03	Data sources shall not be replicated and aggregated into a new data source.	*Must*
FR-04	All the existing data source technologies are supported.	*Must*
FR-04-01	HDF5 store technology is supported	*Must*
FR-04-02	Web services data technology is supported	*Must*
FR-04-03	SQL database technology is supported	*Must*
FR-04-04	CSV files technology is supported	*Must*
FR-04-05	MDX Cube technology is supported	*Should*
FR-05	Data consumers have to be authorized and authenticated for reading data sources.	*Should*
FR-06	Data consumers have to be authorized and authenticated for writing into data sources.	*Should*
FR-07	Data consumers are able to send multiple concurrent requests to the data sources.	*Should*
FR-07-01	Data sources that do not support concurrent access are supported.	*Nice to have*
FR-08	Data consumers have access to the latest version of data.	*Must*

Num	**Requirements**	***Priority***
FR-09	Data consumers have the ability to execute queries on data sources.	*Must*

FR-09-01	Data consumers can filter data based on conditions and record details, for example:
	`SELECT *`
	`FROM Publication`
	`WHERE Publication.country = 'Germany'`
		Must

FR-09-02	Data consumer can select specific fields of data sources, for example:
	`SELECT name`
	`FROM Publication`
		Must

FR-09-03	Data consumers are able to join data from different tables in different data sources, for example:
	`SELECT title, authorname`
	`FROM Books, Authors`
	`WHERE Books.authorid = Authors.authorid`
		Must

3.4 Non-Functional Requirement Specification

Besides the functional requirements, a number of non-functional requirements are identified from the requirements gathering process.

Table 3.3: Non-Functional Requirements Specification

Num	**Name**	**Requirements**	***Priority***
NFR-01	Performance	The system should respond reasonably fast to data requests. The solution response time does not get worse than 30% in compare with the current situation's performance.	*Should*
NFR-02	Extensibility	Data sources can be extended and new data sources can be supported.	*Should*
NFR-03	Learnability	Data consumer have no difficulties regarding writing queries. For example, ODAS users are familiar with Blaze querying language and prefer to write queries in Blaze. In addition, Learnability includes the possibility to work with the solution from different programming environments such as C#, Java, and Python.	*Must*
NFR-04	Flexibility	Data consumers are not limited to predefined queries. They can write their own queries.	*Must*
NFR-05	Scalability	The solution continues to functioning well when data source is changed in size.	*Should*

Each of these requirements were taken into consideration when designing and implementing the solution. In the Verification and Validation as well as the Conclusion Chapter, the requirements are revisited in order to evaluate the success of the solution.

Chapter 4

Solution Analysis

This chapter aims to choose the solution direction and the technology based on the result of the experiment and the evaluation criteria. Solution directions are discussed; technology options are listed; and Performance experiment is performed and finally the technology is chosen.

4.1 Solution Directions

In order to solve the data integration problem mentioned in the previous chapter, the following approaches can be considered:

4.1.1 Individual API for each data source

In this situation, for each data source there is an API or a software library to provide access from the data source to the data consumer. The consumer knows how to access the API. We call this approach *RAW access*. This is exactly the existing implementation between data sources and data consumers in Océ.

4.1.2 Unified API to different data sources

In this situation, the data consumers access the data through a unified API and the API hides the data sources details. We call this *Data Access Layer (DAL)*. A DAL offers numerous advantages:

- When either a data source or a data consumer changes, the others do not need to change.
- Data consumers do not need to know the storage format and technical details of the data sources.
- Data consumers can access a combinations of data sources without knowing the internal structure of each data source.
- There is one unified API to access all the different storage formats, and therefore data access is simple.
- Data consumers access the new data sources quickly and without much integration works (effortless data access).

In spite of multiple advantages, having a DAL between data sources and data consumers has one possible disadvantage depending on the chosen implementation technology and data. DAL might reduce the speed depending on the code that needs to be executed for the data integration. However, Performance is highly dependent on the size and complexity of data sources and the implementation of the DAL. In addition to that, performance is a non-functional requirement and it is measured when choosing the technology for implementation. Since the advantages of having DAL outweigh the possible disadvantages, we chose this approach to solve the problem of data integration. In the following section, we investigate the technical frameworks to create a DAL.

4.2 Technology options

For implementing the DAL, the following technologies are investigated: Foreign Data Wrappers, Blaze, GraphQL, Jupyter Notebook, and Data Warehouse. In this section, each of the technologies are explained. In addition, two important aspects of the technical requirements *data joining* and *Information hiding* are mentioned per option.

4.2.1 Foreign Data Wrappers

Foreign Data Wrappers (FDW) technology was introduced in PostgreSQL 9.1 in 2011. FDWs are extension methods for PostgreSQL server in order to query external data sources. A foreign data wrapper is an object that wraps code for communication with an external data source using regular SQL queries. There are three main concepts when designing a foreign data wrapper [2]:

1. **Foreign server**: An object that specifies how to locate a certain external data source. It belongs to a foreign data wrapper.

2. **Foreign table**: An object that describes the data structure contained in an external data source. It belongs to a foreign server.

3. **User mapping**: An object that contains credentials to authenticate with an external data source. It belongs to a foreign server and a database user.

When a user attempts to access a foreign table, PostgreSQL knows how to reach the data source (via the foreign server that belongs to the foreign table), how to authenticate (via the user mapping) and what functions to use to perform this connection and exchange data (via the foreign data wrapper)[3]. Figure 4.1 shows the communication between the above-mentioned components.

Data joining

Data joining between multiple data sources is possible by writing the join routines in FDW code. In addition to having remote joins on two foreign tables, there is also the possibility to join between a local table and a foreign table .

Information hiding

Foreign tables in PostgreSQL are the interface for data consumers. All the foreign tables are treated as if they were local tables. Therefore, the storage technology, the API, the location, and the access mechanism are hidden from the data consumers.

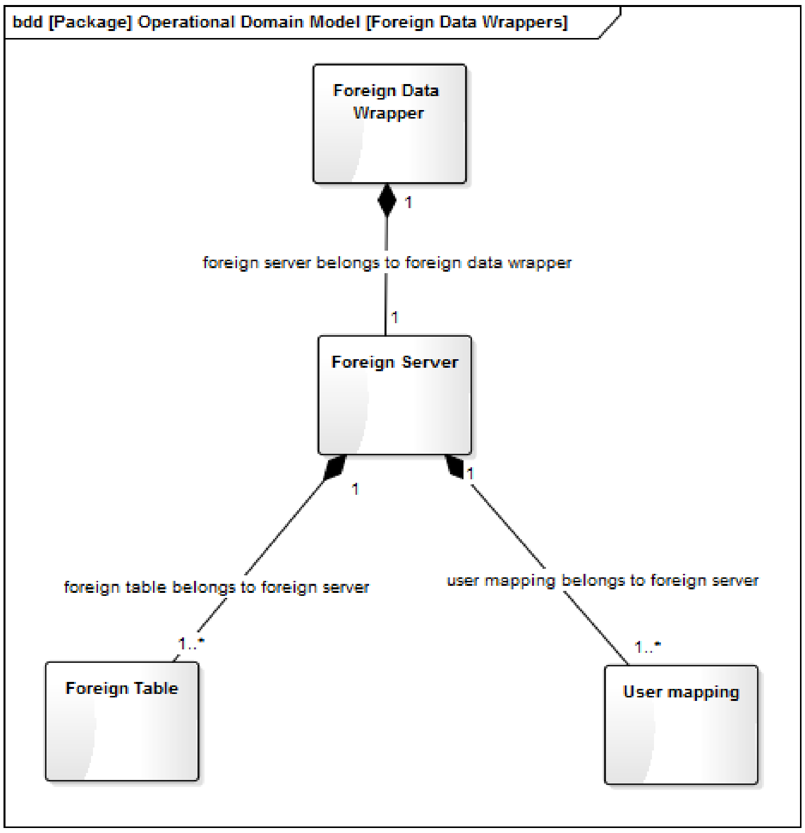
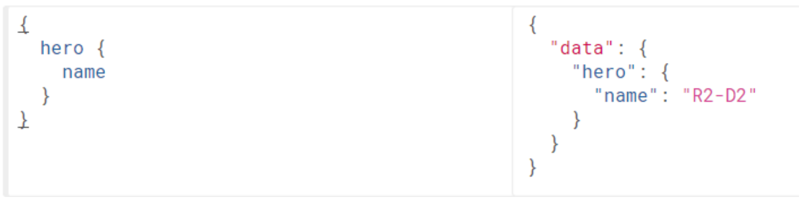

bdd [Package] Operational Domain Model [Foreign Data Wrappers]
Foreign Data Wrapper
1
foreign server belongs to foreign data wrapper
1
Foreign Server
1
1
foreign table belongs to foreign server
user mapping belongs to foreign server
1..*
1..*
Foreign Table
User mapping

Figure 4.1: Foreign data wrappers

4.2.2 GraphQL

GraphQL is an open source query and specification language for APIs. In GraphQL, the following concepts are important:

- **GraphQL query**: A GraphQL query is a string that is sent to the GraphQL server to be executed. For every query, resolver functions explain to the GraphQL server about how to respond to a query. Finally, the GraphQL server returns results in the shape of the requested query [4]. In the example below, the user wants to get the hero name. On the left side, *query* is depicted. On the right side, *data* is depicted.



```
{
  hero {
    name
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2"
    }
  }
}
```

- **GraphQL schema**: GraphQL cannot execute a query without a type system. A schema defines types and their relationships. It is written in the "*GraphQL Schema Language*" [5]. It specifies which queries can be made against the server and what responses can be returned back from the server (See below example).

```
type character {
name: String
}
```

This language is readable in the way:

- Character is a GraphQL object type with some fields.
- Name is the field of the character type. This field can appear in any part of a GraphQL query that operates on the character type.

Data joining

GraphQL server joins the data from multiple data sources by following the relationship that is defined in the schema.

Information hiding

For every data consumer, the query schema, the query resolver, and the connections to the remote data source can be implemented once and used thereafter. Therefore, the data consumer has no information about the storage technology, the access mechanisms, and the APIs of the data sources.

4.2.3 Blaze

Blaze is a Python library that allows the data consumers to query data living in a variety of data sources. Blaze gives Python users a familiar interface to query external data such as SQL databases or raw files. A server hosts data remotely and a Python client can send Blaze queries to the server and get the data. The following example shows a mount of two different resources on the server [6]:

```
server = Server({
        'csvfile': resource('mycsv.csv'),
        'sqlitetable': resource('sqlite:/mysqlitedb.db')
         })
```

Thus, the CSV file and the SQLite table can now be accessed through Blaze Server. Figure 4.2 shows the Blaze solution.

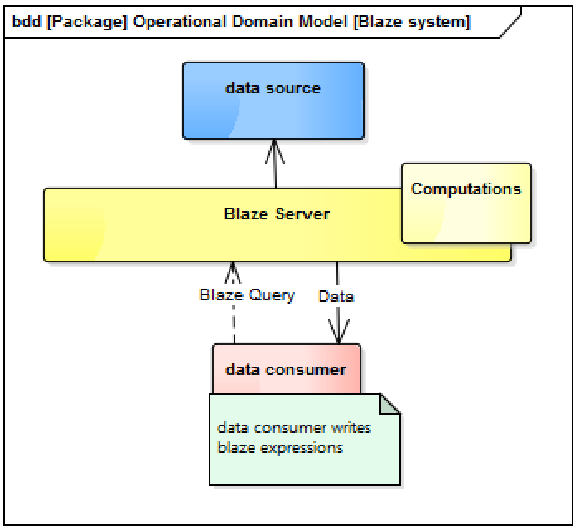

bdd [Package] Operational Domain Model [Blaze system]
data source
Computations
Blaze Server
Blaze Query
Data
data consumer
data consumer writes blaze expressions

Figure 4.2: Blaze

Blaze Expressions

Blaze separates the computations from the query definitions. Users write predefined queries in the format of Blaze expressions and execute the queries on different data sources. Blaze expressions explain the computational work flows symbolically. They allow the developers to write and double check their computations before applying them to data. At the core, Blaze is a way to express data and computations. One Blaze query can work across data sets ranging from a CSV file to a distributed database[7]. In the following example, an abstract table is built:

```
hero = TableSymbol('hero', '{id: int, name: string, droid: int}')

query = hero[hero['id']==2000]['name']

compute(query,dataset)
```

The query is computed on different back-ends no matter what the back-end storage technology is.

Data joining

Blaze does not have a way to join resources in different back ends. This means there is no good way to join the two different data sources [8].

Information hiding

Blaze provides a uniform access to a variety of common data formats. Blaze Server builds off of this uniform interface to host data remotely.

4.2.4 Jupyter Notebook

Jupyter notebook is a web-based notebook environment for interactive computing. Common uses of Notebooks are: data cleaning and transformation, numerical simulation, statistical modeling, and machine learning [9]. Figure 4.3 shows the Jupyter Notebook solution.

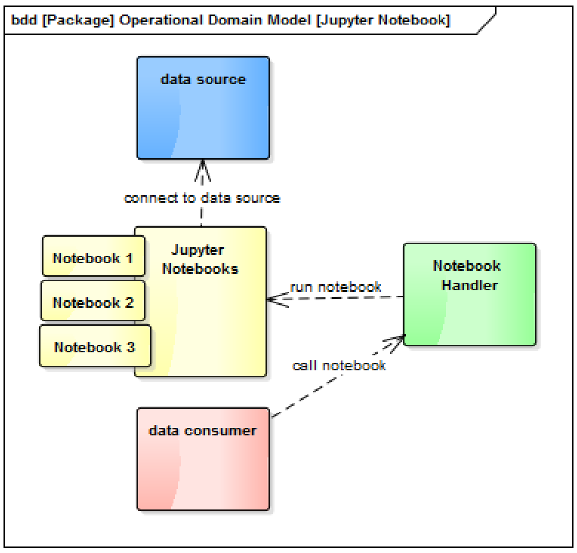


Figure 4.3: Jupyter Notebook

By investigations on the Jupyter Notebook solution, we concluded that it cannot be a DAL. The reasoning behind this conclusion is that the Notebooks are designed as interactive interfaces in which users write code rather than a service to get data. Jupyter Notebook community confirmed this conclusion [10]. Therefore, in the further investigation, Jupyter Notebooks are not included as a solution for the DAL. However, they remain as the data consumer(ODAS).

4.2.5 Data warehouse System(DWH)

The core concepts of data warehouse (DWH) is the Extract, Transform, Load (ETL) process by which data is extracted from data sources, transformed, and loaded into a central repository. This creates a single view from different data sources (See Figure 4.4).

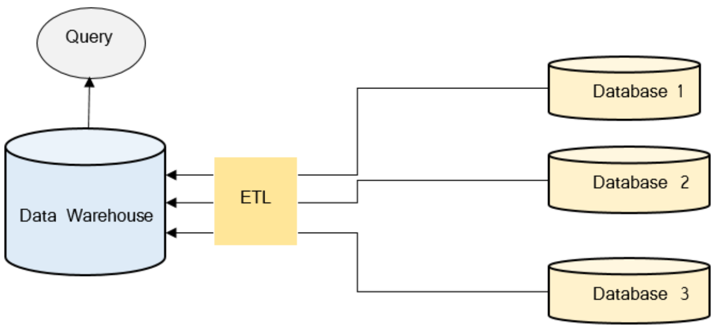


Figure 4.4: Data warehouse

In spite of potential advantages, such as performance, followings are multiple disadvantages for using DWH as the DAL:

- Transferring all data from different back-ends to a single central repository does not satisfy the requirement FR-03, which emphasizes the importance of not having the data replication.
- DWH is not a flexible approach when data sources are frequently updated (such as: interface change, data shape change). When data sources are updated, the ETL process needs to be re-executed for synchronization [11]. Therefore, the requirement FR-08 is not satisfied.

Therefore, in the further investigation, DWH is not further considered as a possible DAL solution.

4.3 Evaluation

In order to design and implement a data access layer that satisfies the functional and non-functional requirements, a thorough evaluation of different technology options is needed. Technology options are Foreign Data Wrappers, GraphQL, and Blaze. Table 4.1, shows the evaluation results of the technology options based on the criteria, derived from the requirements. FDW is the technology that satisfies the criteria(✓for all the criteria with two out-of-the-box solutions).

Table 4.1: Evaluation Criteria based on the Requirements

Req	**Criteria**	**FDW**	**GraphQL**	**Blaze**
FR-01	Information hiding from data consumers	✓	✓	✓
FR-02	Unified API to data consumers	✓	✓	✓
FR-02-01	Standard tools (MATLAB and R) access	✓	✓	✖
FR-03	Non-materialized data access	✓	✓	✓
FR-04-01	HDF5 store technology support	✓	✓	✓
FR-04-02	Web services technology support	✓	✓	✓
FR-04-03	SQL technology support	✓out-of-the-box	✓	✓
FR-04-04	CSV technology support	✓out-of-the-box	✓	✓
FR-04-05	MDX and cube support	✓	✓	✓
FR-05	Authorized and Authenticated data access	✓	✓	✓
FR-06	Data writing support	✓	✓	✖
FR-07	Concurrent access support	✓	✓	✓
FR-08	Real-time data access	✓	✓	✓
FR-09-01, FR-09-02	Data filtering	✓	✓	✓
FR-09-03	Data joining support	✓	✓	✖
NFR-02	Easy to extend for supporting new data sources	✓	✓	✓
NFR-03	Data consumer can still write Blaze queries	✓	✖	✓
NFR-03	Data consumer can still write SQL queries	✓	✓	✓
NFR-03	Access from C# language	✓	✓	✖
NFR-03	Access from Java language	✓	✓	✖
NFR-03	Access from Python language	✓	✓	✓
NFR-04	Flexibility to write queries	✓	✖	✓

For evaluating the requirement NFR-01, an experiment is needed to measure performance of each of the solutions. The experiment determines how the technologies perform in term of responsiveness and stability. In addition, the experiment helps to investigate, measure, and validate other quality attributes of the system (such as NFR-03 and NFR-05). In the next section, the evaluation experiment is explained in details.

4.4 Technology Evaluation setup

For evaluating the performance of the technologies, an evaluation experiment was setup. This experiment led to the decision of which solution best satisfies the requirement regarding performance. The design of the experiment is shown in Figure 4.5. Four programs are created to do the same task of providing data from data sources to data consumers. These programs are: RAW, DAL1 to DAL3. The dashed lines are the information flow from data source to the data consumer through DALs. The solid line is the RAW information flow from data source to the data consumer.

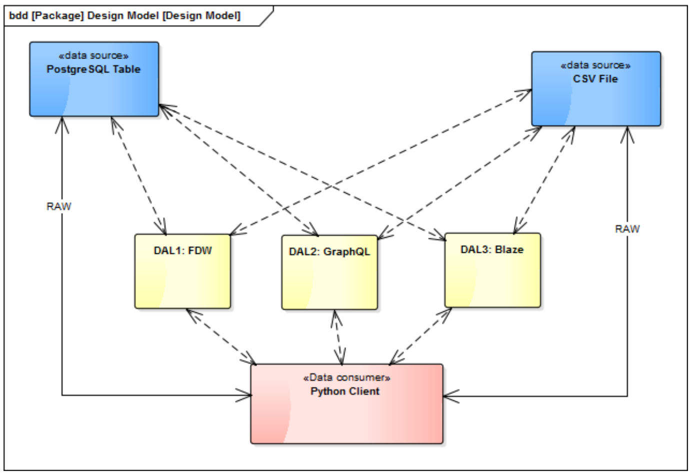

bdd [Package] Design Model [Design Model]
«data source» PostgreSQL Table
«data source» CSV File
RAW
RAW
DAL1: FDW
DAL2: GraphQL
DAL3: Blaze
«Data consumer» Python Client

Figure 4.5: Design of the experiment

4.4.1 Test Process

Data source

In order to compare the solutions, it is essential to develop a collection of typical inputs that can serve as benchmarks. We choose the benchmark inputs such that they are representative of the typical input to the final solution. Therefore, the solution that performs reasonably well on the benchmark inputs is considered to perform well on all inputs.

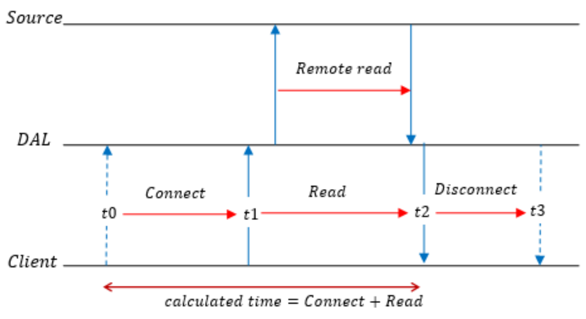
Currently in Océ, data sources that need to be integrated are different from the perspective of storage technology and the access mechanisms. For example Functional Logging data are CSV and HDF5 files while Customer Reporting Service data is stored in an SQL database. Therefore, the performance experiment was carried out for SQL database (relational) and flat files (non-relational). In addition, different sizes of the data sets determines the Scalability of the solutions. Therefore, the input data sets are categorized into three groups including *Empty data set*, *Data set A*, and *Data set B*. Data set A is in the one MB range and Data set B is in the 100 MB range. It is expected that the performance of the solutions has a correlation with the size of the data set.

Query

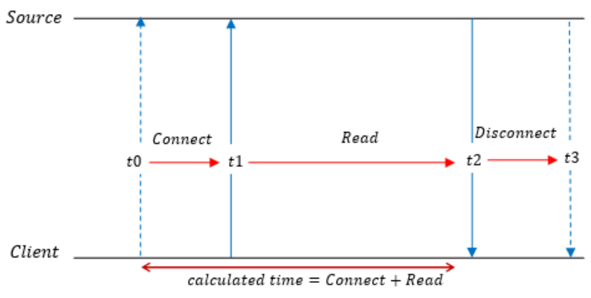
The query group requirement FR-09 are categorized in three groups: Selection of all data, based on conditions and details of a record; Selection of a specific part of the data; and Join of two different data sources. The queries used in this experiment are based on the three categories(See Appendix A). The sizes of data sets were chosen by investigating the typical sizes of data sets within Océ context.

4.4.2 time

The performance time is the sum of the data source connection time (Connect) and data retrieval time (Read)(See Figure 4.6). Figure (a) shows that there is a remote read time when DAL is located between the source and the client. Since the results that are derived from one test might be influenced by external factors, performance time is measured as the mean time of repeating the experiment for 100 times.



(a) Performance Time calculation in DAL solutions



(b) Performance Time calculation in RAW solution

Figure 4.6: Performance Time calculation

In the test, the connection to the data source is established per query, and after the execution of the query, the connection is closed. This helps to perform tests in an environment, which is simulated to the real world. Therefore, this test setup gives a good insight into the performance of the solutions.

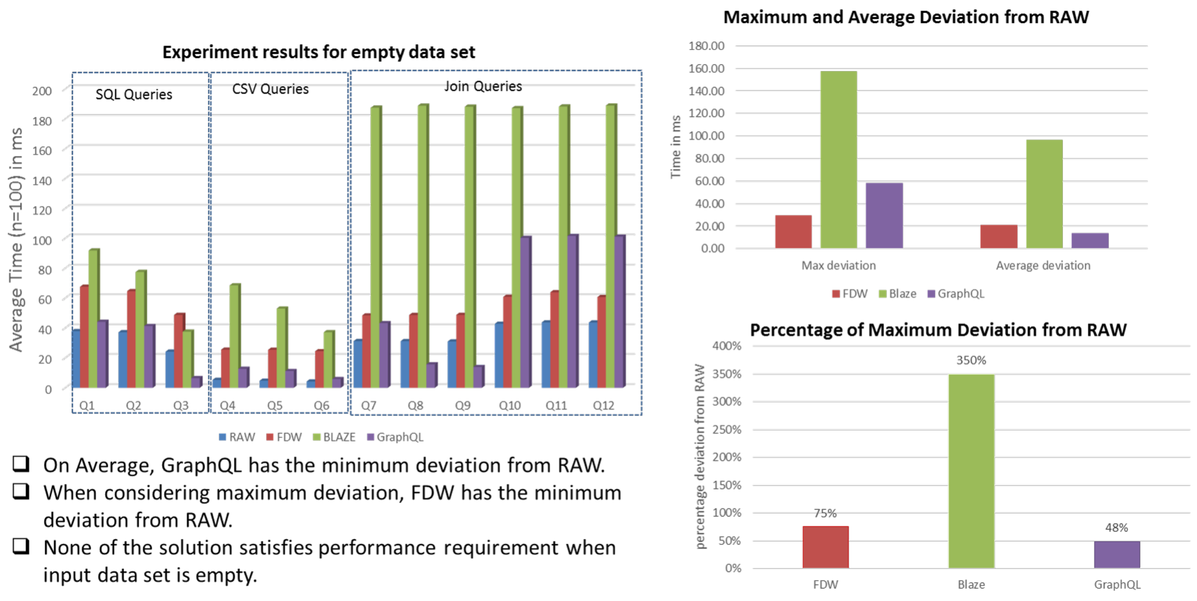
4.4.3 Results of the experiment

The results present the execution time of the queries for each solution when it is compared with the RAW. This gives insight into measuring the speed of each DAL. This helps to choose a technology for implementation. The performance requirement is measured based on the maximum percentage difference of each DALs from RAW.

The results are presented for three data sets. In each case (Figures 4.8, 4.9, and 4.10), the maximum and average difference from RAW solution and the comparison between all solutions for different queries are represented. Looking at the difference between average and maximum deviation demonstrates the stability. In each Figure, Q1-Q12 represents the queries mapped with requirement FR-09 (See Appendix A).

Empty data set

The purpose of having an empty data set is to compare the connection time regardless of the size of the data. This brings a useful insight on understanding the connection overhead of the *DALs* versus *RAW*. Since *RAW* is the direct access, it is also expected to be the fastest solution. However, for combining different data source queries (Q7-Q12), FDW is very closely competing with RAW.



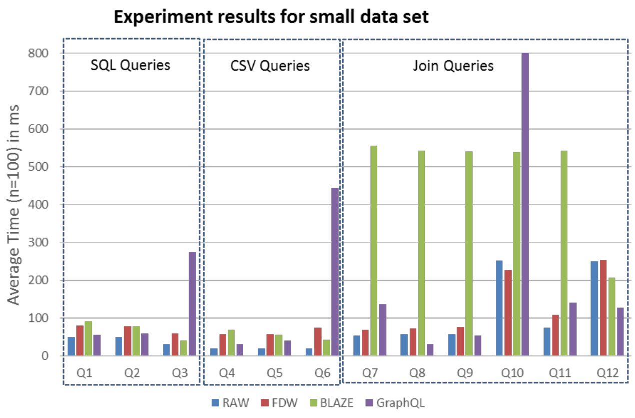
- ❑ On Average, GraphQL has the minimum deviation from RAW.
- ❑ When considering maximum deviation, FDW has the minimum deviation from RAW.
- ❑ None of the solution satisfies performance requirement when input data set is empty.

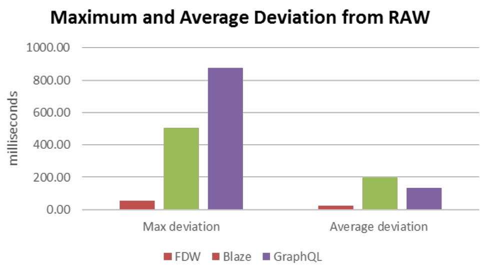
Figure 4.8: Results of the performance experiment on empty data set

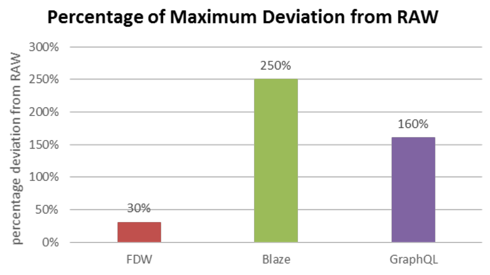
None of the DALs satisfies the performance requirement. This means that *DALs* cannot compete with *RAW* when connection time is the only factor for comparison. This result is similar to what was expected. We expect that *DALs* show their superiority to *RAW* when it comes to Data set A and B when actual data is being retrieved.

Data set A

The purpose of having data set A with 1MB size is to compare the DALs and RAW with small size of data. (See Figure 4.9). FDWs satisfy the performance requirement while GraphQL and Blaze have considerable difference compared with RAW.





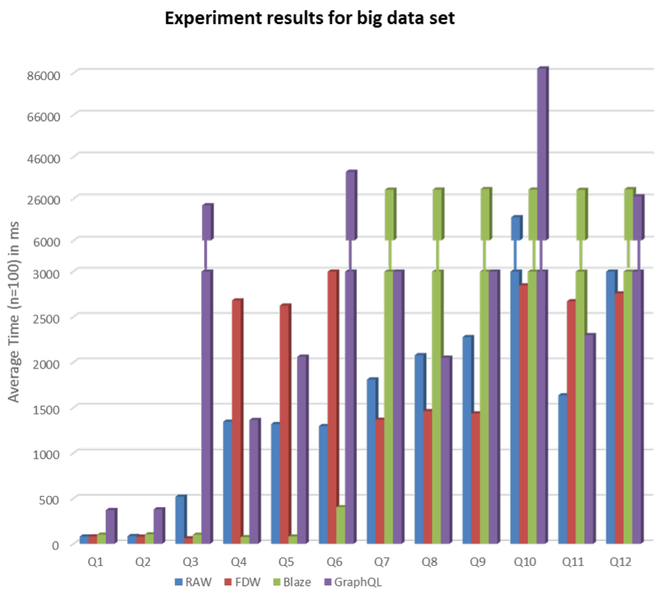


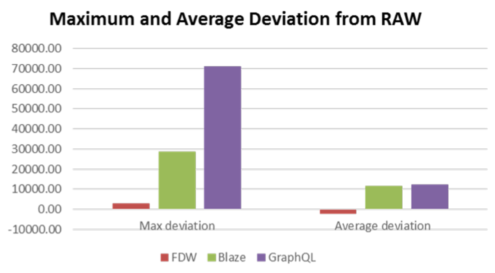
- On Average, FDW has the minimum deviation from RAW.
- When considering maximum deviation, FDW has the minimum deviation from RAW.
- FDW satisfied performance requirement.

Figure 4.9: Results of the performance experiment on data set A

Data set B

The purpose of having data set B with 100MB size is to compare the DALs and RAW with rather large size of data. RAW is the best solution (See Figure 4.10) for queries that point to a single data source. However, for combining different data source queries (Q7-Q12), FDW solution performs faster than RAW. FDWs satisfy the performance requirement while GraphQL and Blaze have considerable difference with RAW.





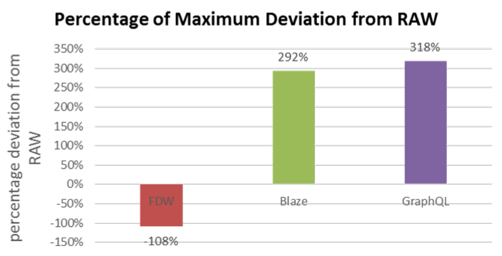


Figure 4.10: Results of the performance experiment on data set B

4.5 Chosen Solution: Foreign Data Wrappers

We summarize the findings as follows:

- Results of the performance experiment show that FDWs satisfy the performance requirement.(Design Criteria: Performance)
- FDWs provide freedom and flexibility in writing queries. Blaze and GraphQL limit the query format to not well-known and specific query language.(Design Criteria: Flexibility)
- FDWs provide SQL query interface, and the querying language is well-known. Note that Blaze and GraphQL can still be implemented on top of the FDWs in case of the specific interest of a data consumer (Design Criteria:Learnability)
- FDWs perform better than RAW when the size of data increases. Therefore, FDWs can satisfy the Scalability requirement by which we expect the solution to continue functioning well when the data source changes in size.(Design Criteria: Scalability)

Hence, based on the result of the evaluation in Table 4.1 and the results of the performance experiment, FDWs are chosen as the solution for the DAL.

Chapter 5

Sytem Architecture

The purpose of designing an architecture for a system is to solve the problem statement formalized with the system requirements. This chapter elaborates on the system architecture based on the chosen technology in Chapter 4. The deployment model is also represented.

5.1 Architecture Overview of the DAL

In this project, Foreign Data Wrappers are chosen for implementation of the DAL. PostgreSQL offers a high-level architecture for this purpose as shown in Figure 5.1. In this client-server model of the DAL, there are three layers: Client, Server, and Data. Any layer is allowed to be modified and changed without affecting the rest of the system as long as the communication interface between the different tiers remains the same. Therefore, the Extensibility requirement (NFR-02), regarding the support of the additional data sources and the data consumers is satisfied. In Figure 5.1, the Extensiblity of data sources is shown by implementing or reusing a new or current foreign data wrapper, as well as extending schema mapping for new data sources. Here, we took advantage of PostgreSQL's Extensible architecture.

The majority of the design choices occur in the supporting mechanism section in Figure 5.1: 1) Schema Mapping (mapping the data sources with the foreign tables), 2) Foreign Tables Generation Mechanism. These topics are discussed in depth in the System Design (Chapter 6).

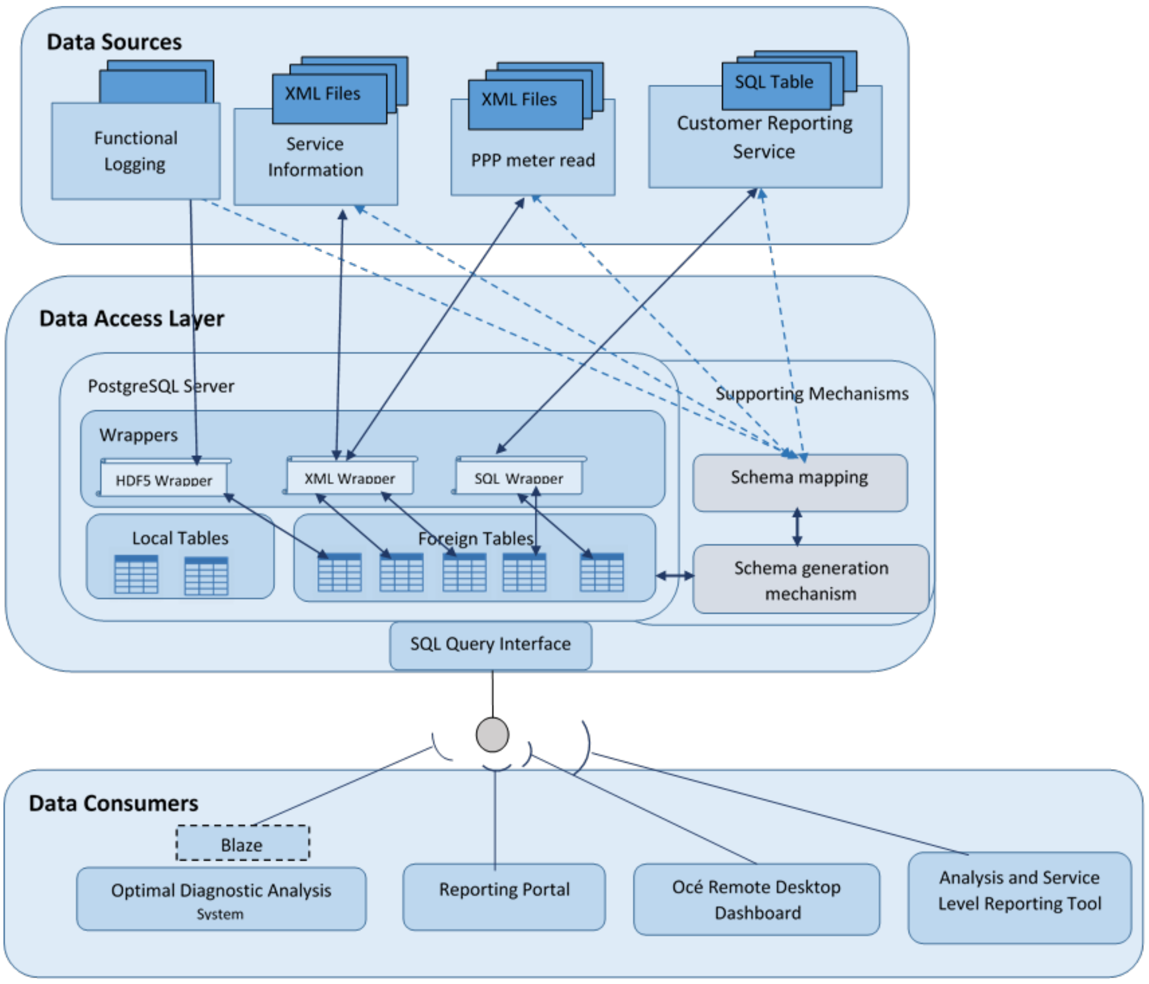


Figure 5.1: Overview of the DAL

The DAL is a system that receives queries from client applications (users). To do this, it retrieves data based on the query specifications from the data sources. Then, the DAL performs data filtering inside the data sources, to return only the requested data to the data consumers and nothing more. Data Filtering can happen at one of the following two levels:

1. At the **Data Source**- When the data source supports the data filtering capabilities: Then the query conditions can be passed to the data source. In this case, the retrieved data from data sources are already filtered based on the query conditions.

2. At the **Database Server**- When the data source does not support the data filtering capabilities: Then the PostgreSQL database Server performs the data filtering .

In both cases, the data consumer gets the filtered data from the DAL. Figure 5.2 demonstrates the sequence diagram of the query execution in the DAL. In Figure 5.2, the "qualifiers" keyword represent the filtering conditions.

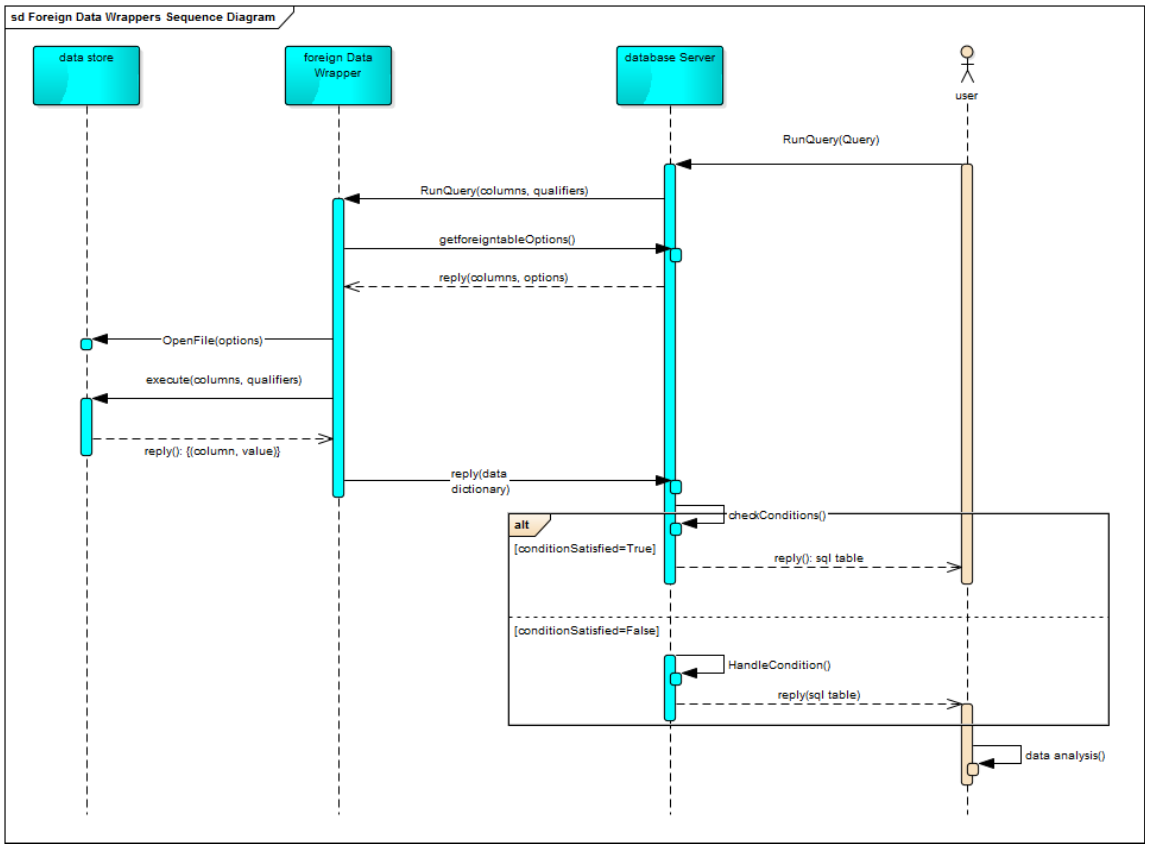


Figure 5.2: Foreign Data Wrappers Sequence Diagram

In addition to data filtering, the DAL supports the data joining by implementing the join mechanism and linked schema in the Foreign data wrappers. This topic is further discussed in the system design chapter.

5.2 Deployment View

The deployment view concerns the structure of the product regarding software to hardware mappings and distribution aspects. Given that the very core of the system is client-server communication, client being the data consumer systems and server being a machine that hosts the service, a straightforward approach is followed when designing the deployment view of the system.

PostgreSQL was chosen as the database since it provides the Foreign data wrappers API. There are several FDWs implemented and added as an extension to PostgreSQL. Other databases have the similar feature. For example, *MySQL's FEDERATED storage engine* can connect to another MySQL database, but not to the other RDBMSs; Microsoft SQL Server has the similar concept in *Linked Servers*. However, the linked servers are configured to query relational data sources.

Figure 5.3 depicts the designed deployment view of the DAL.

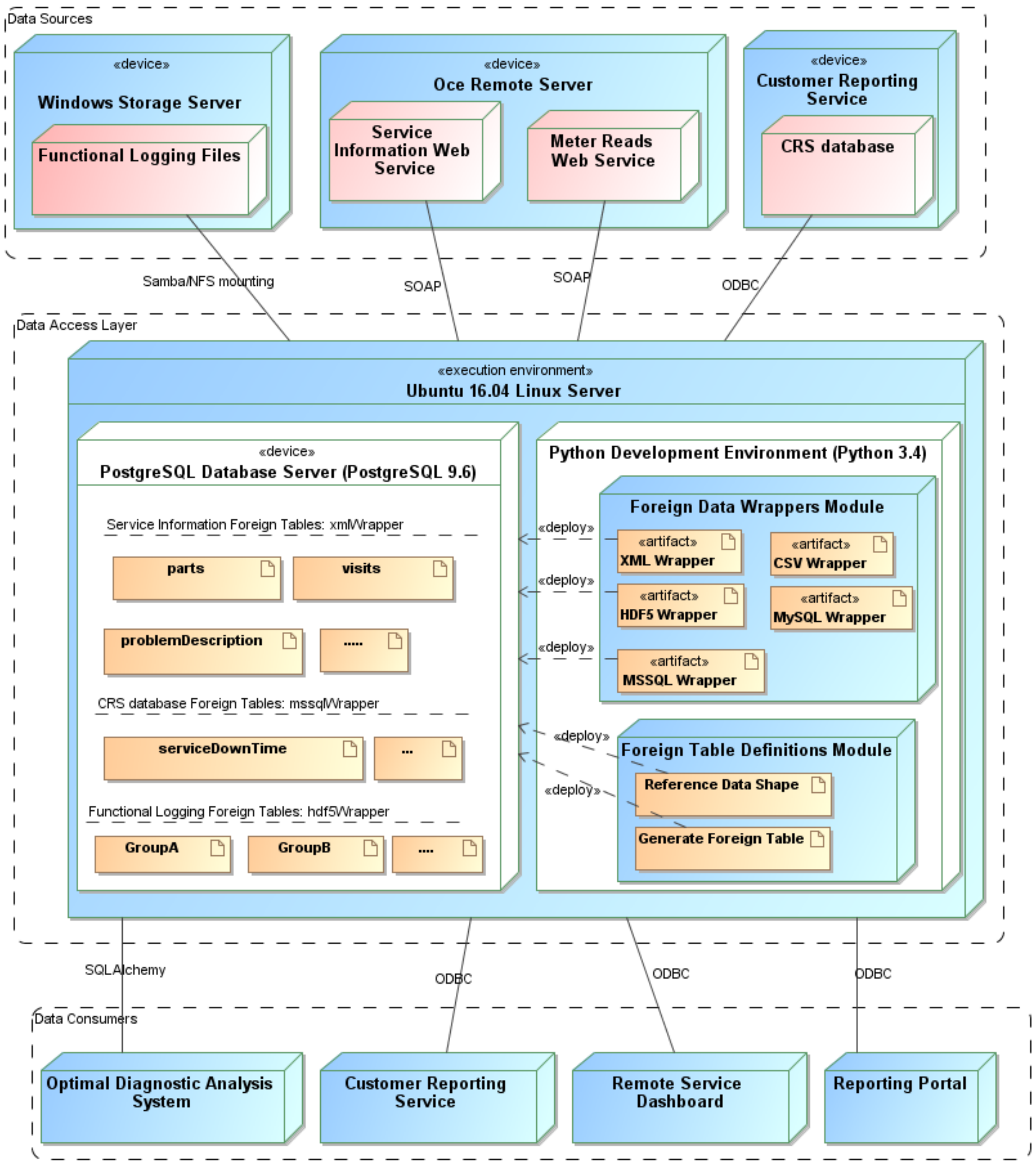


Figure 5.3: Deployment of the DAL

This deployment model satisfies the requirements FR-03, as each data source remains on its current location. The foreign tables are only the table definitions and include 0 bytes of data. In this model, ODAS, as a Python environment, is able to connect to DAL with the *SQLALchemy* library for writing Blaze query while CRS system writes SQL queries over an ODBC connection. Linux is chosen as the execution environment because building the open source packages in windows is not straightforward. By investigating the development communities of foreign data wrappers, Ubuntu is a common Linux distribution for development of the DAL. Foreign Data Wrappers can be implemented in two languages: C and Python. In this project, Python is chosen because of ease of prototyping and the existing Python knowledge. In addition, Python is a de facto standard in the Business Intelligence (BI) community, for which DALs are very relevant.

Chapter 6

System Design

The previous chapter elaborates on the high-level architecture of the DAL. This chapter dices into the system design. The system design includes the schema mapping and the foreign table generation for each use cases.

6.1 Use case 1: Combine the Service Information and the Functional Logging for ODAS

In the problem analysis chapter, each of the data sources as well as each of the data consumers are explained. Use case 1 provides the ODAS users a combination of the Service Information and the Functional Logging. In the System Requirement Chapter, the use case, its actors, and main key concepts are introduced.

The difficulty of the current situation is the amount of time and effort it takes for the ODAS user to access data from these two data sources and join them. As such, this case focuses on providing data access to ODAS and handling the joining of data between sources. In this case, the ODAS user only gets back the parts of data that she/he is interested in. In the next section, the structure of each of the data sources, the mapping of data sources with the foreign tables, and the foreign data wrappers are explained.

6.1.1 Functional Logging: HDF5 files

Functional Logging data is stored in an HDF5 format. HDF5 is both a data model and a file format, for supporting a variety of data types. It is designed for high volume and complex data. Data is stored in a hierarchical format on a daily basis (See Figure 6.1).

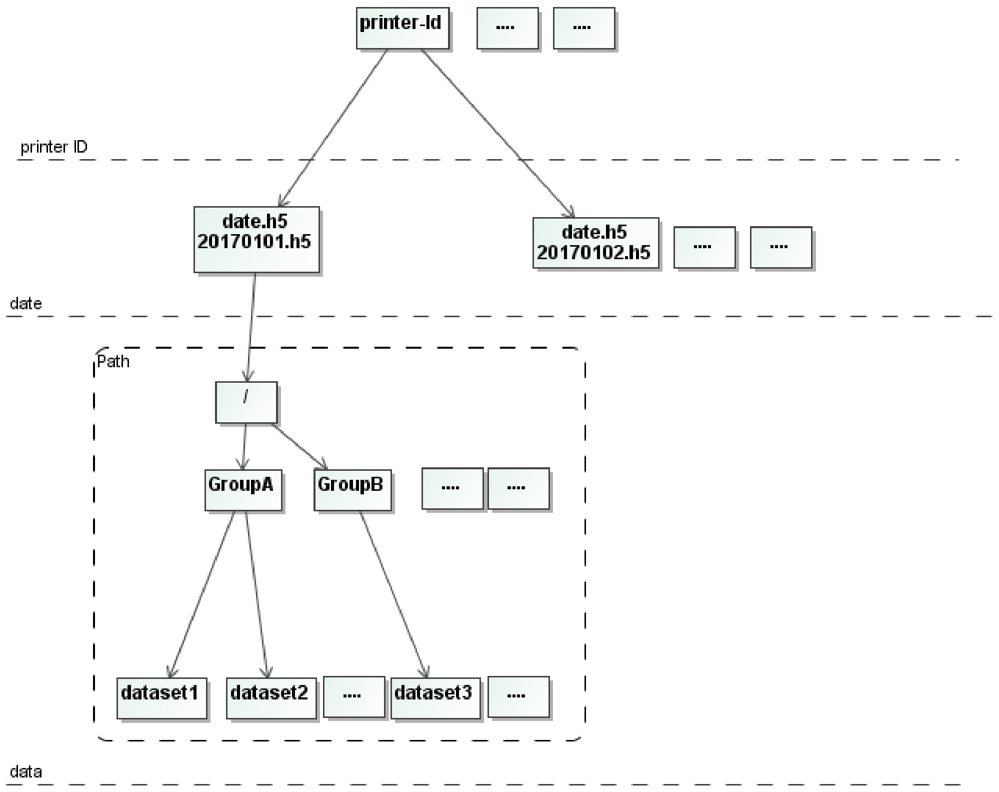


Figure 6.1: Hierarchal Format of HDF5 Files

Each path points to a completely different set of data. In order to map a HDF5 file with a foreign table, the following design decision is made:

Table 6.1: Mapping of the Foreign Tables with the Functional Logging

Choice	Table definition	Number of tables
One table per path	path(printer-id, date, c1, ..., cn) , where c1,...cn is the set of all column names for a path	Equals with the number of path

This approach is user friendly since in the functional logging, a path is requested based on the user interest. This means that the users specifically point to the path that they are interested in. In the HDF5 files, the data shape can be different for each different path. Hence, the mapping problem is solved by having one foreign table per path in the HDF5 file (See Figure 6.2).

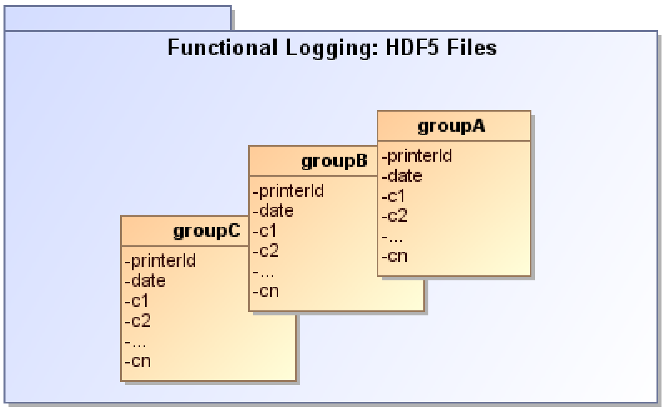


Figure 6.2: Mapping of the Foreign Tables with the Functional Logging

6.1.2 Service Information : Extensible Markup Language (XML) files

Service Information is stored as an XML file per machine. Each file contains the latest logbooks for a specific machine. In the logbooks, multiple interesting groups of data need to be retrieved and mapped with foreign tables. Figure 6.3 shows the entities and the relationships between them, retrieved from logbooks.

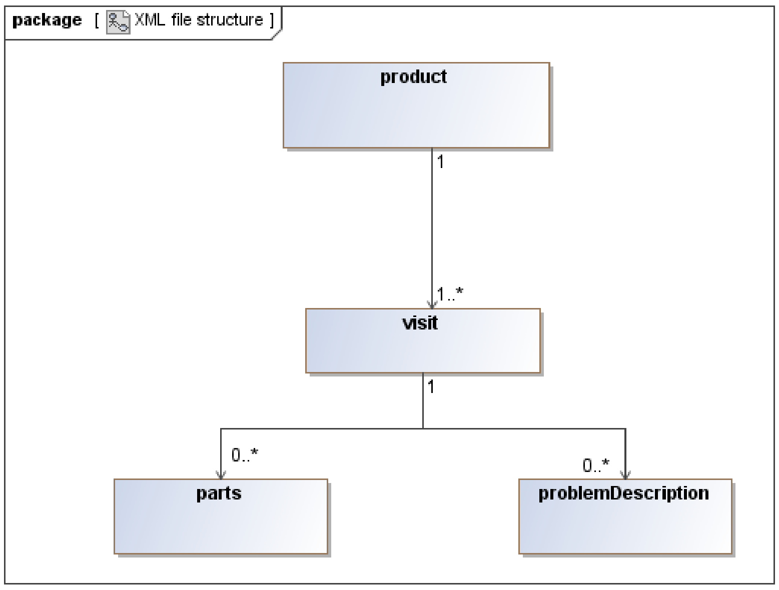


Figure 6.3: The structure of Service Information XML file

By understanding the existing entity-relationship in the XML file, the design decision is to create one foreign table per entity. Entities are *visit*, *parts*, and *problemdescription*. Each *visit* might have multiple *parts* and *problem description*. In order to map foreign tables with these entities of an XML file, there is a need to define relationships between the foreign tables. The *parts* and *problemdescription* entities should be distinguishable for different *visits*. To achieve this, a "Foreign key relationship" between Foreign tables is defined between for *parts* and *problemdescription* entities. Figure 6.4 shows the mapping of the entities with the foreign tables and the relationships between them.

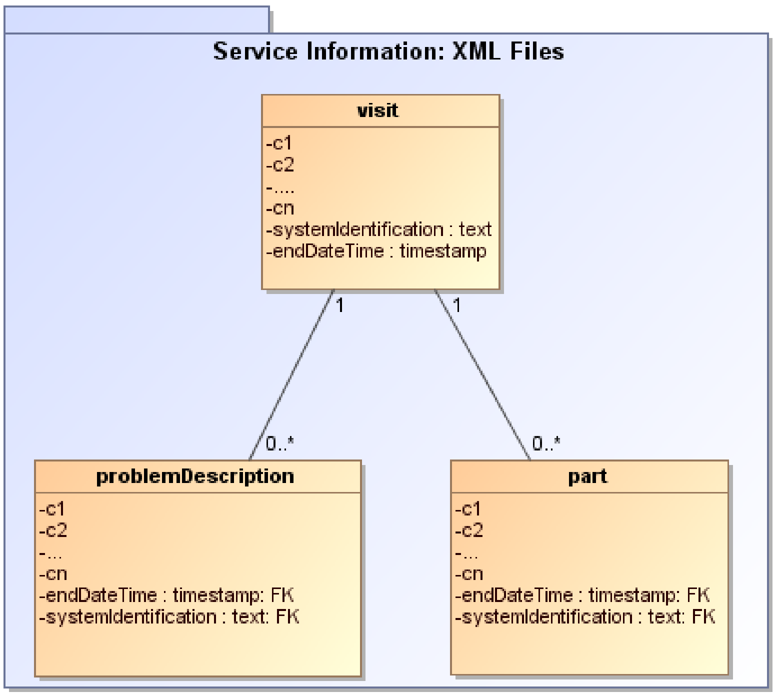

Service Information: XML Files
visit
-c1
-c2
-....
-cn
-systemIdentification : text
-endDateTime : timestamp
1
1
0..*
0..*
problemDescription
-c1
-c2
-...
-cn
-endDateTime : timestamp: FK
-systemIdentification : text: FK
part
-c1
-c2
-...
-cn
-endDateTime : timestamp: FK
-systemIdentification : text: FK

Figure 6.4: Mapping of Foreign Tables with Service Information

6.1.3 Foreign tables Creation Mechanism

In the creation process of the foreign tables, these are two important concepts:

- **Evolving data shape**: The data shape shows the structure of the data in a data set. On one hand, this structure can evolve and change for data sets over time. On the other hand, the foreign table definition depends on the data shape. Thus, if the data shape evolves, then the foreign table needs to evolve and change accordingly. Therefore, a *reference data shape* is proposed.

  A reference data shape is created by choosing a specific data set as the reference. For example in case of Functional Logging files, the reference data shape is stored for a *reference printer Id* and a *reference date*. In the Service Information case the reference data shape is stored in the DAL for a *reference system Identification*. Whenever a new data set is added to the data source, the shape of this data set is compared with the reference data shape. If there is an update in the data shape, the update applies to the reference data shape as well.

- **Compatible data types with postgreSQL data types**: When creating foreign tables, data types of the source data shape are mapped with the acceptable data types in PostgreSQL. Therefore, the compatibility of the data source shape with PostgreSQL acceptable data types is considered in the design of the foreign tables.

The sequence diagram of foreign table creation for the HDF5 files is shown in Figure 6.5. When a HDF5 file is created, its shape is compared with the reference data shape and if a new data type is introduced, then the reference data shape is updated. Accordingly, the corresponding foreign table is regenerated. Figure 6.6 shows the creation of the foreign tables for the HDF5 files. It includes the detailed operations and attributes. This also brings a better insight on the mapping of HDF5 files.

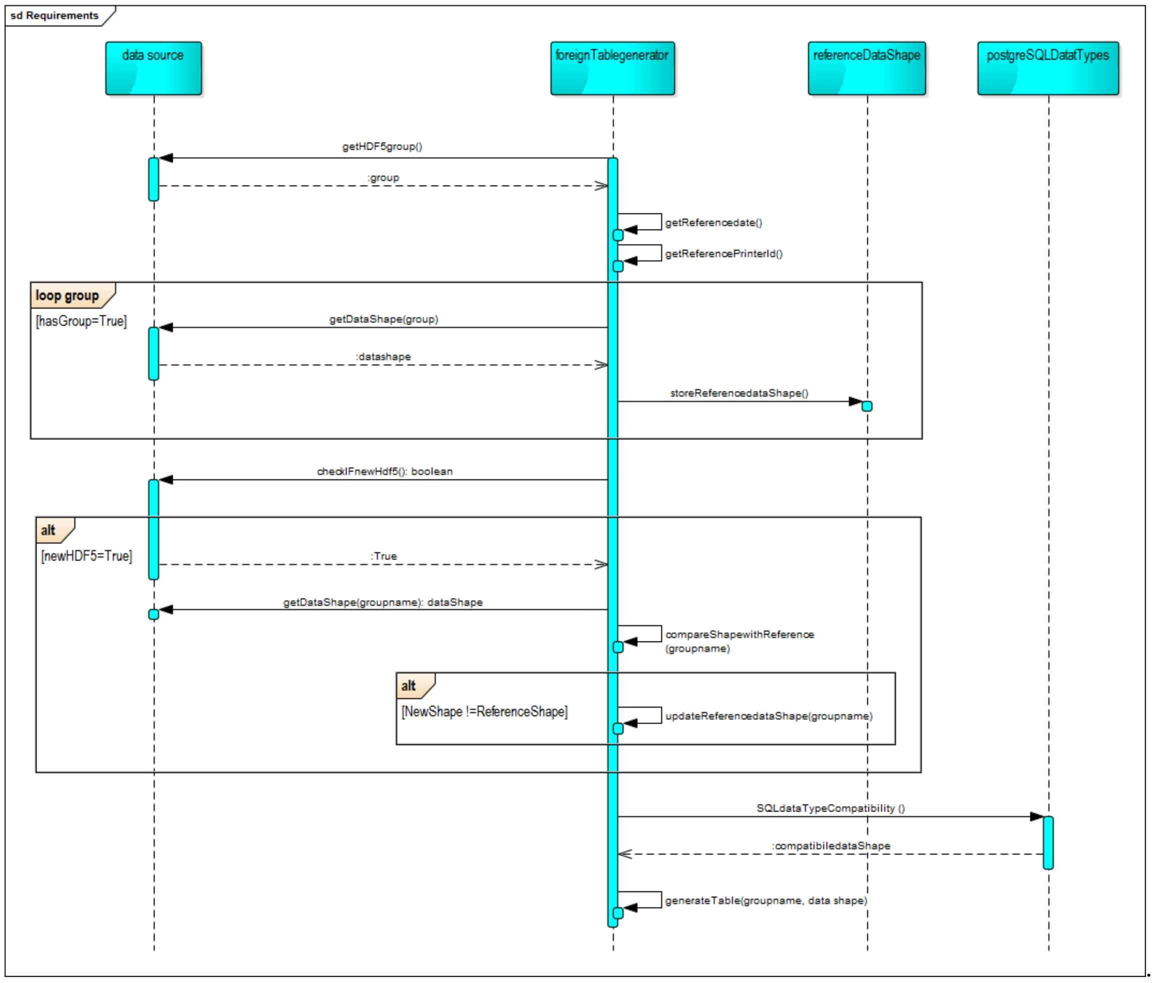

sd Requirements
data source
foreignTablegenerator
referenceDataShape
postgreSQLDatatTypes
getHDF5group()
:group
getReferencedate()
getReferencePrinterId()
loop group
[hasGroup=True]
getDataShape(group)
:datashape
storeReferencedataShape()
checkIFnewHdf5(): boolean
alt
[newHDF5=True]
:True
getDataShape(groupname): dataShape
compareShapewithReference (groupname)
alt
[NewShape !=ReferenceShape]
updateReferencedataShape(groupname)
SQLdataTypeCompatibility ()
:compatibiledataShape
generateTable(groupname, data shape)

Figure 6.5: Foreign table creation Sequence Diagram : Functional Logging

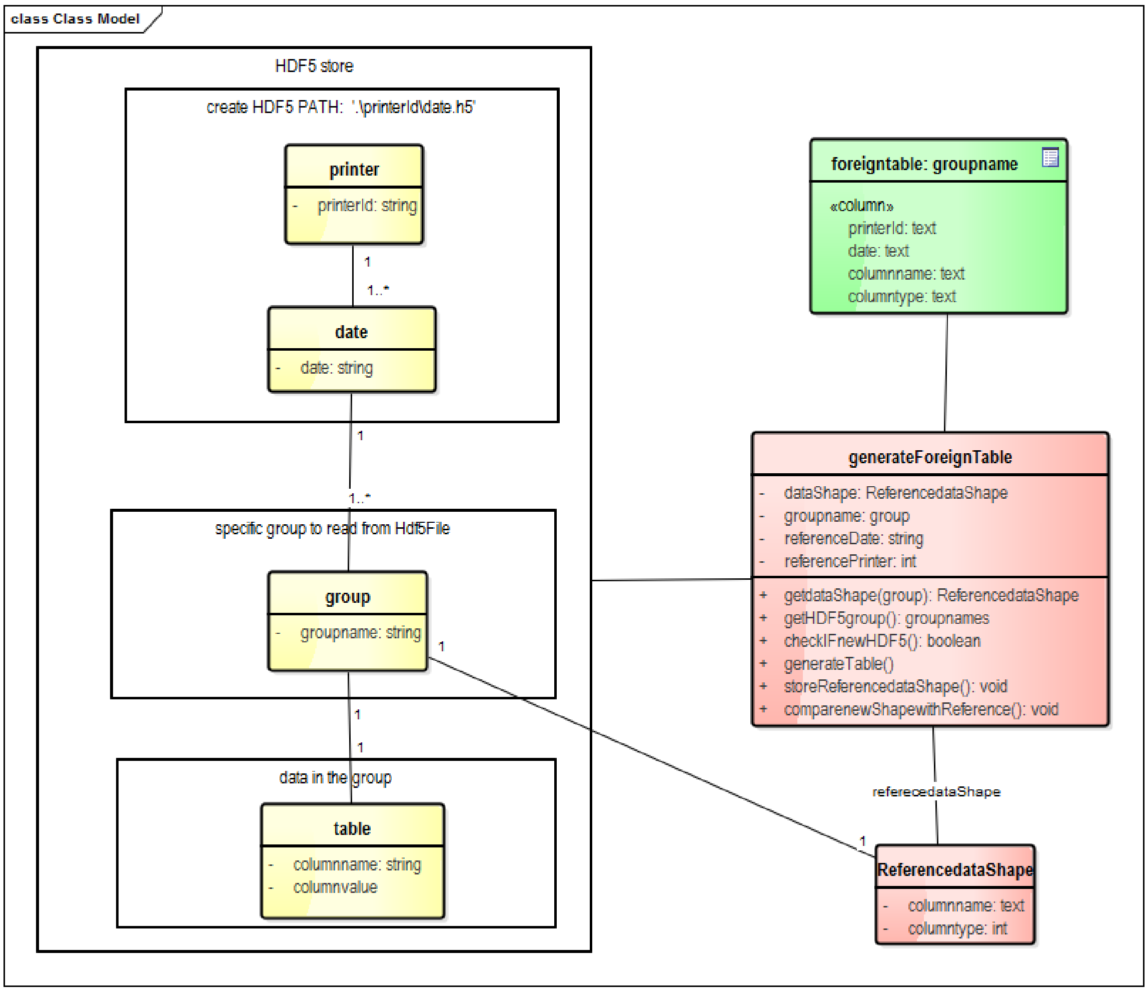


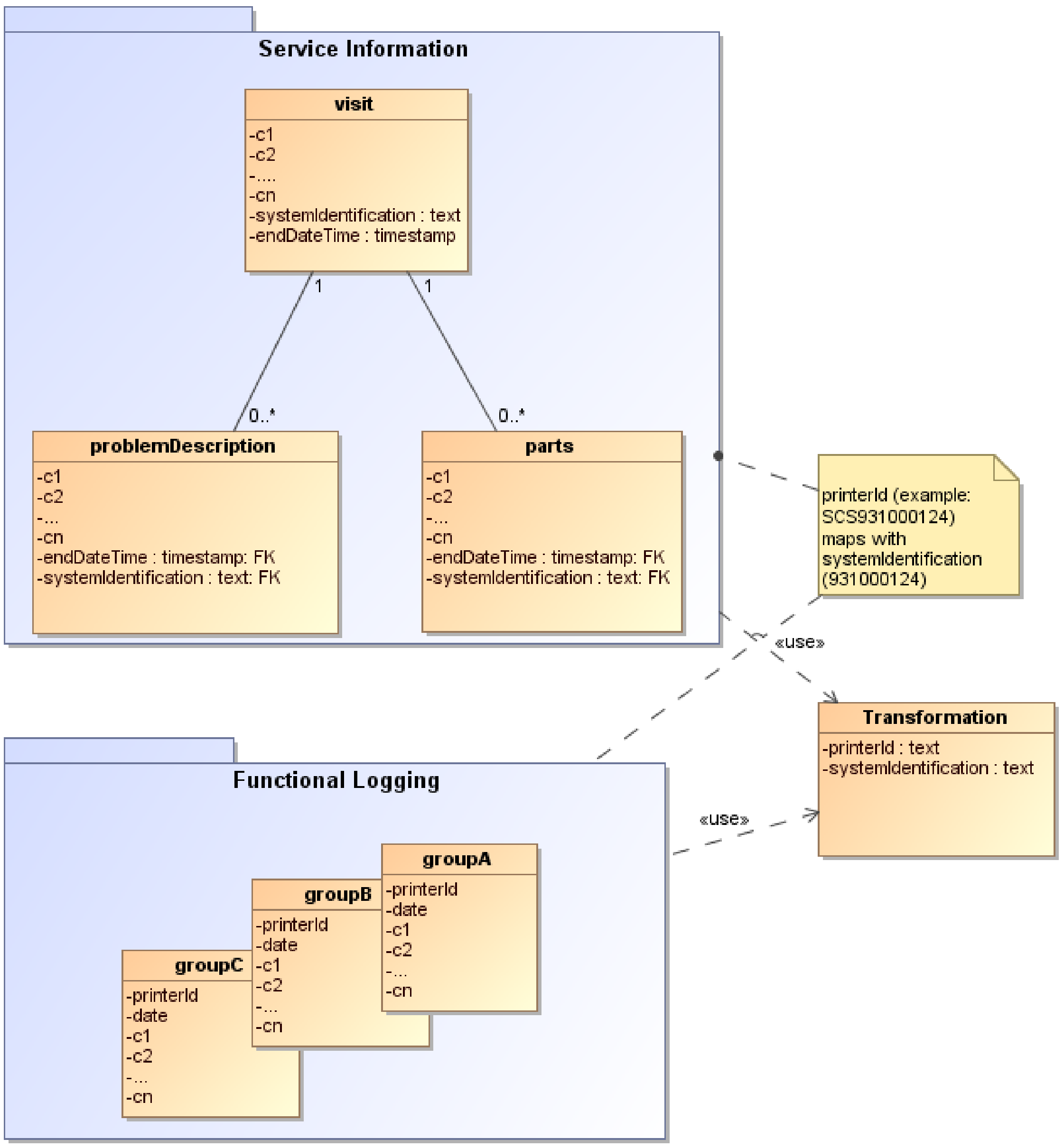
Figure 6.6: Foreign table generation for HDF5 files

6.1.4 Combining the Functional Logging and the Service Information

In order to combine the different data sources, identifying the common columns between them is necessary. In this case,the Functional Logging and the Service Information are identified based on the machine Id. However, the machine Ids have different formats and names in these two different data sources(See Figure 6.7). This incompatibility is an obstacle to join data sources with each other. Therefore, it should be solved. Following approaches can be adopted:

1. **Renaming and Reformatting**: In this approach, the columns which have similar meaning in different data sources are renamed and reformatted to a unified name and format in DAL. For example, *systemIdentification*, *serialNumber*, and *printerId* have the same meaning but are formulated differently in three data sources. In DAL, we can define a unified naming mechanism. This happens in the schema definitions of foreign tables. In this example, we can name every column which are related as *printerId*.

2. **Transformation table**: A transformation table that maps the common columns from different data sources with each other (See Figure 6.7).



This approach makes the queries more complex, since the transformation table should be queried when joining data sources.

Figure 6.7: Combining the Functional Logging and the Service Information

3. **Adding new columns**: We can include all the different formats of a column in the foreign table definition. For the above example, we can have three columns in the schema definition of a data source:*systemIdentification*, *serialNumber*, and *printerId*. This approach is less desired because of the data repetition in DAL.

The design decision is to choose the *Renaming and Reformatting* option. This option brings a unified name and format within different data sources.

6.2 Use Case 2: Combine Service Information and the Customer Reporting Service (CRS) data for ODAS

In this case, a user is interested to combine the *visits*, *parts* and *problem descriptions* from Service Information with specific data of CRS. In order to complete the analysis, ODAS users are interested to have access to the service downtime information from the CRS data store. CRS data is stored in a remote relational data base (Microsoft SQL Server). In order to map SQL tables with foreign tables, one by one mapping is the chosen approach. Therefore, there is one foreign table per SQL table.

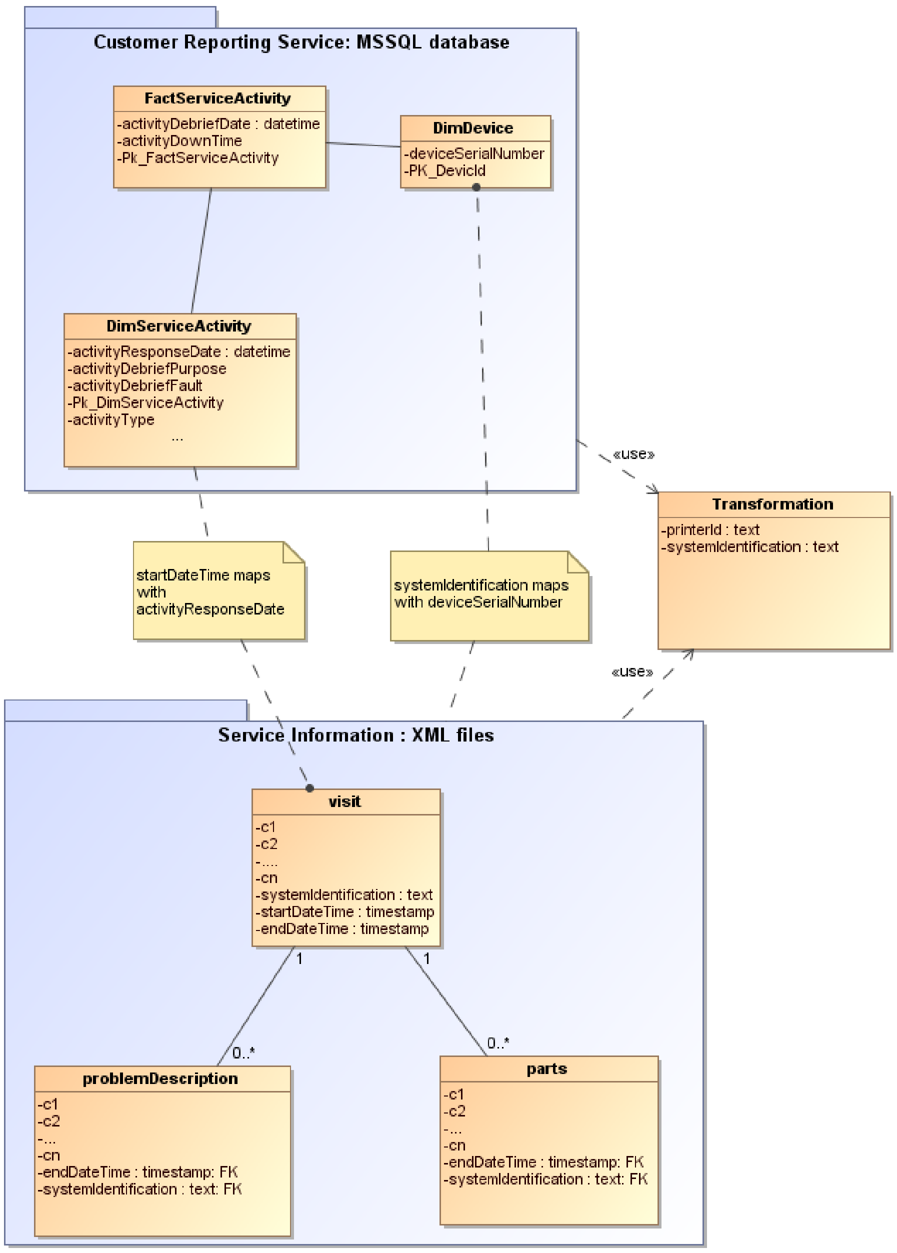

Customer Reporting Service: MSSQL database
FactServiceActivity
-activityDebriefDate : datetime
-activityDownTime
-Pk_FactServiceActivity
DimDevice
-deviceSerialNumber
-PK_DeviceId
DimServiceActivity
-activityResponseDate : datetime
-activityDebriefPurpose
-activityDebriefFault
-Pk_DimServiceActivity
-activityType
...
«use»
Transformation
-printerId : text
-systemIdentification : text
startDateTime maps with activityResponseDate
systemIdentification maps with deviceSerialNumber
«use»
Service Information : XML files
visit
-c1
-c2
-....
-cn
-systemIdentification : text
-startDateTime : timestamp
-endDateTime : timestamp
1
1
0..*
0..*
problemDescription
-c1
-c2
-...
-cn
-endDateTime : timestamp: FK
-systemIdentification : text: FK
parts
-c1
-c2
-...
-cn
-endDateTime : timestamp: FK
-systemIdentification : text: FK

Figure 6.8: Combining the CRS and the Service Information

Figure 6.8 shows the mapping of data types between CRS and Service Information. There are multiple difference regarding the naming of data columns which needs to be handled. This incompatibility is an obstacle to join of data sources with each other. Therefore, the design decision is to define a Transformation table that maps the common columns with each other. The choices discussed in the Section 6.1.4 are also applicable to this case.

Chapter 7

Verification and Validation

The previous chapter described the design of the system. In order to ensure that the system is being correctly created, the process of verification and validation should be put in place. This chapter describes this process.

7.1 Verification

The verification process includes the evaluation on whether the functionality of the system corresponds to the requirements in a good way. The verification process gives an answer to the question: *Did we build the product right?*. Verification is done first by devising an experiment setup to choose the right technology and second by varying inputs and examining the outputs while developing elements of the DAL. The results of the experiment are demonstrated in Chapter 4.

7.2 Validation

Assuring that the DAL meets the needs of the stakeholders is the focus of the validation process. The validation process gives an answer to the question: *Did we build the right product?* In order to perform the system validation, a feedback survey is performed when users started working with the DAL. The survey questions are devised based on the requirements (See Appendix B for the survey questions). Table 7.1 shows the results of the survey.

Table 7.1: Survey Results

Question	**Response**
Without Knowing that the Service Information logbook is a web service, could you access it?	Yes
Without Knowing that the Functional Logging is an HDF5 store, could you access it?	Yes
How simple is it to connect to DAL from your system?	its really easy
How simple is it to join data of different data sources with DAL?	its really easy
Does the DAL provides you the possibility to write query in the language that you are already familiar with?	Yes
How do you rate the functionality of DAL, in terms of performance?	In case of Functional Logging, its slower than what I am using but acceptable. In case of Service Information, its much faster than what I am using now.
Did using the DAL improve the usage of multiple data sources instead of directly accessing them?	Yes
Did you encounter any difficulty while working with DAL?	• I could connect from R to DAL. However, I encountered with the connectivity issues from PowerBi Desktop(Certificate issues: not installed at the client system) • I could query multiple printers from Functional Logging. However, querying multiple printers from Service Information was not possible • With DAL accessing the data is easy. However, there is still an underlying question of who may access which information.
Recommendations	DAL should inform the user if she/he has no access rights.

In addition to survey results, we analyzed whether the DAL meets the requirements. DAL provides a unified query interface in which users have no knowledge of the storage format and the access detail of the data sources. This validates FR-01 and FR-02 requirements. In addition, DAL was tested regarding the access of the third-party software such as MATLAB and EXCEL. This validates FR-02-01.

DAL already supports the existing data sources: HDF5, Web Services, SQL databases, and CSV files (FR-04-01, FR-04-02, FR-04-03, and FR-04-04). The MDX cube data source (FR-04-05) was not included in the use cases. However, since the DAL supports Extensibility, implementing a foreign data wrapper for cube data sources is possible.

By combining the data sources of the Service Information and the Functional Logging, we validated that the DAL is able to join data sources that contain information of the same machine, but originate in different sources. In addition to data joining, data filtering based on conditions and record details is accomplished. Data has remained in its original source and every time that the user writes a query, the latest version of the data is returned. Hence, the user requirements listed as FR03, FR08, and FR09 are supported by the system.

Data consumers are able to send multiple concurrent requests to DAL if the data source supports the concurrency (FR-07). Concurrency of queries against the same foreign table is just like for the local tables handled by PostgreSQL. Therefore, there can be arbitrarily many concurrent readers. In implementation of DAL, to access the Service Information, the latest version of the file is retrieved per query and a unique name is assigned to it so that concurrency of requests is possible. This feature of the DAL validates the requirements FR-07. However, the capabilities of the underlying data store affects the concurrency mechanism. For example, to have concurrent read requests with Functional Logging, HDF5 file system must allow concurrent processes to open the file for reading simultaneously. Requirement FR-07-01 remains as future work.

Data writing support including INSERT, UPDATE and DELETE can be supported in the implementation of foreign data wrappers. In this project, this feature is not implemented because of the stakeholders priorities and interests. In addition, the data writing support requires that the authentication and authorization policies are in place. This remains the future work. Therefore validation of FR-05 and FR-06 are delegated to future work.

Table 7.2 gives an overview of the requirements validation.

Table 7.2: An overview of the requirements validation

Req	**Validation Results**
FR-01: Hiding the technical details	✓
FR-02: Unified API	✓
FR-02-01: Access from MATLAB, Excel, and R	✓
FR-03: No data replication	✓
FR-04: Support of all existing data sources	✓
FR-04-01: Support of HDF5 technology	✓
FR-04-02: Support of Web service technology	✓
FR-04-03: Support of SQL database technology	✓
FR-04-04: Support of CSV technology	✓
FR-04-05: Support of MDX technology	Future work
FR-05: authorized and authenticated SELECT	Future work
FR-06: authorized and authenticated WRITE, INSERT, UPDATE	Future work
FR-07: concurrent request to data sources	✓
FR-07-01: concurrent request to sources that do non supported concurrency	Future work
FR-08: access to latest version of data	✓
FR-09: ability to execute queries including	✓
FR-09-01: data filtering based on conditions and record details	✓
FR-09-02: data filtering based on specific fields of data sources	✓
FR-09-03: data joining from different data sources	✓

Generally, the validation was taken into consideration by the agile way of working: weekly progress meetings, regular evaluations and discussion about the ideas and requirements which were prioritized by the stakeholders.

Chapter 8

Conclusions

This chapter explains the results achieved by this project, the limitations, and the recommendations for the future works.

8.1 Results

The result of the project is a system that provides access to a combination of several data sources. The functionality provided by the system is accessed using its query interface. After implementation, the system became available to selected users in order to support them with their data access activities. They also evaluated the system.

Choosing the right technology is a difficult process, since preferences and needs of the users are different. Users have different set of skills and they prefer to work with the technology that they are familiar with. This was a challenge during the project. Therefore, Learnability was considered an important requirement. The system achieved the Learnability since the query languages is based on the user preference. For example, ODAS users could still write Blaze queries while the default query language of the system is SQL.

Providing a high-performance data access layer depends on factors such as the implementation technology. Therefore, performance was taken into consideration by designing an experiment to choose the technology that has least performance difference with the current situation of data access in Océ. In addition, bringing this possibility for the user to retrieve only the required data had a positive impact on the performance.

Since data is generated from different machines and services, their formats are different and they often have incompatibility with each other. This was also a challenging aspect of the project. To handle this, a schema and type system in the format of foreign tables was introduced. By having the schema mapping, related data in different data sources were linked.

8.2 Limitations and Future Work Recommendations

This section introduces the identified future works. It also includes the limitations due to constraints, such as time.

- Due to the time of the project and also the complexity of the security topic, the authorization and authentication policies were not taken into considerations when designing the DAL. However, with not having the adequate policies, users might access restricted and sensitive data. Therefore, the system should be extended to provide comprehensive security policies.

- The desired goal in performing data filtering is to pass as many data filtering conditions as possible to the data sources. This helps to improve the performance by only returning back the requested data. However, the format of the data sources has influenced the implementation of DAL. For example, the Functional Logging are HDF5 files and their storage format does not support the data filtering. Therefore, DAL performed filtering when it received the unfiltered data from the data source. Changing this will have positive impact on the performance. Therefore, it is strongly suggested for the data source providers to considering the data filtering functionality when storing the data. In case of Functional Logging, this has been already communicated with the data providers.
- In the system design and implementation, we focused on two use cases. A set of use cases was introduced along with the system requirements. Including more data sources that support the data analysis and reporting process, makes the DAL a more comprehensive system. Thus, the data consumers refer to DAL for their data analysis purposes. This leads to the acceptance and usage of the system. Therefore, it is recommended to extend the DAL to include more use cases.

Chapter 9

Project Management

This chapter describes the management process as well as the issues and challenges throughout the project.

9.1 Project Planning

A set of agreements about the way of working and the processes was established between the trainee and the supervisors. Weekly progress meetings were organized in order to discuss the progress of the work and the Sprint planning. Project Steering Group (PSG) meetings were organized on a monthly basis including TU/e supervisor and Océ supervisors. The PSG meetings were planned to discuss the status, the progress and the planning. The project was divided into three parts:

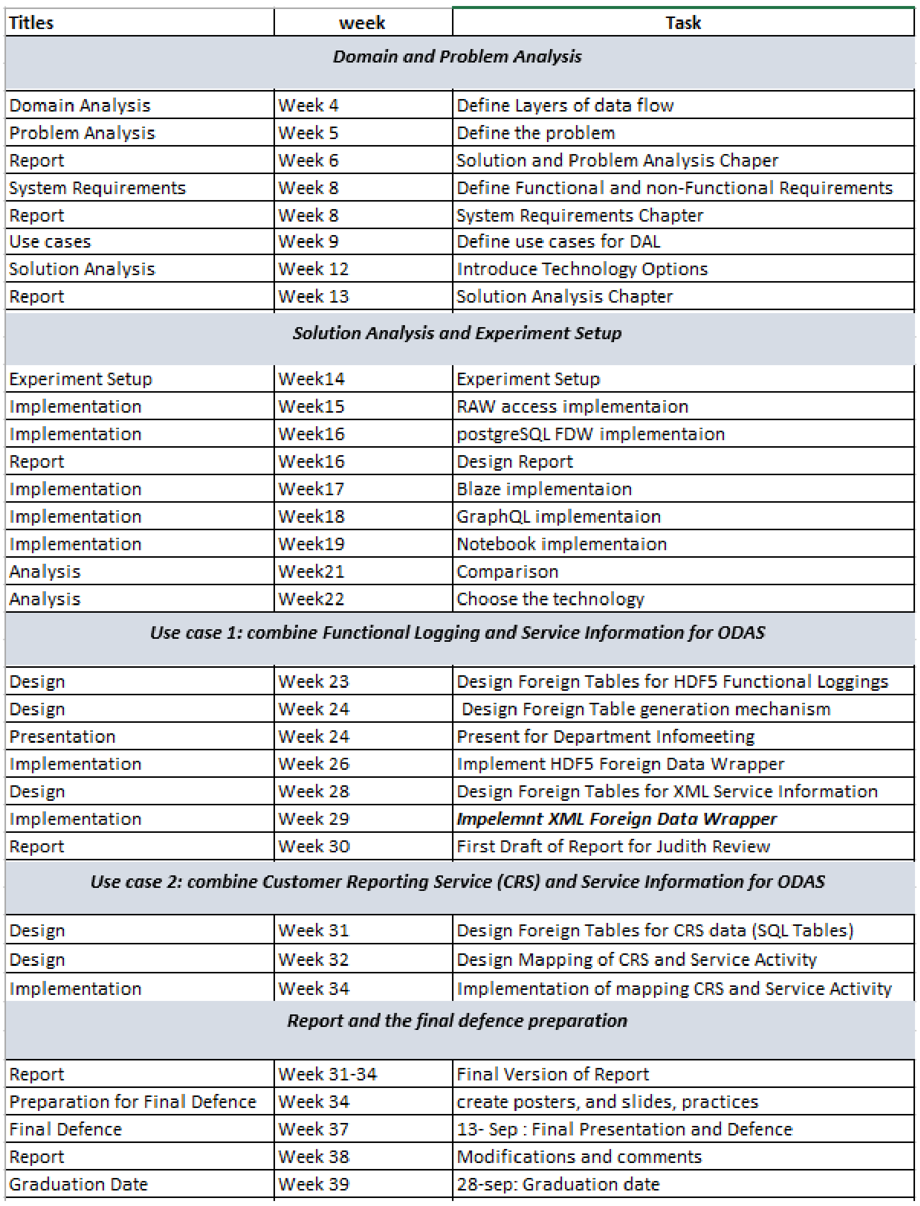
- Domain and problem analysis
  Recognize the different layers of data flow in Océ.
  Understand the difficulties of the current situation and define the desired situation.
  Introduce the different use cases for the DAL in Océ.
  Formalize the requirements for the DAL.
- Solution analysis and experiment setup
  Investigate the technology choices for the DAL.
  Set up an experiment to measure the performance.
  Choose the technology.
- Design and Implementation
  Design the DAL.
  Implement the DAL.
  Validate and Verify the results of the DAL.

The initial project planning devised in the beginning did not match the actual project execution. Several adjustments were introduced along the way, as the knowledge and understanding deepened. In the initial version, three use cases were defined:

- Use case 1: combine Functional Logging and Service Information for ODAS

- Use case 2: combine Customer Reporting Service (CRS) and Service Information for ODAS
- USe case 3: combine Océ Meter Reads in the CRS Report

In the final version, an experiment setup was added to the project plan. This was to have a better insight on each of the technology choices and choosing the correct technology. By this decision, the project plan was revised to include only the first two use cases. Figure 9.1 shows the Final project planning.



Titles	week	Task
Domain and Problem Analysis		
Domain Analysis	Week 4	Define Layers of data flow
Problem Analysis	Week 5	Define the problem
Report	Week 6	Solution and Problem Analysis Chaper
System Requirements	Week 8	Define Functional and non-Functional Requirements
Report	Week 8	System Requirements Chapter
Use cases	Week 9	Define use cases for DAL
Solution Analysis	Week 12	Introduce Technology Options
Report	Week 13	Solution Analysis Chapter
Solution Analysis and Experiment Setup		
Experiment Setup	Week14	Experiment Setup
Implementation	Week15	RAW access implementaion
Implementation	Week16	postgreSQL FDW implementaion
Report	Week16	Design Report
Implementation	Week17	Blaze implementaion
Implementation	Week18	GraphQL implementaion
Implementation	Week19	Notebook implementaion
Analysis	Week21	Comparison
Analysis	Week22	Choose the technology
Use case 1: combine Functional Logging and Service Information for ODAS		
Design	Week 23	Design Foreign Tables for HDF5 Functional Loggings
Design	Week 24	Design Foreign Table generation mechanism
Presentation	Week 24	Present for Department Infomeeting
Implementation	Week 26	Implement HDF5 Foreign Data Wrapper
Design	Week 28	Design Foreign Tables for XML Service Information
Implementation	Week 29	***Impelemnt XML Foreign Data Wrapper***
Report	Week 30	First Draft of Report for Judith Review
Use case 2: combine Customer Reporting Service (CRS) and Service Information for ODAS		
Design	Week 31	Design Foreign Tables for CRS data (SQL Tables)
Design	Week 32	Design Mapping of CRS and Service Activity
Implementation	Week 34	Implementation of mapping CRS and Service Activity
Report and the final defence preparation		
Report	Week 31-34	Final Version of Report
Preparation for Final Defence	Week 34	create posters, and slides, practices
Final Defence	Week 37	13- Sep : Final Presentation and Defence
Report	Week 38	Modifications and comments
Graduation Date	Week 39	28-sep: Graduation date

Figure 9.1: The Final project plan

9.2 Issues

This section describes the issues and challenges that are encountered during the lifetime of the project.

9.2.1 Issues and challenges

Considering different systems and stakeholders, there are different or even conflicting approaches and expectations toward the project. Establishing agreements between stakeholders and defining concrete requirements are challenging activities.

In addition, having access to the data sources for the purpose of understanding the internal structure of the data and developing the data access layer is critical for delivering the end product of this project as it affects the design, modeling, and implementation. Data availability and access is a risk identified and elaborated in the following section.

9.2.2 Risks

Several risks are identified during this project. This section explains the risks identified, their impact on the project, and the corresponding mitigation techniques.

Table 9.1: Risks and mitigation strategies

ID	Description	Impact	Mitigation strategy
R-01	Not having access to data sources for developing and evaluating the data access layer.	Development of general data access layer, which covers less relevant design problems.	Create and use data that satisfies the specifications of real-word data source.
R-02	Not all requirements can be met given the limited time frame of the project	Data access layer would not meet all requirements from the identified stakeholders.	Prioritize the requirements with the agreement of stakeholders. Continuously revise the project plan and reflect on the project progress. Consider the Extensiblity of system for future updates.
R-03	Stakeholders conflicting expectations from the project	Stakeholders dissatisfaction with the result of the project, and depending on the impact of the stakeholder, the impact of this can be high (project failure) or less.	Prioritize stakeholders, regarding their power and interest in the project. Regular meetings with Stakeholders helps to reach agreement.

Chapter 10

Project Retrospective

This chapter finalizes the document by providing a reflection on the project based on the author's perspective.

10.1 Reflection

This project included different aspects and challenges. Before this project, we had an in-house project at Océ. Hence, I became familiar with some parts of the domain which was related to my current project. This increased the speed of domain analysis. In addition, the stakeholders from Océ provided the support and necessary information to complete the domain picture for me. I received constructive feedback from them which helped me to make sure that I am on the right path. This gave me the confidence to continue the project with a strong understanding of the domain and the application of the project in the domain.

The challenging parts include having a good project management process, defining the problem scope, analyzing the solution direction, and translating the requirements into tangible results. In the first months of the project, I faced difficulties regarding project management and planning. Fortunately, I received feedback from Océ stakeholders in the early phases. Having this feedback helped me to focus on my planning skills to define a realistic plan, establish agreements, and confirm the plan with the stakeholders. I learned this rule of thumb: "Revising a plan is not a problem, the problem is to stick to an unrealistic plan".

Since the project was started from scratch, no decision was already made regarding the solution direction. This was challenging to choose the solution direction that satisfies the requirements. Therefore, among different solution directions, an experiment setup was conducted to measure the performance and the other quality attributes. Changing the plan to include the experiment was a wise choice to gain insight on each of the technologies and choosing the proper technology.

Overall, the project was a fulfilling experience. I improved in the design and the technical skills as well as the communication skills. Cooperating with people and managing expectations of stakeholders are crucial and these were repeatedly practiced during the project. On top of that, several new technologies were used that broadened my knowledge in the field of Software Development.

10.2 Design Opportunities Revisited

During the problem analysis process, design opportunities were identified, namely Learnability, Flexibility, and Extensibility. Since the goal of the project was to make a system for users, such as data scientists and function designers, special consideration was taken to ensure that the DAL is easy to use from the interface and query language perspective. The DAL provides a unified interface. Following the iterative development approach and receiving constant feedback, stakeholders were able to evaluate the Learnability, look, and feel of the product. Flexibility of the DAL regarding no limitation for writing different queries were considered in the whole process of design and implementation. The design of the DAL considered the Extensibility of the data sources in terms of evolving data shapes and new data sources. Formally, the use of a survey provided a feedback from a larger group of users. Results of the survey shows that the DAL provides functionality that meets the expectations and user requirements.

Glossary

Term	Definition
Data Access Layer (DAL)	The technology that offers data consumers a unified API for querying data from heterogeneous set of data sources.
Data consumer	Applications that receive data from data sources and use it for analysis and reporting purposes.
Data source	Any source of data, for example, an SQL database, an HDF5 store, or a web service.
Authentication	The process of verifying that a data consumer is the entity that it claims to be for the data source.
Authorization	The process of determining which actions an authenticated data consumer is allowed to perform on the data source. These actions are typically SELECT, INSERT, UPDATE, DELETE from/into the data source. This process also defines whether the data consumers are allowed to see specific parts of the data.
Data joining	The process in which data from different data sources are brought together.
Data filtering	The process in which specific parts of data is retrieved from a data source
Materialized data	Data which is copied from one data source to another data source
Multidimensional Expressions (MDX)	A query language for online analytical processing using a database management system
Concurrent access	A mechanism in which multiple users access the data sources at the same time
Raw access	A direct access from each data consumer to each data source, assuming that there is one individual API for each data source.
Jupyter Notebook	An open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.

Bibliography

[1] CorporatePresentations. (2017) Océ corporate presentation. 2

[2] R. Obe and L. Hsu, *PostgreSQL: Up and Running*, ser. A practical guide to the advanced open source database. O'Reilly, 2012. [Online]. Available: https://books.google.nl/books?id=Q8jkIZkMTPcC 18

[3] ThePostgreSQLDevelopmentGroup. (2017) Create foreign data wrapper. [Online]. Available: https://www.postgresql.org/docs/current/static/sql-createforeigndatawrapper.html 18

[4] J. Helfer. (2017) Tutorial: How to build a graphql server. [Online]. Available: https://dev-blog.apollodata.com/tutorial-building-a-graphql-server-cddaa023c035_ga=2.177939621.188959742.1503063144-760494447.1503063144 19

[5] G. Miller. (2017) Graphql schema definition language. [Online]. Available: https://www.graph.cool/docs/faq/graphql-sdl-schema-definition-language-kr84dktnp0/ 19

[6] ContinuumAnalytics. (2017) The blaze server. [Online]. Available: http://blaze.readthedocs.io/en/latest/server.html 20

[7] C. Analytics. (2017) The blaze expressions. [Online]. Available: http://blaze.readthedocs.io/en/latest/expr-design.html 21

[8] BlazeCommunity. (2017) Have more than one resources on blaze server. [Online]. Available: https://github.com/blaze/blaze/issues/1631 21

[9] JupyterNotebook. (2017) The jupyter notebook. [Online]. Available: http://jupyter-notebook.readthedocs.io/en/latest 21

[10] JupyterNotebookCommunity. (2017) Access to notebook from python client. [Online]. Available: https://github.com/jupyter/help/issues/170 22

[11] R. Wrembel and C. Koncilia, *Data Warehouses and OLAP: Concepts, Architectures, and Solutions*. IRM Press, 2007. [Online]. Available: https://books.google.nl/books?id=XFivorxZDm8C 22

Appendices

Appendix A

Appendix 1: Experiment setup

The queries were defined based on the data filtering and data joining requirements (FR-09): Data consumers can filter data based on conditions and the details of a record:

```
Q1: SELECT * FROM table1 WHERE ullid=49876996433 AND sheetid=6394503;

Q2: SELECT * FROM table2 WHERE ullid=49876883285 AND sheetid=6391745;

Q3: SELECT * FROM table1 WHERE color='K';

Q4: SELECT * FROM table2 WHERE highestpointheight=421;
```

Data consumers can select specific fields of data sources:

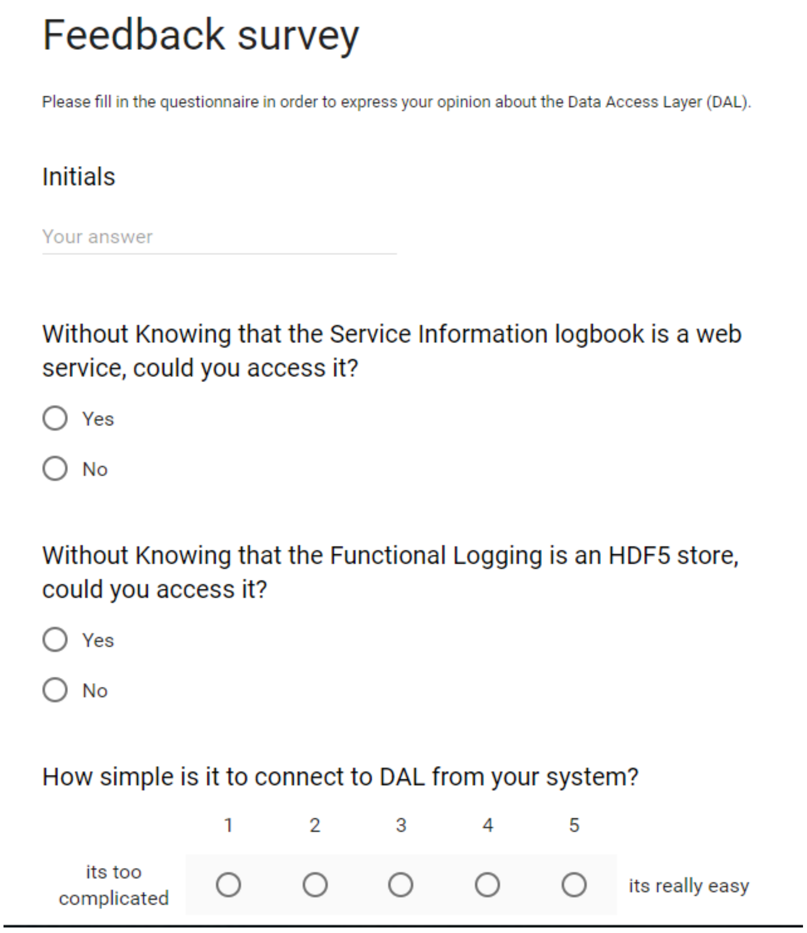
```
Q5: SELECT sheetid FROM table1;

Q6: SELECT sheetid FROM sheetheight;
```

Data consumers are able to join data from different tables in different data sources:

```
Q7: SELECT table1.color FROM table1 ,table2
WHERE table1.ullid = table2.ullid AND table1.headid=2;

Q8: SELECT table2.defectfound FROM table1, table2
WHERE table1.ullid = table2.ullid AND table2.camera='PrintUnitRearCamera';

Q9: SELECT table1.color, table2.defectfound FROM table1, sheetheight
WHERE table1.ullid= table2.ullid;

Q10: SELECT * FROM table1 JOIN table2 USING (ullid);

Q11: SELECT * FROM table1 JOIN table2 USING (ullid)
 WHERE table1.color='K' AND table2.defectfound='yes';

Q12: SELECT * FROM table1 JOIN table2 USING (ullid)
 WHERE table2.camera='PrintUnitRearCamera';
```

Appendix B

Appendix 2: Feedback Survery



Feedback survey

Please fill in the questionnaire in order to express your opinion about the Data Access Layer (DAL).

Initials

Your answer

Without Knowing that the Service Information logbook is a web service, could you access it?

○ Yes

○ No

Without Knowing that the Functional Logging is an HDF5 store, could you access it?

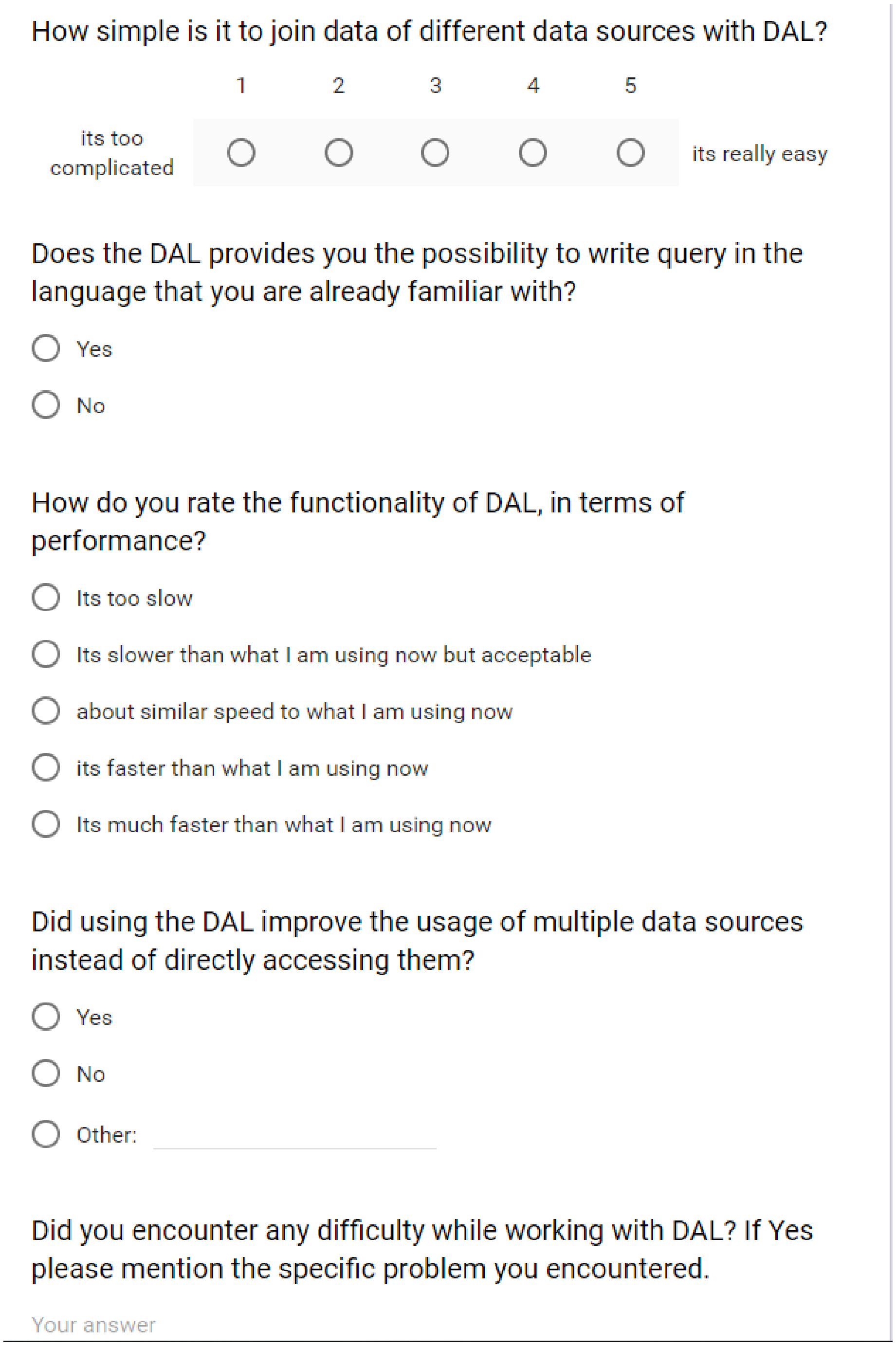
○ Yes

○ No

How simple is it to connect to DAL from your system?

	1	2	3	4	5	
its too complicated	○	○	○	○	○	its really easy

Figure B.1: DAL Feedback Survey



How simple is it to join data of different data sources with DAL?

	1	2	3	4	5	
its too complicated	○	○	○	○	○	its really easy

Does the DAL provides you the possibility to write query in the language that you are already familiar with?

○ Yes

○ No

How do you rate the functionality of DAL, in terms of performance?

○ Its too slow

○ Its slower than what I am using now but acceptable

○ about similar speed to what I am using now

○ its faster than what I am using now

○ Its much faster than what I am using now

Did using the DAL improve the usage of multiple data sources instead of directly accessing them?

○ Yes

○ No

○ Other:

Did you encounter any difficulty while working with DAL? If Yes please mention the specific problem you encountered.

Your answer

Figure B.2: DAL Feedback Survey(Con'd)

About the Author

Fariba Safari received her Bachelor degree in Information Technology Engineering from SHBU University of Isfahan, Iran in 2010. She followed her studies in Information Technology with the focus on advanced information systems at the University of Tarbiat Modares(TMU), Tehran, Iran from 2011-2013. She wrote her Master thesis on *"Developing a Decision Support System for the Adoption of Software-as-a-Service in Small and Medium-sized Enterprises*". From September 2015, she worked at the Eindhoven University of Technology, as PDEng trainee in the Software Technology program from the 3TU.Stan Ackermans Institute. During her graduation project, she worked for Océ on a project focused on the design and engineering of a Data Access Layer (DAL).

Where innovation starts