Architecture for supporting data-driven processes on layer hen production data

Citation for published version (APA):

Document status and date:
Published: 24/10/2018

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.
Architecture for Supporting Data-Driven Processes on Layer Hen Production Data

Wei-Sheng Chen
Architecture for Supporting Data-Driven Processes on Layer Hen Production Data

Wei-Sheng Chen

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

The design described in this report has been carried out in accordance with the TU/e Code of Scientific Conduct.
Hendrix Genetics collects a great variety of data during the day-old chick production chain. The data could be analyzed to generate insights for production improvement. However, the data spread in multiple systems that behave as silos. It makes the data analyses that combine multiple datasets difficult or impossible. The report describes a project to design a system to enable data-driven applications by aggregating, storing, and providing the semi-structured and unstructured data collected during the production chain. A system prototype was implemented based on the design with several Microsoft Azure products. The design and implementation were verified by a data-driven application regarding day-old chick quality, and the result has shown the system could support data-driven applications.

Keywords
data-driven, layer hen, production data

Preferred reference

Partnership
This project was supported by Eindhoven University of Technology and Hendrix Genetics.

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 Hendrix Genetics. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or Hendrix Genetics, and shall not be used for advertising or product endorsement purposes.

Disclaimer Liability
While every effort will be made to ensure that the 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 service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright
Copyright © 2018. 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 Hendrix Genetics.
Foreword

One of the biggest challenges of innovation is to find the persons that are capable to integrate multiple disciplines. For instance, it is possible to find an animal geneticist capable in software technology. Or a software engineer with experience in data analysis. Often, however, the successful execution of an innovative idea requires a person who masters all three disciplines. Conceptually, these unique people operate in the small multidisciplinary space defined by the union of a three-circle Venn diagram (data science, data engineering and animal breeding). Almost impossible to find, these people are therefore called “unicorns”.

Wei-Sheng has worked on improving the quality of the hatching process in our laying hen division. The hatching process has a large impact on the quality of the one-day old chick. Hatching machines and related systems generate what we nowadays refer to as “big data”; endless streams of temperature, humidity, performance and other records. We asked Wei-Sheng to make an effort to link all the data with the objective of generating new insights resulting from previously unseen interactions in the process. These insights should lead to better business decisions.

With proven skills in software design, Wei-Sheng made configuring the Azure data lake and subsequently populating it with our data seem easy. However, where he really exceeded our expectations was the way in which he adopted the layer breeding domain knowledge. I believe that, during the second month of the project, I told him jokingly that he knew more about the subject than a laying hen geneticist. According to him, he had simply been proactive in communicating with experts and stakeholders. At the risk of insulting all software engineers, this is not something that I had expected.

Wei-Sheng has raised eyebrows with his approach. Not only within the layer business unit, but also within our IT department. Wei-Sheng has put the concept of cloud based and data-driven solutions on the map for Hendrix Genetics, largely due to his effort. His work will be the start of wide-spread adoption of this strategy in our company.

Wei-Sheng, thank you very much for your pro-active and pleasant way of working and helping the further development of our company. And congratulations on becoming a much sought after unicorn!

Bram Visser
Geneticist
Hendrix Genetics
October 2018
Preface

The report summarizes the “Architecture for Supporting Data-Driven Processes on Layer Hen Production Data” project carried out by Wei-Sheng Chen as the final part Professional Doctorate in Engineering (PDEng) program in Software Technology, provided by the Eindhoven University of Technology, Stan Ackermans Institute. The project lasted for ten months and was conducted at Hendrix Genetics, Boxmeer.

This document describes a system design and prototype implementation based on the problem, domain, and requirements analysis. Besides, the management process of the project is also elaborated in this document. The audience of this report can be both technical or non-technical readers. However, it is intended for the IT professionals of the company. The table below lists the chapters a reader should read based on different concerns.

<table>
<thead>
<tr>
<th>Concern</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>The motivation and challenges of this project</td>
<td>Chapter 1 Introduction</td>
</tr>
<tr>
<td></td>
<td>Chapter 3 Problem Analysis</td>
</tr>
<tr>
<td></td>
<td>Chapter 4 Domain Analysis</td>
</tr>
<tr>
<td>The usage of the project results</td>
<td>Chapter 6 System Requirements</td>
</tr>
<tr>
<td></td>
<td>Chapter 11 Conclusions</td>
</tr>
<tr>
<td>The technical details of the system</td>
<td>Chapter 6 System Requirements</td>
</tr>
<tr>
<td></td>
<td>Chapter 7 System Design</td>
</tr>
<tr>
<td></td>
<td>Chapter 8 Design and Technology Decisions</td>
</tr>
<tr>
<td></td>
<td>Chapter 10 Verification and Validation</td>
</tr>
<tr>
<td>The project management</td>
<td>Chapter 2 Stakeholder Analysis</td>
</tr>
<tr>
<td></td>
<td>Chapter 5 Feasibility Analysis</td>
</tr>
<tr>
<td></td>
<td>Chapter 12 Project Management</td>
</tr>
<tr>
<td></td>
<td>Chapter 13 Project Retrospective</td>
</tr>
</tbody>
</table>

Wei-Sheng Chen
October 2018
Acknowledgements

I would like to thank everyone who helped or supported me during the awesome ten-month journey. This project is finished successfully because of you.

First, I would like to express my sincere gratitude to Bram Visser, my company supervisor. You helped me a lot, especially when I encountered pitfalls, you guided me to solve those situations. I would also like to thank Jakob de Vlieg, my university supervisor. You referred me to the company and guided me to pay attention to not only the technical aspects of this project but also the business aspects to meet the stakeholders’ expectation. My gratitude also goes to Johan van Arendonk, the chief innovation and technology officer of Hendrix Genetics, without you, I would not have had the opportunity to contribute to the company and the animal breeding domain. And I would like to thank all Hendrix Genetics colleagues who involved in my project. By the way, I am proud that I was one of the RTC team members.

Next, I would like to thank the PDEng ST program team, especially Yanja Dajsuren, Ad Aerts, and Désirée van Oorschot, for the opportunity to be part of the PDEng ST program, the help, and the guidance during the past two years. I would also like to thank all the colleagues from PDEng ST 2016 generation for the awesome memories, and I also learned so much from each of you.

Last but not least, my deepest gratitude goes to my family, especially my wife and my daughter. Thank you all for your support.

Wei-Sheng Chen
October 2018
Executive Summary

The day-old chick production of layer hens is an essential business to Hendrix Genetics, and every improvement in the production can lead to a positive impact on the company. It is believed that data-driven applications, for instance, data analyses, can help identify the improvements in the production.

Currently, a great variety of data is collected during the production chain, and the insights of improvements could be generated based on these data. However, different types of data are collected or generated by different systems that behave as information silos and can be retrieved as files. Some types of data are even collected by people with computer files or paper forms. These issues lead to the difficulty to develop the data-driven applications for indicating improvements in the production. As a result, the technical goal of this project is to develop a system that aggregates and stores the data files collected during the production chain and provides the data to enable data-driven applications.

The domain analysis was carried out to understand the day-old chick production chain, including the process, systems, and collected data. Besides, the overview of the current data-driven processes at the company was also created. During the analysis phase, CAFCR, a framework for architecting a system, was applied to identify the customer needs and the requirements of the system. The requirements were analyzed based on various documents and meetings. The system was designed based on the requirement and illustrated with the “4+1” view model. In order to avoid the customization of standard products in the implementation, some user actions were introduced in the system design to realize the usage of interfaces between two components.

The system consists of the following six components that interact with each other:

- Central Storage: store and provide files.
- File Specification Manager: manage metadata of a data type.
- File Aggregator: put files manually into the system.
- Data Retriever: provide data to data consumers.
- Automated Aggregation Manager: aggregate files automatically.
- Access Controller: control access.

A system prototype was implemented with some products from Microsoft Azure cloud platform and no customization. The design and implementation were verified by a data-driven application regarding day-old chick quality, and the result has shown the system could support data-driven applications.
# Table of Contents

Foreword ......................................................................................................................... v
Preface .............................................................................................................................. vi
Acknowledgements .......................................................................................................... vii
Executive Summary ......................................................................................................... viii
Table of Contents ............................................................................................................ ix
List of Figures ................................................................................................................... xii
List of Tables .................................................................................................................... xiv

Chapter 1 Introduction .................................................................................................... 1
  1.1 Context ....................................................................................................................... 1
  1.2 Background ................................................................................................................ 1
    1.2.1 Hendrix Genetics ............................................................................................... 1
    1.2.2 Production of Layers ....................................................................................... 1
    1.2.3 Similar IT Project ............................................................................................ 2
    1.2.4 IT Strategy ........................................................................................................ 2
  1.3 Outline of Report ...................................................................................................... 2

Chapter 2 Stakeholder Analysis ....................................................................................... 3
  2.1 Hendrix Genetics ..................................................................................................... 3
  2.2 Eindhoven University of Technology ...................................................................... 4

Chapter 3 Problem Analysis ........................................................................................... 5
  3.1 Business Point of View ......................................................................................... 5
  3.2 Technical Point of View ....................................................................................... 5

Chapter 4 Domain Analysis ............................................................................................ 6
  4.1 Day-Old Chick Production .................................................................................... 6
    4.1.1 Production Process .......................................................................................... 6
    4.1.2 Data collection and retrieval approaches ......................................................... 8
  4.2 Data-Driven Processes in the Company ................................................................ 8
    4.2.1 Data Collection ............................................................................................... 9
    4.2.2 Data Analysis ................................................................................................ 9

Chapter 5 Feasibility Analysis ......................................................................................... 10
  5.1 Issues and challenges ............................................................................................ 10
  5.2 Risks ....................................................................................................................... 10

Chapter 6 System Requirements ...................................................................................... 12
  6.1 Introduction .............................................................................................................. 12
  6.2 Scope ....................................................................................................................... 12
  6.3 Requirement Sources ............................................................................................ 13
  6.4 Customer objectives view and Application view ............................................... 13
  6.5 Functional view ..................................................................................................... 14
6.5.1 Use cases ............................................................................................................. 14
6.5.2 Requirements ..................................................................................................... 16
Chapter 7 System Design .............................................................................................. 18
  7.1 Design Process ....................................................................................................... 18
  7.2 Development View ................................................................................................. 19
    7.2.1 Central Storage ................................................................................................. 24
    7.2.2 File Specification Manager ............................................................................... 24
    7.2.3 File Aggregator ............................................................................................... 24
    7.2.4 Data Retriever ................................................................................................. 24
    7.2.5 Automated Aggregation Manager ..................................................................... 24
    7.2.6 Access Controller ............................................................................................ 24
    7.2.7 Minimum Application Prototype ....................................................................... 24
  7.3 Logical View ........................................................................................................... 25
  7.4 Process View .......................................................................................................... 26
    7.4.1 Create file specification .................................................................................... 28
    7.4.2 Create Automated Aggregation ......................................................................... 29
    7.4.3 Aggregate files ................................................................................................. 30
    7.4.4 Retrieve data .................................................................................................... 31
    7.4.5 Set Permissions ............................................................................................... 32
    7.4.6 Versioning Data Types ..................................................................................... 32
  7.5 Physical View .......................................................................................................... 34
Chapter 8 Design and Technology Decisions ................................................................ 35
  8.1 Directory structure ................................................................................................. 35
  8.2 File aggregation versioning .................................................................................... 36
    8.2.1 First approach .................................................................................................. 37
    8.2.2 Second approach ............................................................................................. 38
    8.2.3 Third approach ................................................................................................ 39
  8.3 Central Storage implementation ............................................................................. 40
    8.3.1 Candidate products selection .......................................................................... 40
    8.3.2 Candidate products comparison ...................................................................... 40
    8.3.3 Technology decision for the Central Storage implementation ........................... 42
Chapter 9 Implementation ............................................................................................ 43
  9.1 Introduction ............................................................................................................. 43
  9.2 Implementation of the Production Data Platform ..................................................... 43
    9.2.1 Access Controller ............................................................................................. 46
    9.2.2 Central Storage ............................................................................................... 47
    9.2.3 File Aggregator ............................................................................................... 48
    9.2.4 Data Retriever ................................................................................................. 49
    9.2.5 File Specification Manager .............................................................................. 49
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Hybridization and generations of layers</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Process of DOC production chain</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3</td>
<td>CAFCR model [3]</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Requirement sources</td>
<td>13</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Use cases of the Production Data Platform</td>
<td>15</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Design process</td>
<td>18</td>
</tr>
<tr>
<td>Figure 7</td>
<td>The “4+1” view model [4]</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Development view of the Production Data Platform</td>
<td>20</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Development view of the data-driven application prototype</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Traceability of components – functional requirements</td>
<td>22</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Traceability of components – non-functional requirements</td>
<td>23</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Logical view of the key data of the Production Data Platform</td>
<td>25</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Process view of the Production Data Platform</td>
<td>27</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Process view of the Production Data Platform – detailed</td>
<td>28</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Sequence of creating file specification</td>
<td>29</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Sequence of creating automated aggregation</td>
<td>30</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Sequence of aggregating files</td>
<td>31</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Sequence of retrieving data</td>
<td>32</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Sequence of setting permissions</td>
<td>32</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Flow of versioning data types</td>
<td>33</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Physical view of the Production Data Platform</td>
<td>34</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Directory structure of the Central Storage</td>
<td>35</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Directory structure of the first versioning approach</td>
<td>37</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Directory structure of the second versioning approach</td>
<td>38</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Directory structure of the third versioning approach</td>
<td>39</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Implementation of components</td>
<td>44</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Deployment of the prototype implementation</td>
<td>45</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Azure Portal</td>
<td>46</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Access Controller implementation with Azure Active Directory</td>
<td>47</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Central Storage implementation with Azure Data Lake Store</td>
<td>48</td>
</tr>
<tr>
<td>Figure 31</td>
<td>File Aggregator implementation with Microsoft Azure Storage Explorer</td>
<td>49</td>
</tr>
<tr>
<td>Figure 32</td>
<td>File Specification Manager implementation with Azure Data Catalog</td>
<td>50</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Automated Aggregation Manager implementation with Azure Data Factory</td>
<td>51</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Get data from ADLS by Power BI</td>
<td>51</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Implementation of DOC quality application prototype</td>
<td>52</td>
</tr>
<tr>
<td>Figure 36</td>
<td>WBS of this project</td>
<td>57</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Final project plan</td>
<td>58</td>
</tr>
</tbody>
</table>
Figure 38 Meeting plan for the design and implementation phases ........................................ 59
Figure 39 Requirements analysis result – part one .................................................................. 63
Figure 40 Requirements analysis result – part two .................................................................. 64
List of Tables

Table 1 Hendrix Genetics business stakeholder analysis ................................................................. 3
Table 2 Hendrix Genetics technical stakeholder analysis ................................................................. 3
Table 3 TU/e stakeholder analysis ................................................................................................... 4
Table 4 Data retrieval approaches of production systems ................................................................. 8
Table 5 Risk management plan ......................................................................................................... 11
Table 6 Customer objectives view items ......................................................................................... 13
Table 7 Application view items ........................................................................................................ 14
Table 8 Functional requirements ..................................................................................................... 16
Table 9 Non-functional requirements ............................................................................................... 16
Table 10 Comparison of file aggregation versioning approaches ................................................... 37
Table 11 Comparison of Central Storage candidates ....................................................................... 40
Table 12 Datasets for DOC quality application development ......................................................... 52
Table 13 Validation of system requirements ..................................................................................... 53
Chapter 1 Introduction

This chapter describes the context of this project and the relevant background. In addition, the outline of this report is mentioned briefly at the end of this chapter.

1.1 Context

This project was initiated by Hendrix Genetics, an international multi-species animal breeding company. It was offered as a graduation project to Wei-Sheng Chen, a Professional Doctorate in Engineering Software Technology\(^1\) trainee. The project goal is to enable the identification of improvements and quality prediction for layer hen production. Before this project, there was already another similar project that focused on the structured business data in the Information Technology (IT) plan. The similar project was later conducted parallelly at the same time with this project. Therefore, the focus of this project was scoped to the infrastructure to enable the analysis of semi-structured and unstructured data collected during production, especially the layer production chain.

1.2 Background
1.2.1 Hendrix Genetics

Hendrix Genetics\(^2\) is an international multi-species animal breeding company that is headquartered in Boxmeer in the Netherlands. Its vision is to deliver better breeding that leads to value in animal protein value chain, for example, reducing footprint, improving animal welfare, and increasing efficiency.

The goal of animal breeding is producing improved offspring [1]. Genes determine many of the desirable qualities of an animal. Animal breeding is a circular process of recording animal performance (phenotypes), estimating the effect of genes, and selecting animals as parents to produce the next generation. Nowadays genomics is used to enhance the animal breeding process further.

1.2.2 Production of Layers

Hendrix Genetics is one of the world’s leading breeders and distributors of white and brown layers. A series of processes is carried out before layer hen can be shipped to the customer. This project aims to provide the infrastructure to gain insights from the data collected during the day-old chick (DOC) production process to identify potential improvements for the business. However, the outcome of this project is expected to apply to other processes and species in the future.

In layer production, there are four main generations of layers: pure line (PL), grandparent stock (GPS), parent stock (PS), and commercial stock (CS). The relations between these four generations are shown in Figure 1. PL generation is the starting point of layer production. The sire and dam from the same PL produce the GPS generation. Then, the process is followed by the PS generation production with the GPS sire and dame from different lines. Finally, the PS sire and dam produce the CS generation, which produces consumer eggs.

\(^1\) https://tue.nl/softwaretechnology
\(^2\) https://www.hendrix-genetics.com/
1.2.3 Similar IT Project

Before this project started, there had been another similar project planned. Its project goal is also to build the infrastructure to gain insights from the data collected in the company. However, the data that the similar project focused on is the structured data from business systems such as the enterprise resource planning (ERP) and financial systems. As a result, the scope of this project was changed to focus on only the semi-structured and unstructured data collected during production chain, called “production data.”

Because the similar project’s context is about the Business Intelligence (BI) infrastructure, it is called “ongoing BI project” in this technical report.

1.2.4 IT Strategy

In addition to the similar IT project, “cloud first” was defined as one of the core IT strategies in the company. The main idea of “cloud first” is to adopt public cloud platforms for hosting the systems used in the company, which means the existing systems will be migrated to a public cloud and new systems must be based on a public cloud as well. For the systems that are not directly related to the animal breeding process, Microsoft Azure was chosen as the preference of the public cloud since the current IT infrastructure relies mainly on Microsoft solutions, and Microsoft Azure provides excellent compatibility to this kind of IT infrastructure.

1.3 Outline of Report

The next four chapters elaborate different analyses to initiate the project. Chapter 2 describes the identified stakeholders and their interests in the project. In Chapter 3, the analysis of the problem is elaborated in the business and the technical point of views. The study of the relevant processes in the company is described in Chapter 4. Chapter 5 explains the issues, challenges, and risks during the project period.

The system analysis, design, and implementation are described in Chapter 6, Chapter 7, and Chapter 9 respectively. Chapter 8 elaborates the design and technology decisions made during the design and implementation phases. The verification and validation of the system are described in Chapter 10. Chapter 11 gives the conclusion of the project. The project management of this project is addressed in Chapter 12. Chapter 13 finalizes the report with a reflection from the author’s point of view.

---

3 https://azure.microsoft.com/
Chapter 2 Stakeholder Analysis

This chapter provides the summary of the stakeholder analysis throughout the project. A stakeholder is “a person, group or organization that has interests in, or can affect, be affected by, or perceive itself to be affected by, any aspect of the project [2].” The main stakeholders of this project are mostly from Hendrix Genetics since the project is about the company’s business and IT infrastructure. Another group of main stakeholders is from Eindhoven University of Technology (TU/e) because the project is a collaboration between TU/e and the company.

2.1 Hendrix Genetics

Hendrix Genetics is an animal breeding company, and this project focuses on the data-driven architecture to enable the data-driven applications about layer production. The main interests of the stakeholders are how the outcome of the project could help the business and the data analysis results. At the beginning of this project, the focus was the preliminary data analysis of production data, so the production director and quality control manager were identified as the most important stakeholders. However, as the project progressed, the focus changed to the data-driven architecture to enable data-driven applications, so the IT manager, architect, and data analyst became the most important stakeholders.

In summary, the stakeholders from the company can be categorized into two main groups: business stakeholders and technical stakeholders. The business stakeholders care about the benefits brought by the outcome of this project to the business, and the technical stakeholders mainly care about the technical impacts from the outcome of this project. Table 1 and Table 2 show the analysis of business and technical stakeholders respectively.

Table 1 Hendrix Genetics business stakeholder analysis

<table>
<thead>
<tr>
<th>Role</th>
<th>Representative</th>
<th>Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief Technology &amp; Innovation Officer</td>
<td>Johan van Arendonk</td>
<td>• Apply data-driven technology to identify the improvements for DOC production.</td>
</tr>
<tr>
<td>Company Supervisor</td>
<td>Bram Visser</td>
<td>• The project results help the development of the company.</td>
</tr>
<tr>
<td>Production Director</td>
<td>Ron Jöerissen</td>
<td>• The data analysis results based on the production data.</td>
</tr>
<tr>
<td>Quality Manager</td>
<td>Dorothé Ducro</td>
<td>• The data analysis results based on the production data.</td>
</tr>
<tr>
<td>Data Analyst</td>
<td>Bram Visser</td>
<td>• The outcome helps the data analyses of production data.</td>
</tr>
</tbody>
</table>

Table 2 Hendrix Genetics technical stakeholder analysis

<table>
<thead>
<tr>
<th>Role</th>
<th>Representative</th>
<th>Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT Manager</td>
<td>Ryan Rokven</td>
<td>• The outcome fits the IT strategy and plan.</td>
</tr>
<tr>
<td>IT Architect</td>
<td>Eric van den Acker</td>
<td>• The outcome fits the IT strategy and plan.</td>
</tr>
</tbody>
</table>
2.2 Eindhoven University of Technology

Eindhoven University of Technology (TU/e) offers Professional Doctorate in Engineering (PDEng) degrees, and the Software Technology (ST) is one of the tracks. The PDEng ST program is a two-year post-master program to train software designers within an industrial context. A software designer has the practical knowledge of software development lifecycle and can help industries regarding software engineering. This project was offered as a graduation project to a PDEng ST trainee, and one TU/e supervisor was assigned to the trainee. The main stakeholders from TU/e and their interests are shown in Table 3.

Table 3 TU/e stakeholder analysis

<table>
<thead>
<tr>
<th>Role</th>
<th>Representative</th>
<th>Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDEng Trainee</td>
<td>Wei-Sheng Chen</td>
<td>• Conduct the project successfully and on time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The project results help the company.</td>
</tr>
<tr>
<td>TU/e Supervisor</td>
<td>Jakob de Vlieg</td>
<td>• Deliver data-driven knowledge and resources to the company.</td>
</tr>
<tr>
<td>PDEng ST Program Director</td>
<td>Yanja Dajsuren</td>
<td>• The quality of the project process and results.</td>
</tr>
</tbody>
</table>
Chapter 3 Problem Analysis

This chapter describes the problem that this project aims to solve. After the initial stakeholder analysis was conducted, the problem of this project was analyzed to make the focus of this project clearer.

3.1 Business Point of View

The business units of the company keep improving themselves to have a better outcome. The layer business unit, the largest BU of the company, is one of them. One of the goals that they work to achieve is to improve the production process to reduce DOC mortality. The improvement will not only lower the unnecessary expense but also higher purchasing value for customers. Eventually, it results in the willingness of the customers to purchase again.

Since a great variety of data is collected during the whole production chain, the layer BU wants to improve the production by analyzing the data and generating insights into improvement. In the future, it is expected to go even further: predicting the product quality. Therefore, identifying improvements is the ultimate business goal of this project. However, the current environment cannot support this goal because of the technical limitations.

3.2 Technical Point of View

From the technical perspective, the distribution of various data leads to difficulties in conducting data analysis. The whole DOC production chain contains multiple steps, and each step involves one or more types of data collection. Different types of data are collected by different systems, and the systems work as information silos. Moreover, some types of data are even not collected systematically. These realities make the goal of this project difficult to be achieved. Hence, the main technical problem that the company faces is a lack of the infrastructure to aggregate different types of data across the whole production chain, which is needed to enable data-driven applications. Hence, the project focuses on the architecture for aggregating various types of production data.
Chapter 4 Domain Analysis

This chapter describes the modeling of the topics that are relevant to this project. The domain analysis focuses on the data-driven processes at the company: how the data is collected and how the data analysts analyze it.

4.1 Day-Old Chick Production

The ultimate business goal of this project is to identify improvements to the DOC production chain, and the technical goal is to make the aggregation of production data collected during the DOC production chain possible. Hence, having an overview of the DOC production chain was necessary, and the essential part of this domain analysis is the understanding of the process, the data collection approaches, and data retrieval approaches.

In the whole layer DOC production process, the breeding of layer pure lines (PL) and grandparent stocks (GPS) is related to animal selection and mating, and it is not directly related to the parent stock (PS) DOC qualities that we wanted to know in this project. We wanted to know if the process involved in the production chain affects the DOC quality instead of the effects of the performance and genes of parents. Therefore, the DOC production chain focused in this project is defined with the production director as the process from rearing GPS until finishing the delivery of PS DOCs to customers. Besides, the context of this domain analysis was the DOC production in the Netherlands.

4.1.1 Production Process

The process of production chain contains the following activities (as shown in Figure 2):

1. GPS rearing farmers rear GPS DOCs until the DOCs become pullets.
2. GPS pullets are transported from the GPS rearing farm to a GPS production farm.
3. The GPS birds produce PS hatching eggs in the GPS production farm.
4. PS hatching eggs are transported to a PS hatchery.
5. The PS hatchery receives the PS hatching eggs, and the eggs can be stored in the storage rooms at the hatchery.
6. The PS hatching eggs are incubated in the PS hatchery. The incubation process consists of two periods: setting and hatching. The setting period is from the first day to incubate eggs to around the eighteenth day. After setting, the eggs will be candled to see if the egg is fertile and an embryo growing. Then, the eggs will be transferred to continue the hatching period until the twenty-first day to hatch.
7. After the PS eggs are hatched, the PS DOCs are given required treatments in the PS hatchery.
8. The PS DOCs are packed in the PS hatchery.
9. The packed PS DOCs are stored and prepared for transportation in the PS hatchery.
10. The packed PS DOCs are transported to a customer.
11. The customer sends back the 7-day mortality information to a sales team member.
Figure 2 Process of DOC production chain
4.1.2 Data collection and retrieval approaches

The data generated during the DOC production is mostly collected by systems automatically. The brief descriptions of the systems are the following, and the relations with the production chain activities are shown in Figure 2:

- **Prinzen Egg Setting System**: the system used to filter out the eggs that do not meet the quality of hatching eggs.
- **Pas Reform SmartCenter**: the system used to control the setters (for setting periods) and hatchers (for hatching periods) of eggs.
- **iD Projects Candling System**: the system used to remove infertile eggs or the eggs with early mortality by candling them.
- **Innovatec Sexing Lines**: the system used to collect DOC sexing statistics data.
- **Priva TC Manager**: the system used to control the infrastructure of the hatchery.
- **Sensitech Temperature Recorder**: the system used to record the temperature during the transportation from the hatchery to customers.

The systems mentioned above all work as silos, and they do not provide the built-in application programming interfaces (API) for automating the process to retrieve data out for aggregation. The approaches to retrieve data from the systems can be categorized into two types. The first type of approaches is to copy files through the physical interface on the machines. And the second type is to export data as files through the web-based user interface (UI) and download them. The data retrieval approaches of the systems are listed in Table 4.

<table>
<thead>
<tr>
<th>System</th>
<th>Data retrieval approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prinzen Egg Setting Machine</td>
<td>Copy files from the machine to a USB flash drive</td>
</tr>
<tr>
<td>Pas Reform SmartCenter</td>
<td>Export data and download as files through the web-based user interface</td>
</tr>
<tr>
<td>iD Projects Candling Machine</td>
<td>Copy files from the FTP (File Transfer Protocol) site on the machine through a private network</td>
</tr>
<tr>
<td>Innovatec Sexing Lines</td>
<td>Copy files from the machine to a USB flash drive</td>
</tr>
<tr>
<td>Priva TC Manager</td>
<td>Export data and download as files through the web-based user interface</td>
</tr>
<tr>
<td>Sensitech Temperature Recorder</td>
<td>Export data and download as files through ColdStream Portal, a web-based user interface</td>
</tr>
</tbody>
</table>

Besides, **Excel**, the spreadsheet tool from Microsoft, is also commonly used to collect data by people during the DOC production chain. It means that some data exists as files, which are semi-structured data.

In addition to the systems and tools mentioned above, some production statistics data is also stored in **MTech**, the ERP system of the company, and one of the data sources in another ongoing BI project.

4.2 Data-Driven Processes in the Company

Because the technical goal of this project is to design an architecture for aggregating various types of production data, the current way of conducting data-driven analysis needed to be investigated. In
summary, data collection and data analysis are the two main activities of the data-driven process in the company. The essential part of this domain analysis is the understanding of the categories of data, the users of the data, and the data analytics tools used in the company.

4.2.1 Data Collection

During the whole process related to DOC production, the data collected can be categorized into three main types: animal breeding data, production data, and business data.

Animal breeding data is the genotype and phenotype data collected before the production of final products, day-old chicks. The type of data is collected by people with various equipment and in-house-built FleXYbreed and FleXYtool systems.

Production data is the data collected during the production process of final products, which are the PS DOCs in this project context. This type of data is mainly collected by different systems automatically, and most of the systems are in a hatchery. In addition to the systems, people also use Excel or paper forms to collect data and store them on SharePoint, a web-based team collaboration tool. Among these data, some critical statistics data is stored in MTech, the ERP system of the company.

Business data includes sales and financial data, for example, the data about orders, shipments, and payments. The main systems used for these types of data collection are MTech and AX, another ERP system provided by Microsoft.

4.2.2 Data Analysis

The users regarding data analysis can be categorized based on the three main types of data: users who focus on animal breeding data, production data, and business data respectively. Different types of users use different tools to analyze the collected data, and the same type of users from different branches even use different tools. The data analytics tools include:

- **Excel**, a spreadsheet tool by Microsoft
- **Oracle BI**, a visual analytics tool by Oracle
- **QlikView**, a business discovery platform by Qlik
- **MTech Report**, a built-in reporting function from MTech by MTech Systems
- In-house built reporting tools
- **R**, a free software environment for statistical computing and graphics

Nowadays each branch has the freedom to choose the data analytics tool to use, for instance, QlikView or Oracle BI. In the current IT plan, it is going to adopt Power BI, a suite of business analytics tools by Microsoft, as the recommended interactive data analytics tool for the whole company.

In the context of this project, the Excel, MTech Report, and the in-house built reporting tool are used by production team members and quality control experts. In addition, R&D team members use R as the primary tool to analyze data.
Chapter 5 Feasibility Analysis

After the problem analysis and the domain analysis, a feasibility analysis was carried out over time to identify potential issues, challenges, and risks. This chapter describes the issues, challenges, and risks.

Several aspects were taken into account to analyze the feasibility:

- **Financial:** the cost for building the system prototype could be relatively low since cloud-based products usually use the pay-as-you-go plan for billing cost.
- **Technical:** cloud platforms provide a great variety of products for storages, computing, and many other purposes, so applying the “cloud first” strategy to the system would be possible.
- **Organizational:** the company has DOC production related facilities and teams in the Netherlands, so it should lower the effort to communicate and get the help for the project. Besides, it is not difficult to contact different stakeholders within the company.
- **Schedule:** even though the project period is limited, the project could be carried out with an agile-like approach to make sure that the project results fit the expectation.

However, there were still some issues and challenges emerging over the project.

5.1 Issues and challenges

There were several issues and challenges throughout the project. The first challenge was defining the project scope because of the broad project goal. At the beginning of the project, it was an issue that other related projects were not taken into account to define the scope. Later, the IT context in the company was investigated, and another similar project, which is the ongoing BI project, was noticed. Finally, the project scope was defined with the goal and the scope of the similar project.

Another challenge was about acquiring the right requirements. There were several reasons for it. First, there was no system playing a similar role in the company. Second, the stakeholders knew only what they want to achieve in the business point of view. Besides, there were not all types of users of this system existing in the company. Therefore, it was difficult to get the right requirements for the system development. A part of the analyses and design was based on some assumptions about the usage of the system. The assumptions were confirmed with the stakeholders.

This project focused on the integration of the systems collecting data during the DOC production process as data sources, so the number and diversity of those systems also created a challenge for this project. Later, because the focus of this project was changed to the proof of concept of a system to contain and expose files for analyzing, the challenge did not exist anymore. However, it introduced another challenge: having a good showcase of the proof of concept. The potential users of data-driven applications supported by this system did not know how they could do with the system to get benefits for them.

5.2 Risks

Several potential risks were identified throughout the whole project. Some risks identified were about the amount and the quality of the data since the project originally focused on the data analysis regarding DOC quality. After finalizing the project scope definition, the risks regarding data became not applicable. The risks identified are listed in Table 5.
Table 5 Risk management plan

<table>
<thead>
<tr>
<th>ID</th>
<th>Risk Impact</th>
<th>Probability of Occurrence</th>
<th>Risk Description</th>
<th>Project Impact</th>
<th>Risk Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>High</td>
<td>High</td>
<td>Lack of the knowledge of required technologies</td>
<td>Slows down research and development process.</td>
<td>Find online resources, ask for external resources if possible and needed</td>
</tr>
<tr>
<td>R2</td>
<td>High</td>
<td>Medium</td>
<td>Stakeholder is unable to review products</td>
<td>Incorrect direction</td>
<td>Early-scheduled appointments, find the substitute of the stakeholder, find a comfortable way for stakeholder to review</td>
</tr>
<tr>
<td>R3</td>
<td>High</td>
<td>Low</td>
<td>Miscommunication with stakeholders</td>
<td>Misunderstanding of project goals, delayed, inefficient, or incorrect development</td>
<td>Contact stakeholders frequently, ask for clarification, confirm with stakeholders, emails for essential messages and decisions</td>
</tr>
<tr>
<td>R4</td>
<td>Medium</td>
<td>Low</td>
<td>Key stakeholders are not available due to sickness, holidays, or other reasons</td>
<td>Slows down research and development process. Makes main source of information unavailable</td>
<td>Ask stakeholders for planned leave. Find another source of information.</td>
</tr>
<tr>
<td>R5</td>
<td>High</td>
<td>Low</td>
<td>Only interview the roles involved in a specific step of DOC production chain</td>
<td>A lack of comprehensive picture of production chain and lead to incorrect outcome</td>
<td>Interview also the role taking care of whole production chain</td>
</tr>
<tr>
<td>R6</td>
<td>Medium</td>
<td>Low</td>
<td>Production system doesn’t provide any built-in functionality to export data</td>
<td>Not possible to come up with an insightful result by incorporating the dataset into the analysis</td>
<td>Inspect the database schema and retrieve data directly from the database</td>
</tr>
<tr>
<td>R7</td>
<td>High</td>
<td>Low</td>
<td>Limited knowledge about current IT architecture and plan</td>
<td>Design is incompatible with IT architecture and plan</td>
<td>Study current IT architecture and plan, discuss with IT experts</td>
</tr>
<tr>
<td>R8</td>
<td>Medium</td>
<td>Medium</td>
<td>Too many missing values in essential attributes of a dataset</td>
<td>Not possible to come up with an insightful result or not possible to analyze the dataset</td>
<td>Get same type of data from one partner of Hendrix Genetics, change the scope of the data-driven application</td>
</tr>
<tr>
<td>R9</td>
<td>Medium</td>
<td>Medium</td>
<td>Essential data is not collected by a system</td>
<td>Slows down research and development process due to extra efforts on preprocessing data</td>
<td>Leave out the data if possible, change the scope of the data-driven application</td>
</tr>
<tr>
<td>R10</td>
<td>Medium</td>
<td>Medium</td>
<td>Lack of the knowledge of data analysis</td>
<td>Inappropriate data-driven approaches for developing the application</td>
<td>Study data-driven approaches, consult with experts</td>
</tr>
<tr>
<td>R11</td>
<td>Low</td>
<td>High</td>
<td>Production system doesn’t have any non-UI interface to provide data</td>
<td>Unable or difficult to automate data extraction process of a system</td>
<td>Leave out the system if possible, change the scope of the data-driven application</td>
</tr>
<tr>
<td>R12</td>
<td>Medium</td>
<td>Low</td>
<td>Unexpected important event occurs</td>
<td>Unable to deliver products based on the schedule</td>
<td>Divide the products with smaller granularity and finish highest priorities, discuss the prioritization with stakeholders</td>
</tr>
<tr>
<td>R13</td>
<td>High</td>
<td>Medium</td>
<td>Requirement with a high priority cannot be satisfied</td>
<td>Incomplete system design</td>
<td>Signal the situation, discuss possible workaround with stakeholders, manage expectation</td>
</tr>
</tbody>
</table>
Chapter 6 System Requirements

This chapter describes the requirements analysis of the main product of this project. After the analyses of the problem, the domain, and the feasibility, the scope and the focus were defined. Then, the requirements analysis was carried out.

6.1 Introduction

The aim of this project is the architecture for enabling future data-driven application on production data. In this project, the CAFCR model [3] was adopted to architect the main product. CAFCR contains five views related to each other: Customer objectives view, Application view, Functional view, Conceptual view, and Realization view (as shown in Figure 3). The architecting process based on the CAFCR model starts with the customer objectives view to know what the customer needs in the product. Then, the process is followed by the application view to know in what ways the customer wants to achieve the objectives. The functional view results from the application view, and it states what the product is, including the use cases, functional, and non-functional requirements. The conceptual view and realization view are both about how to realize the product.

![Diagram of CAFCR model]

Figure 3 CAFCR model [3]

The requirements analysis of this project was done with the CAF views, and the complete analysis result is shown in Figure 39 and Figure 40. Later subsections describe the CAF views of the main product respectively as well as the source of the customer objectives.

6.2 Scope

The final scope of this project was affected by another ongoing BI project. The main idea of the BI project is to apply a data warehouse to store the business data and conduct data analysis with Power BI. The business data is the structured data from the ERP system and the financial system. Therefore, it was decided to make this project focus on the infrastructure to enable the data analysis regarding production data, especially the semi-structured and unstructured data. In addition, a DOC quality application prototype based on production data is also needed to demonstrate the infrastructure since this project is a proof of concept.
6.3 Requirement Sources

The sources used to analyze requirements can be categorized into three types: document, meeting, and event. “Meeting” type means formal and informal discussions, and the “event” type means other events except meeting type events. For example, the period of the preliminary analysis is categorized as an event. The overview of the sources used in the analysis is shown in Figure 4.

![Figure 4 Requirement sources]

6.4 Customer objectives view and Application view

Based on the requirement sources, the customer objectives were extracted. Each stakeholder had different concerns, which were the objectives. However, some of stakeholders shared the same objectives. Table 6 shows the list of extracted customer objectives view items.

<table>
<thead>
<tr>
<th>Name</th>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables the analysis and monitoring of quality along the DOC production chain</td>
<td>CTO</td>
</tr>
<tr>
<td>Focus on semi-structured and unstructured production data</td>
<td>IT Manager</td>
</tr>
<tr>
<td>The product as a proof of concept</td>
<td>IT Manager</td>
</tr>
<tr>
<td>Cloud first strategy</td>
<td>IT Manager</td>
</tr>
<tr>
<td>Avoid duplicate data from a same data source</td>
<td>IT Manager and IT Architect</td>
</tr>
<tr>
<td>Security</td>
<td>IT Architect</td>
</tr>
</tbody>
</table>
Avoid customization of standard products | IT Architect
Avoid affecting existing IT infrastructure | IT Architect
Data availability | IT Architect
Interoperability with Power BI | Business Analyst
Interoperability with R | Data Analyst
Analyze large amount of data efficiently | Data Analyst
Relation between Hatchability of Transferred (HOT) and mortality | Production Director and Quality Control Manager
Relation between HOT and outside climate data | Production Director and Quality Control Manager
Relation between the duration of transit time and mortality | Production Director and Quality Control Manager

Table 7 Application view items

<table>
<thead>
<tr>
<th>Name</th>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store production data in a central storage</td>
<td>Business Analyst and Data Analyst</td>
</tr>
<tr>
<td>Retrieve various production data from a central storage</td>
<td>Business Analyst and Data Analyst</td>
</tr>
<tr>
<td>Demonstrate the product with a minimum application prototype</td>
<td>IT Manager</td>
</tr>
<tr>
<td>Cloud Platform</td>
<td>IT Manager</td>
</tr>
<tr>
<td>Single source of truth</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Access Control</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Encrypt data if necessary</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Standard Product without Customization</td>
<td>IT Architect</td>
</tr>
<tr>
<td>License Checking</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Data Replication</td>
<td>IT Architect</td>
</tr>
<tr>
<td>High Availability</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Minimum application prototype</td>
<td>IT Manager, Production Director, and Quality Control Manager</td>
</tr>
</tbody>
</table>

Then the customer objectives led to what applications that the stakeholders wanted to adopt to achieve the objectives. Table 7 shows the list of derived application items.

6.5 Functional view

In the function view, the use cases and relevant requirements were modeled based on the items from the application view. The system for fulfilling the stakeholders’ concerns was proposed and called “Production Data Platform (PDP)”.

6.5.1 Use cases

Based on the customer objectives and applications, there were two main use cases concluded: UC1 – Aggregate data files and UC2 – Retrieve data. The third use case, UC3 – Set permission, was introduced because it was also an essential part of the system even though the main functionality was not about security.

The actors of the system and their responsibilities were modeled as well. The responsibilities were modeled with the knowledge from domain analysis and the understanding of data-driven approaches. The identified actors of the system are the following:
• Data Engineer (DE): a person who sets up the required things for file aggregations and manage files manually in the PDP (if necessary).
• File Provider (FP): a person or a system that provides files to be aggregated by the PDP. File Providers only know the intermediate storage (called “staging store”) to put the files, and they do not have direct interactions with the PDP.
• Data Analyst (DA): a person who analyzes data with a software tool (data consumer) to achieve a goal. This actor only has the read permission for the resources in the Production Data Platform.
• Administrator (ADM): a person who manages the permissions to access PDP for different actors.

Figure 5 shows the relationships between the use cases and the actors.

Figure 5 Use cases of the Production Data Platform

UC1 – Aggregate Data Files

The first use case is derived from the application view item: “Store production data in a central storage.” Since the production data is file-based semi-structured or unstructured data, they need to be aggregated as files into a central place to facilitate data retrieval by Data Analysts.

UC1 can be specialized into two scenarios. The first one, UC1.4, is to put data files manually into the PDP by Data Engineers because Data Engineers have the knowledge and the responsibility to manage files manually if necessary. Another specialized use case, UC1.5, is to put files by Data File Providers into a place that the PDP can reach and aggregate them automatically. It means Data File Providers only need the knowledge of the pre-defined place that the PDP can reach.

Before the two specialized use cases are executed, Data Engineers need to set up the corresponding file specification and the aggregation job to support the automated file aggregation.
**UC2 – Retrieve Data**

The second use case is derived from the application view item: “Retrieve various production data from a central storage.” After the data files are stored in the PDP, which the task UC1 does, Data Analysts can use data analytics tools to retrieve data that they want to analyze.

**UC3 – Set Permission**

The third use case is derived from the application view item: “Access Control.” Before Data Engineers, Data File Providers, and Data Analysts can fulfill their jobs, Administrators need to set up the permission for them to interact with the PDP and access the files and data.

### 6.5.2 Requirements

The requirements generated from the Application view cover both functional and non-functional requirements. For each requirement, a priority was assigned and confirmed with stakeholders to make sure important ones will be finished first. The derived functional and non-functional requirements are listed in Table 8 and Table 9 respectively.

#### Table 8 Functional requirements

<table>
<thead>
<tr>
<th>Name</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR11 – The system must be able to store files</td>
<td>High</td>
</tr>
<tr>
<td>FR12 – The system must provide a function to put files to store manually</td>
<td>Medium</td>
</tr>
<tr>
<td>FR04 – The duration for keeping files in central storage must be configurable</td>
<td>Medium</td>
</tr>
<tr>
<td>FR07 – The aggregation process must be able to be automated</td>
<td>Medium</td>
</tr>
<tr>
<td>FR06 – The system must update the existing data instead of creating duplicate data when the specification of data aggregation has been changed</td>
<td>High</td>
</tr>
<tr>
<td>FR13 – The system must maintain the mapping between data source and data files</td>
<td>Medium</td>
</tr>
<tr>
<td>FR08 – Data must be able to be retrieved by Power BI</td>
<td>High</td>
</tr>
<tr>
<td>FR09 – Data must be able to be retrieved by R</td>
<td>Medium</td>
</tr>
<tr>
<td>FR10 – The system must provide the interoperability to data preprocessing process before data retrieval</td>
<td>Low</td>
</tr>
</tbody>
</table>

#### Table 9 Non-functional requirements

<table>
<thead>
<tr>
<th>Name</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF01 – One minimum application prototype must be used to demonstrate the POC</td>
<td>High</td>
</tr>
<tr>
<td>NF02 – The data used in the minimum application prototype must be from DOC production chain</td>
<td>High</td>
</tr>
<tr>
<td>NF03 – The system must be running on a cloud platform.</td>
<td>High</td>
</tr>
<tr>
<td>NF04 – The operations of the system must be access controlled based on existing account management system</td>
<td>High</td>
</tr>
<tr>
<td>NF05 – The data must be protected with a security mean if necessary</td>
<td>Low</td>
</tr>
<tr>
<td>NF06 – The system implementation should be based on standard products.</td>
<td>High</td>
</tr>
<tr>
<td>NF07 – The system implementation should avoid customization of standard products</td>
<td>Medium</td>
</tr>
<tr>
<td>NF14 – The licenses of used standard products should not affect existing IT infrastructure</td>
<td>Medium</td>
</tr>
<tr>
<td>NF09 – The system must support data replication</td>
<td>Low</td>
</tr>
<tr>
<td>NF10 – The system must support high availability</td>
<td>Low</td>
</tr>
</tbody>
</table>
The following subsections explain some requirements mentioned in Table 8 and Table 9.

Retrieved data by Power BI and R (FR08 and FR09)

Data analysts in the company can be divided into two groups: the people who use an interactive UI to analyze data and the people who use programming languages to analyze data.

In the current process, the first group of data analysts uses several different tools to analyze data, including Excel, QlikView, Oracle BI. However, the decision of adopting Power BI as the unified interactive data analytics tool has almost been made. Hence, it was selected as one requirement for data retrieval of the system.

Regarding the second group of data analysts, many of them use R programming language to analyze data, especially the research and development teams of different business units. Therefore, R was also selected as a requirement for data retrieval of the system.

Run the system on a cloud platform (NF03)

Before this project started, one strategy about IT in the company had been made: “Cloud first.” The main idea of “Cloud first” is to adopt public cloud platforms for hosting the systems used in the company, which means the existing systems will be migrated to a public cloud and new systems must be based on a public cloud as well. In the early phase of this project, Microsoft Azure was chosen as the preference of the public cloud.

Implement the system with standard products and avoid customization (NF06 and NF07)

The core business of the company is not about software development, so some of the concerns are about the approaches to system development. First, it is better to use standard products in the market to realize the system. The “standard products” means software or services that can be configured or customized to fulfill specific needs. Moreover, it is even better to avoid customization of the standard products if possible. In that way, the efforts of maintaining the system will be more about configurations instead of programming.

Demonstrate the system (NF11, NF12, and NF13)

During the discussion of the preliminary data analysis on the DOC production data, the production director and quality manager defined these three relations between data as the business questions to solve. These are good showcases to demonstrate the system. Among these three requirements, NF12 was set with a high priority since the data involved is from both the Production Data Platform and the ERP system, which was included in another ongoing BI project.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF11</td>
<td>The minimum application prototype must provide the function to show the relation between HOT and mortality</td>
<td>Medium</td>
</tr>
<tr>
<td>NF12</td>
<td>The minimum application prototype must provide the function to show the relation between HOT and outside climate data</td>
<td>High</td>
</tr>
<tr>
<td>NF13</td>
<td>The minimum application prototype must provide the function to show the relation between the duration of transit time and mortality</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Chapter 7 System Design

This chapter elaborates the system design and the process of conducting design. The system design was based on the requirements analysis results.

7.1 Design Process

The system design process after modeling requirements can be summarized into several steps (as shown in Figure 6): grouping requirements, proposing components, designing interactions, designing the directory structure, and refining the design.

After modeling requirements, the system design process started from the grouping of correlated requirements. The requirements that have related responsibilities were grouped together. Then different components were proposed to fulfill different requirement groups. For example, the component “Central Storage” was proposed because of the requirement that the system must be able to store files (FR11). In this step, the key part was the responsibility definition of each component.
After proposing the components, the process focused on the design of component-level interactions and the flows between the components instead of the detailed design of each component. The reason was that the non-functional requirements about running standard products without customization on a cloud platform (NF06, NF07, and NF03) are high priorities. It implied that the design should focus on the integration of components since the standard products have already take care of the detailed design.

Then the directory structure of the files stored in the system was designed. This is because we want to make the directory structure of each data type consistent to prevent disorganized storage and reduce the efforts to find a data type.

The design process was a continuous refinement. During the whole design process, various experiments and prototyping were carried out to see if the design works and the potential improvements.

In the system design, the “4+1” view model [4] was used to show the software architecture of the system. The “4+1” view model contains four views and is illustrated by scenarios, which is also known as use case view, as shown in Figure 7. The following sections elaborate each view with the help of Unified Modeling Language (UML) diagrams.

![Figure 7 The “4+1” view model [4]](http://www.uml.org/)

7.2 Development View

The development view in the 4+1 view model focuses on the software module organization.

This system design focuses more on how the components (software modules) are organized and interact with each other. The reason is that the components are going to be implemented by using standard products to satisfy the high-priority non-functional requirement about implementing the system based on standard products (NF06).

Based on the requirements, several components related to each other were proposed to fulfill different responsibilities, and they are shown in Figure 8 by using the component diagram in UML.

---

4 http://www.uml.org/
In addition to the components in the Production Data Platform, one minimum application prototype must be used to demonstrate the proof of concept according to the non-functional requirement NF01. The organization between the Production Data Platform and the minimum application prototype is shown in Figure 9.

Figure 8 Development view of the Production Data Platform
The traceability of the system components is shown in Figure 10 and Figure 11.
<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR04 - The duration for keeping files in central storage must be configurable</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR05 - The system must update the existing data instead of creating duplicate data when the specification of data aggregation has been changed</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR07 - The aggregation process must be able to be automated</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR08 - Data must be able to be retrieved by Power BI</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR09 - Data must be able to be retrieved by R</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR10 - The system must provide the interoperability to data preprocessing before data retrieval</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR11 - The system must be able to store files</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR12 - The system must provide a function to put files to store manually</td>
<td>Access Controller</td>
</tr>
<tr>
<td>FR13 - The system must maintain the mapping between data source and data files</td>
<td>Access Controller</td>
</tr>
</tbody>
</table>

Figure 10 Traceability of components – functional requirements
<table>
<thead>
<tr>
<th>Source</th>
<th>Access Controller</th>
<th>Automated Aggregation Manager</th>
<th>Central Storage</th>
<th>Data Retriever</th>
<th>File Aggregator</th>
<th>File Specification Manager</th>
<th>Minimum Application Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF01 - One minimum application prototype must be used to demonstrate the POC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF02 - The data used in the minimum application prototype must be from DOC production chain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF05 - The data must be protected with a security mean if necessary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF06 - The system implementation should be based on standard products.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF07 - The system implementation should avoid customization of standard products</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF04 - The operations of the system must be access controlled based on existing account management system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF03 - The system must be running on a cloud platform.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF10 - The system must support high availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF09 - The system must support data replication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF11 - The minimum application prototype must provide the function to show the relation between HOT and mortality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF12 - The minimum application prototype must provide the function to show the relation between HOT and outside climate data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF13 - The minimum application prototype must provide the function to show the relation between the duration of transit time and mortality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF14 - The licenses of used standard products should not affect existing IT infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11 Traceability of components – non-functional requirements
The components are described separately in more detail in the subsections below.

7.2.1 Central Storage

Central Storage aims to store files. It provides the File Manager interfaces for putting files in and the Data Provider interface for getting data out. In addition, it also provides the File Info Provider interface that exposes the basic information of files and folders, for example, the list of folders.

7.2.2 File Specification Manager

File Specification Manager aims to manage the metadata of data types, for example, the information about the data source of the files that belong to the same data type. It provides the Metadata Provider interface to expose metadata, and it also needs the implementation of a File Info Provider interface to get the metadata of files and folders.

7.2.3 File Aggregator

File Aggregator aims to put files based on a specific file specification into the Central Storage. It also provides a UI for users to put files manually.

It uses the metadata exposed by the implementation of Metadata Provider interface to determine where to put the files. After knowing the place to store the files, it uses the implementation of a File Manager interface to put the files.

7.2.4 Data Retriever

Data Retriever aims to retrieve data out of the Central Storage. It uses the implementation of a Metadata Provider interface to know the place of the files of a data type, and then it uses the implementation of a Data Provider interface to retrieve data. The data retrieved is exposed by Power BI Data Provider interface (FR08) and R Data Provider interface (FR09) for Power BI and R respectively.

7.2.5 Automated Aggregation Manager

Automated Aggregation Manager aims to manage the processes of file aggregations and handle the job scheduling of automation. Therefore, it needs an implementation of the File Manager interface to manage files during file aggregation.

7.2.6 Access Controller

Access Controller aims to keep the information that indicates which data file can be accessed or which action requests can be performed by which users or user groups. It provides an Access Control interface to authenticate a user and authorize a request from a user. It also needs to be compatible with the existing account management system in the company.

7.2.7 Minimum Application Prototype

Since this project is a proof of concept that needs a demonstration, a minimum application prototype is also proposed. It aims to demonstrate the functionality and process of the Production Data Platform. It uses an implementation of the Power BI Data Provider interface to get data from the PDP, and then it visualizes the data for users.
7.3 Logical View

The logical view in the 4+1 view model is the object model of the design. The main logical view of this system is about the key data used in the system operation. It is illustrated using the class diagram in UML, as shown in Figure 12.

Figure 12 Logical view of the key data of the Production Data Platform

The key data in the system are the following:

- **Data Type**: the class that represents one data type from a data source, and it has at least one Data Type Versions.
- **Data Type Version**: the class that represents one version of a Data Type, and it has one File Specification to describe itself. Besides, one Data Type Version also associates with one Automated Aggregation Profile to carry out the automated file aggregation.
- **File Specification**: the class that describes a Data Type Version about what schema it is, where it can be found, and how it can be accessed. These information is mandatory for the Data Engineers and Data Analysts to use the data type.
- **Automated Aggregation Profile**: the class that describes where to obtain the files of a Data Type Version and where to put them into the Central Storage.
- **Automated Aggregation Job**: the class that describes when to execute the Automated Aggregation Profile.

Regarding the directory structure in the Central Storage, different types of folder are related to each of the BUs, Facilities, Data Types, and Data Type Versions. One BU Folder has one or more Facility
Folders since one BU has one or more facilities. One Facility Folder has one or more Data Type Folders since one facility collects one or more data types. One Data Type folder has one or more data type Version Folders since one data type might have one or more versions over time. For each Version Folder, it has one Raw File Folder. If the data type version can be preprocessed and has a preprocessing pipeline, there will be one Preprocessed File Folder and another two folders to contain the files with different states, which are not shown in the figure. Please refer to section 8.1 for the details about the directory structure.

7.4 Process View

The overview of the main usage flow of the Production Data Platform is shown in Figure 13. First, Data Engineer set up the required things for aggregating the files of a data type. There are two parts for it: the file specification and the automated aggregation profile. Second, the files are aggregated either automatically or manually. Then, Data Analysts retrieve data to develop data-driven applications. And meanwhile, Data Engineers need to set up a new version of the file aggregation when the file schema of the same data type is changed. The more detailed flow of the PDP is shown in Figure 14.

Before the main usage flow of the system, it is recommended that the domain experts propose new important data types and discuss them with Data Engineers. Then, domain experts and Data Engineers prioritize the proposed data types due to the different potential and limited resources. After the decision of aggregating the new data type, different actors participate in the main system process shown in Figure 13.

The detailed interactions between components are illustrated with several sequences and described in the subsections below. The NF07 – the system implementation should avoid customization of standard products was considered to design the system flow, so some interactions between users and two components were designed to fulfill the assembly of interfaces between two components without customizations of standard products.
Figure 13 Process view of the Production Data Platform
7.4.1 Create file specification

When a data type from a data source is proposed to be aggregated into the Production Data Platform, the creation of the required resources for the aggregation of the data type is necessary. First, Data Engineers create a set of folders according to the directory structure in the Central Storage for storing the files of the data type. Then, if the aggregated files should be expired after a period of time, Data Engineers set up the expiry date rules. After the setup in the Central Storage, Data Engineers use the File Specification Manager to register the created folders and add relevant metadata, consisting of the
location in the Central Storage, the key people, the schemas, and the information about how to access them.

The registration and addition of relevant information are in a folder level instead of file level since the folder contains the files from the same data source with the same format. In this way, every file aggregated to the folder can be considered as the same format. The process is illustrated in Figure 15.

![Figure 15 Sequence of creating file specification](image)

7.4.2 Create Automated Aggregation

If the files of a data type can be aggregated automatically, Data Engineers create an automated aggregation profile for this purpose. Data Engineers first create the connection to the staging store that contains the files of the data type and tests the connection. Then, the connection to the folder in the Central Storage is also created and tested by Data Engineers. After the creation of the connections, Data Engineers create an automated aggregation profile to copy the files from the staging store to the folder in the Central Storage. Finally, Data Engineers schedule jobs based on the automated file aggregation specification for aggregating files automatically. The process is illustrated in Figure 16.
7.4.3 Aggregate files

After the file specification and the automated aggregation profile are created, files can be aggregated into the Central Storage. There are two kinds of processes here: aggregate manually and automatically.

For the manual aggregation, Data Engineers put the files directly into the location specified in the file specification of the data type with the File Aggregator. Before it, if Data Engineers do not know where to put the files, they first check the File Specification Manager to find out the location of the data type in the Central Storage.

The assembly of the Metadata Provider interface between the File Aggregator and the File Specification Manager is realized with user actions. First, Data Engineers go to the File Specification Manager to search and get the file specification of a data type. Then Data Engineers use the location information in the file specification as a reference to ask the File Specification Manager to put files into the Central Storage.

For the automated aggregation, File Providers are responsible for putting files in the location (staging store) specified in the automated aggregation profile. Then the files are aggregated when the
automated aggregation job is triggered by the Automated Aggregation Manager. The process is illustrated in Figure 17.

![Figure 17 Sequence of aggregating files](image)

7.4.4 Retrieve data

When Data Analysts want to retrieve data out of the PDP, they first use the File Specification Manager to find out what data types they can use and where they can find the files. If they realize they do not have access to the data, they request the access from Administrators. Then, they use the Data Retriever to get the data out of the Central Storage based on the file specification. The process is illustrated in Figure 18.
7.4.5 Set Permissions

Whenever Data Analysts or Data Engineers find out that they do not have the permission to fulfill their jobs, they contact Administrators to set appropriate permissions for them. Administrators first create a credential for the person if the person does not have one. Then Administrators set the requested permissions to the credential and return it to the users who requested it. The process is illustrated in Figure 19 with an example that Data Analysts request the permissions for accessing folders in the Central Storage.

7.4.6 Versioning Data Types

To satisfy the requirement about the single source of truth principle, which is FR06 – avoid duplicate data, a flow of versioning file aggregation is proposed. When the schema of the raw files is changed,
Data Engineers follow a flow to create a new version of the data aggregation to aggregate the raw files with new schema. A part of the flow is similar to the creation of a file specification.

First, Data Engineers create a new set of related folders in the Central Storage to contain the raw files with the new schema. Then Data Engineers register and add metadata to the created folders. Then, if the existing automated aggregation profile is still needed to run, Data Engineers create a new automated aggregation profile for the new version of the data type. Otherwise, Data Engineers change the destination specified in the original automated aggregation profile. The detailed process is illustrated in Figure 20.

Figure 20 Flow of versioning data types
### 7.5 Physical View

The physical view in the 4+1 view model describes the mapping between the software and the nodes running the software. The system is expected to run on a cloud platform, so most of the components are deployed on a cloud platform, which encapsulates the setup of hardware. Only File Aggregator can be deployed on a local machine. The deployment diagram is shown in Figure 21.

![Figure 21 Physical view of the Production Data Platform](image-url)
Chapter 8 Design and Technology Decisions

This chapter describes the decisions made during the design and realization. The main content of each topic is about two parts: the comparison and the decision.

8.1 Directory structure

For long-term usage, there must be a well-defined directory structure for organizing all files since the Production Data Platform aims to store the files from all data sources during the production chains in the company, and the number of the data types aggregated must grow over time. This section describes the reasoning of the design of the directory structure in the Central Storage. The directory structure was designed with taking several things into account:

- The organizational structure of the company.
- The process that Data Analysts carry out data analyses.

It leads to a directory structure with five levels to store the files aggregated, as shown in Figure 22.

```
|-- business-unit-1
|   |-- facility-1
|   |   |-- data-type-1
|   |   |   |-- v1
|   |   |   |   |-- preprocessed
|   |   |   |   |-- raw-preprocessed-archive
|   |   |   |   |-- raw-unprocessed
|   |   |   |   `-- raw-invalid
|   |   | `-- v2
|   |   |   |-- preprocessed
|   |   |   |-- raw-preprocessed-archive
|   |   |   |-- raw-unprocessed
|   |   |   `-- raw-invalid
|   `-- data-type-2
|   |   |-- v1
|   |   |   |-- preprocessed
|   |   |   |-- raw-preprocessed-archive
|   |   |   |-- raw-unprocessed
|   |   |   `-- raw-invalid
|   | `-- v2
|   |   |-- preprocessed
|   |   |-- raw-preprocessed-archive
|   |   |-- raw-unprocessed
|   |   `-- raw-invalid
| `-- facility-2
   `-- cross-facilities
|-- business-unit-2
|-- business-unit-3
```

Figure 22 Directory structure of the Central Storage

The first level is organized by business units. Hendrix Genetics is a multi-species company, and there are multiple BUs that focus on different species. Moreover, the data analyses conducted in the company are mostly species-oriented, for example, the layer species.

The second level is organized by the facilities owned by the BU. In each BU, there are multiple facilities to support their production during the whole production chain. For example, the hatchery located at Boxmeer and owned by layer BU. In addition, some data are collected across more than one facility in the same BU, in this case, there will be another folder to store these data, “cross-facilities” for instance.
The third level is organized by data types. A data type means a type of data collected by a system for a purpose. Each facility has multiple systems to collect different types of data, so the third level could be organized by systems and contains multiple data type folders. However, it is better to organize the third level by the data types collected by the systems in the facility. The reason is that after Data Analysts know which BU and facilities that they can find the data they need for their analyses, they care about the data. It does not matter much to them where the data type is from.

Even though the system information does not matter much for Data Analysts, the folder name for the third level is recommended to be a combination of the system information and the data type collected by the system. The system part of the folder name does not need to be exactly the system name. Instead, a common term that most people know is recommended. The reason is that only the live operation people know the exact system name. Therefore, one of the example from the Boxmeer hatchery could be “hatcher-climate,” it means the climate log from the hatcher system, which is the Pas Reform SmartCenter system.

The fourth level is organized by the versions of the data type. The schema of a data type might change over time, so the versioning of the file aggregation for a data type is proposed. Please refer to the next subsection for details.

The fifth level is organized by different stages of the files that belong to the same version of a data type. The reason is that the system aims to facilitate data-driven applications, and if the data type version can be preprocessed to deal with the most common data quality issues, for example, the data inconsistency of a field, there will be these folders to support the preprocessing. The folders are the following:

- The “raw-unprocessed” folder: it contains the raw files aggregated into the system. It is the entry point for raw files. The folder should be empty after each preprocessing.
- The “preprocessed” folder: it contains the preprocessed result (flattened the structure, solved content issues, etc.) of the raw files. The preprocessed data should contain only the attributes from raw files without the introduction of new attributes.
- The “raw-preprocessed-archive” folder: it contains the raw files that have been preprocessed. Raw files are moved from the “raw-unprocessed” folder to “raw-preprocessed-archive” folder.
- The “raw-invalid” folder: it contains the raw files that cannot be preprocessed successfully after preprocessing. It means the invalid raw files are moved from the “raw-unprocessed” folder to the “raw-invalid” folder.

The folders in the fifth level should be organized with three additional levels: the year, month, and date. For example, the sub-directory structure could be “2018/10/09,” where the slash symbol stands for the hierarchy. The folder of a date contains the files aggregated on that date.

### 8.2 File aggregation versioning

The Production Data Platform tends to be the single source of truth for production data, so the changes in the schema of raw files in same data type will not result in duplicate data. Hence, the versioning process is proposed to tackle this problem.

There are three approaches to versioning the file aggregations of a data type in the PDP, and each has different pros and cons. The summary is shown in Table 10, and the third approach is chosen as the design decision.
Table 10 Comparison of file aggregation versioning approaches

<table>
<thead>
<tr>
<th></th>
<th>First approach</th>
<th>Second approach</th>
<th>Third approach (chosen)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Use the same folder to contain multiple versions of files of the same data type.</td>
<td>Create a new data-type level folder to contain the raw files with the new schema.</td>
<td>Create a new version folder under a data-type level folder to contain the raw files with the new schema.</td>
</tr>
<tr>
<td><strong>Directory structure</strong></td>
<td>`</td>
<td>-- data-type-1 `</td>
<td>-- subfolders</td>
</tr>
<tr>
<td><strong>Complexity for Data Engineers</strong></td>
<td>Higher</td>
<td>Lower</td>
<td>Lower</td>
</tr>
<tr>
<td><strong>Complexity for Data Analysts</strong></td>
<td>Lower</td>
<td>Higher</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Complexity for Administrators</strong></td>
<td>Lower</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td><strong>Storage usage for the setup</strong></td>
<td>Higher</td>
<td>Lower</td>
<td>Lower</td>
</tr>
<tr>
<td><strong>Able to handle multiple versions</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Need to preprocess files</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

The following subsections describe each approach in detail.

8.2.1 First approach

The first approach is to use the same directory structure to contain all files with multiple version of schema from the same data source. The directory structure remains the same after versioning, as shown in Figure 23.

```
|-- data-type-1
 ||-- preprocessed
 ||-- raw-preprocessed-archive
 ||-- raw-unprocessed
```

Figure 23 Directory structure of the first versioning approach

The steps of this approach are the following:

- **Step 1.** Preprocess the unprocessed raw files with original schema into data
- **Step 2.** Document the change of the raw file schema in the data type’s file specification
- **Step 3.** Preprocess the raw files with the new schema
- **Step 4.** Merge the preprocessed data from Step 1 into the new preprocessed data to make sure all preprocessed data is in the same format
- **Step 5.** Document the change of the preprocessed data schema
- **Step 6.** Change the destination of the existing automated aggregation profile to aggregate the raw files with the new schema
In this approach, the files with the new schema go to the same raw file folder. Since there is no structured way to determine which files in the same folder are with which schema, it is necessary to preprocess the files with original schema and make the folder empty before aggregating the files with the new schema. Then the raw file folder contains only the files with the new schema. The effort of preprocessing is considered as one of the main disadvantages of this approach. Compared to another two approaches, the required preprocessing also causes the relatively higher storage usage before Data Analysts use the data. Another disadvantage is that it cannot handle the situation that parallel running of multiple versions because multiple versions share the same folder, and there is no way to tell the files with multiple versions of the schema.

This approach creates significant complexity for Data Engineers. Even though Data Engineers do not need to set up another file aggregation for the raw files with the new schema, they need to preprocess the existing raw files with original schema and also transform them to fit the new preprocessed data schema.

This approach brings no impact to Data Analysts and Administrators since the directory structure remains the same. Data Analysts do not need to know another new place to access the data. Meanwhile, Administrators also do not need to change the permissions for the Data Analysts to access the data.

8.2.2 Second approach

The second approach is to create a new data-type level folder to contain the files with the new schema. The directory structure after versioning is shown in Figure 24.

```
|-- data-type-1_v1
  |-- preprocessed
  |-- raw-preprocessed-archive
  `-- raw-unprocessed

|-- data-type-1_v2
  |-- preprocessed
  |-- raw-preprocessed-archive
  `-- raw-unprocessed
```

Figure 24 Directory structure of the second versioning approach

The steps of this approach are the following:

Step 1. Create a data-type level folder with a version number and relevant subfolders for the aggregation of the raw files with the new schema.

Step 2. Register and annotate these folders.

Step 3. Document the new schema and the relation between the new and the original file aggregation.

Step 4. Change the destination of the existing automated aggregation profile or create a new one if the original one still needs to be run.

This approach asks Data Engineers to carry out a very similar flow of creating the file aggregation for a data type, so the complexity for Data Engineers stays the same. In this way, Data Engineers do not need to preprocess the unprocessed raw files with the original schema.

The impact caused by this approach for Data Analysts is that they need to be aware of the new file aggregation. Meanwhile, Administrators need to add new permissions for Data Analysts to access the new file aggregation.
8.2.3 Third approach

The third approach is to create a completely new set of folders under a main data source folder to contain the raw files with the new schema. The directory structure after versioning is shown in Figure 25.

```
|-- data-type-1
  |-- v1
  |   |-- preprocessed
  |   |-- raw-preprocessed-archive
  |-- v2
     |-- preprocessed
     |-- raw-preprocessed-archive
     |-- raw-unprocessed
```

Figure 25 Directory structure of the third versioning approach

The steps of this approach are the following:

Step 1. Create a version-level folder and relevant subfolders under the same data-type level folder for the raw files with the new schema

Step 2. Register and annotate these folders

Step 3. Document the new schema and the relation between the new and the original file aggregation

Step 4. Change the destination of the existing automated aggregation profile or create a new one if the original one still needs to be run

The complexities created by this approach for Data Engineers are similar to the second approach. The difference is the location to contain the raw files with the new schema.

For Data Analysts, since the new location is under the same data-level folder, they do not need to pay additional effort to find it even if they do not check the file specification for changes. Meanwhile, Administrators do not need to add new permissions for Data Analysts to access the new location because the new location is under the same data-level folder, which means Data Analysts have already had the permissions.

Compared to the second approach, this approach is even better since Data Analysts do not need to be aware of the new file aggregation for the raw files with the new schema. When they browse the data-type level folder, they know the new version without additional effort.

In the end, the third approach is chosen because of the following reasons:

- Data Engineers only need to follow a similar flow to set up the new file aggregation.
- Administrators do not need to configure the access for the Data Analysts to the new folders.
- Data Analysts do not need to know another data-type level folder for a different version.
- It can handle the file aggregations for multiple versions of the schema.
- No storage is required during the setup of a new version.
8.3 Central Storage implementation

The most important component in the system design is the Central Storage since it was designed to fulfill the core functional requirements, which are about storing and exposing files. This section describes the reasoning of finding the appropriate products to realize the design of the Central Storage.

8.3.1 Candidate products selection

The decision-making process started with finding the suitable products with the most important non-functional requirements. The first non-functional requirements applied to find the qualified candidate products are NF03 – the system must be running on a cloud platform and NF06 – the system implementation should be based on standard products. Since it has been decided to adopt Microsoft Azure as the preferred cloud platform for hosting systems in the company, the candidates are better from the Azure product family.

Meanwhile, the non-functional requirement NF07 – the system implementation should avoid customization of standard products was also taken into account. Hence, only the ready-to-use products that function as storages were focused. It means the products like virtual machines was ignored since it needs extensive customizations to realize the Central Storage component.

After the filtering by the requirements mentioned above, some candidates popped up. The filtering process was continued with the core functional requirement: FR11 – the system must be able to store files. In this stage, some Azure products that do not meet the requirements, Azure Table Storage for instance, were dropped. Then, the requirements about retrieving data out of the storage are considered. After filtering, Azure Data Lake Store (ADLS) and Azure Blob Storage were chosen to be the candidates for the Central Storage.

8.3.2 Candidate products comparison

The decision-making process continued with the comparison of these two candidate products. The summary of comparison is shown in Table 11. The comparison ignores the preview features, which are not generally available.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Azure Data Lake Store (ADLS)</th>
<th>Azure Blob Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR12 – manual file aggregation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FR04 – expiry date</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>FR08 – Power BI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FR09 – R</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FR10 – interoperability of preprocessing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NF04 – account management integration</td>
<td>Yes, by folders and files</td>
<td>Yes, by storage account</td>
</tr>
<tr>
<td>Access control</td>
<td>Yes, by folders and files</td>
<td>Yes, by storage account</td>
</tr>
<tr>
<td>NF05 – security mean</td>
<td>See the two rows below</td>
<td>See the two rows below</td>
</tr>
<tr>
<td>Security on file transmission</td>
<td>Yes, with cloud firewall and HTTPS(TLS)</td>
<td>Yes, with cloud firewall, HTTPS, SMB 3.0 with encryption, or Client-Side</td>
</tr>
</tbody>
</table>

6 https://azure.microsoft.com/en-us/services/storage/blobs/
<table>
<thead>
<tr>
<th>Security on file storage</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF09 – data replication</td>
<td>Yes, only one option</td>
</tr>
<tr>
<td>NF10 – high availability</td>
<td>SLA 99.9%</td>
</tr>
<tr>
<td>Size limit</td>
<td>No</td>
</tr>
</tbody>
</table>

**FR12 – manual file aggregation**

Both candidates provide web UI for file management, and the UI can be accessed via the Azure Portal. In addition, Microsoft also provides a cross-platform application called “Azure Storage Explorer”, which can be used to manage the files in both candidates.

**FR04 – expiry date**

Currently, only ADLS provides the mechanism to set expiry dates of files, and the expiry dates are file-based.

**FR08 – Power BI**

Power BI supports getting data from both candidates. The ways to carry out the authentication and authorization are the same for both candidates.

**FR09 – R**

The scripts written in R can retrieve files from both candidates. The fundamental way to achieve this is their REST APIs\(^7\). Also, one R package called AzureSMR\(^8\), which is maintained by Microsoft, can be used to retrieve files from both candidates. R scripts can also access and process a great amount of data by using other computing systems, for example, Apache Spark\(^9\).

**FR10 – interoperability of preprocessing**

Both candidates provide various APIs for exposing files for preprocessing, and Azure also provides some products that can be used to preprocess the files stored in both candidates.

**NF04 – account management integration**

Hendrix Genetics currently uses Active Directory (AD) with Azure Active Directory (Azure AD), which is the cloud version of AD, as the account management system. Both candidates can integrate with Azure AD to carry out access control.

The access control of ADLS is based on folders and files, which means it is possible to set the access to a part of the files for Data Analysts. Since the access control of Blob Storage is based on storage accounts and containers, it cannot achieve such detailed control like ADLS.

---

\(^7\) [https://azure.microsoft.com/en-us/features/storage-explorer/](https://azure.microsoft.com/en-us/features/storage-explorer/)

\(^8\) [https://docs.microsoft.com/en-us/rest/api/datalakestore/](https://docs.microsoft.com/en-us/rest/api/datalakestore/)


\(^10\) [https://github.com/Microsoft/AzureSMR](https://github.com/Microsoft/AzureSMR)

NF05 – security mean

This requirement can be divided into two based on the lifecycle of files: the security on file transmission and the security on file storage.

For the security on file transmission, both candidates provide HTTPS for securing the file transmission, and it is also possible to set up a cloud-based firewall to restrict the network access to the storage. In addition, Blob Storage provides “client-side encryption,” which means files can be encrypted before data transmission, and Blob Storage decrypts the files after receiving.

For the security on file storage, both candidates encrypt and decrypt the files with the AES-256 algorithm, and the keys used in the encryption and decryption can be managed by customers.

NF09 – data replication

ADLS provides only Locally (LRS) replication, and by contrast, Blob Storage provides Zone (ZRS), Geo (GRS), or Read-access geo (RA-GRS) in addition to Locally (LRS) replication.

NF10 – high availability

For high availability, this comparison uses Service Level Agreement (SLA) to identify their built-in robustness for high availability. Microsoft guarantees that ADLS will be available at least 99.9% of the time. The availability of Blob Storage depends on the data replication option: 99.9% for RA-GRS and 99% for LRS, ZRS, and GRS.

Other differences

Another difference between these two candidates is the size limit of an instance. There is no size limit on accounts and files for ADLS where Blob Storage has the size limits.

8.3.3 Technology decision for the Central Storage implementation

Based on the comparisons of both candidates, ADLS was chosen to be the product to fulfill the Central Storage of the prototype. There are several reasons for the decision.

First, the authorization mechanism of ADLS is more suitable for the storage that is accessed by multiple Data Analysts. The authorization of ADLS is based on client credentials, which can be created individually for each different Data Analysts. However, the authorization of Blob Storage requires an access key that should not be shared with anybody, so it makes Blob Storage not a proper candidate.

Second, the access control of ADLS can support the scenario in which Data Analysts are authorized to access not all files but only part of files since it is based on folder level or file level. Blob Storage can only authorize the client credential on storage level or container level, which is not fine-grained as ADLS.

Third, ADLS is able to set expiry dates of files where current Blob Storage cannot.

Last but not least, even though ADLS provides only locally replication, the SLA of ADLS is higher than the Blob Storage with the same replication mechanism. After the high availability design and implementation, the implementation with ADLS instances provides higher availability.
Chapter 9 Implementation

This chapter provides the overview of the implementation of the Production Data Platform prototype and the minimum application prototype based on the given design and the decision of technology choices.

9.1 Introduction

The implementation started with only the high-level design of each component. The decision was made because of one of the highest non-functional requirement NF06 – the system should be based on standard products. If some products can be used to realize the design, the detailed design about how a component works internally is not necessary. It is also the reason why a series of experiments was carried out during the design phase. Therefore, most of the effort for the implementation is about finding suitable products and their usage to satisfy the high-level design.

Furthermore, the products used in the implementation must be running on cloud platforms according to the non-functional requirement NF03. The company preferred Microsoft Azure as the cloud platform, so the products from Azure were considered as priority candidates to implement the system. At the end of the implementation, all components were realized with the products from Azure. Besides, the implementation with cloud-based products also implies the deployment is done when the instance of a product is created.

The sections below describe the implementation of the Production Data Platform prototype and the minimum application prototype.

9.2 Implementation of the Production Data Platform

The implementation of each component of the Production Data Platform prototype is shown in Figure 26.
Figure 26 Implementation of components
The deployment of the File Aggregator implementation with the Microsoft Azure Storage Explorer was on a local machine. The implementations of other components except the File Aggregator were deployed on the Microsoft Azure cloud platform. The deployment of the prototype implementation is shown in Figure 27.

The sections below describe the implementation of each component based on the products from Azure cloud platform. All components except the File Aggregator were implemented by creating and configuring the instance of different Azure products through the Azure Portal\(^\text{12}\) (as shown in Figure 28), which is the single and unified web-based console for all Azure products.

---

9.2.1 Access Controller

The Access Controller is the key component responsible for the authentication and authorization of each type of users and request. The current account management system in the company, which is based on Azure Active Directory (Azure AD), was used to realize the component design. Azure AD provides several ways to authenticate and authorize users and requests, including using the account credential and using the secret key of a registered application. Most of the implementation of components in this prototype use account credential to do the authentication and authorization. The screenshot of the Azure AD instance is shown in Figure 29.

9.2.2 Central Storage

Based on the responsibility of the Central Storage component, two Azure products were selected and compared: the Azure Data Lake Store (ADLS) and the Azure Blob Storage. The ADLS was chosen as the product to realize the Central Storage. The detailed reasoning about this decision can be found in section 8.3. The screenshot of the Azure Data Lake Store instance is shown in Figure 30.

One instance of ADLS was created by the Azure Portal with the implementation of the designed directory structure to contain aggregated files. The design of the directory structure is described in section 8.1.

The users and requests of the ADLS instance are authenticated and authorized with account credentials. ADLS provides the interfaces for putting files into it and exposing files to outside, which can be used to implement the File Manager interface and the Data Provider interface respectively. Besides, ADLS also provides the interface to expose the information of folders and files, which can be used to implement the File Info Provider interface.

The functionality of setting expiry dates of files was realized with the built-in web UI and API.
Figure 30 Central Storage implementation with Azure Data Lake Store

9.2.3 File Aggregator

There are two possible choices for the implementation of the File Aggregator component. The first choice is the “Data explorer” functionality from the web-based UI of the ADLS instance, which is the implementation of the Central Storage. The second choice is the Microsoft Azure Storage Explorer\(^\text{14}\) (as shown in Figure 31), which is a standalone cross-platform application for the management of the instances of Azure storage products. Both interact with the ADLS instance by the implementation of the File Manager interface. Data Engineers can choose either one of them to put files into the Central Storage.

Both choices use the Access Control interface implementation from the Azure AD instance, the implementation of the Access Controller, to authenticate and authorize users and requests.

The usage of the Metadata Provider interface from the File Specification Manager is implemented as a separate user action. Data Engineers search for the file specification on the implementation of the File Specification Manager, then they refer to the location specified in the file specification to put files.

\(^{14}\) https://azure.microsoft.com/en-us/features/storage-explorer/
9.2.4 Data Retriever

A part of the functionalities of the ADLS instance was used to implement the Data Retriever component. The R Data Provider interface is implemented with the REST APIs provided by the ADLS instance, and the APIs enable the process of getting files from the ADLS instance. The APIs internally implemented the usage of the Data Provider interface of the Central Storage component. The implementation of the Power BI Data Provider shares the same logic with the implementation of R Data Provider.

Every request for getting data is authenticated and authorized by the Azure AD instance. The authentication and authorization for using the R Data Provider interface are done by using a generated secret key where for the Power BI Data Provider interface the account credential is used.

The usage of the Metadata Provider interface is implemented with an action carried out by Data Analysts between File Specification Manager and this component. Data Analysts use the implementation of the interface from the File Specification Manager instance to obtain information such as the location and then feed the information to this component. In this way, Data Analysts can specify where to obtain the files.

9.2.5 File Specification Manager

File Specification Manager is responsible for managing the metadata of data types, and it was realized with one Azure Data Catalog instance, a product used to manage the registration and discovery of data assets. The Metadata Provider interface was implemented with the UI and user actions. Data Engineers and Data Analysts use the Azure Data Catalog UI (as shown in Figure 32) to obtain a file specification for the usage of this interface from other components, for example, the Data Retriever component.

The Azure Data Catalog instance uses Azure AD with an account credential to authenticate and authorize every request of using it from users.

---

Besides, Azure Data Catalog has a built-in capability to retrieve the fundamental information of files and folders, so the usage of the File Info Provider interface of the Central Storage is implemented.

![Microsoft Azure Data Catalog](https://azure.microsoft.com/en-us/services/data-factory/)

Figure 32 File Specification Manager implementation with Azure Data Catalog

### 9.2.6 Automated Aggregation Manager

Automated Aggregation Manager is responsible for maintaining the profile and jobs to aggregate files automatically at the scheduled time. Azure Data Factory[^16] was chosen to fulfill this component. Azure Data Factory is a product that can be used to copy files from and to various types of data stores, for example, FTP and ADLS, at a scheduled time. The usage of the File Manager interface from the Central Storage instance was implemented with Azure Data Factory’s built-in ADLS connector used to manage files in an ADLS instance. In addition, the Azure Data Factory instance uses Azure AD with an account credential to authenticate and authorize every request of using it from users. The screenshot of the implementation is shown in Figure 33.

9.3 Implementation of the DOC quality application prototype

The minimum application prototype was developed for verifying and demonstrating the Production Data Platform. It was designed and implemented with Power BI\(^\text{17}\), the chosen interactive data analytics tool for the whole company. Power BI has the built-in functionality to connect to Azure Data Lake Store (as shown in Figure 34), which is the product used to implement the Central Storage.

\[^{17}\text{https://powerbi.microsoft.com/}\]

Figure 33 Automated Aggregation Manager implementation with Azure Data Factory

Figure 34 Get data from ADLS by Power BI
In order to satisfy the non-functional requirements about the showcases provided by this application prototype (NF02, NF11, NF12, and NF13), several relevant file-based datasets were aggregated to the Production Data Platform, as listed in Table 12.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production statistics</td>
<td>MTech</td>
</tr>
<tr>
<td>Outside temperature log</td>
<td>Priva TC Manager</td>
</tr>
<tr>
<td>Hatcher temperature log</td>
<td>Pas Reform SmartCenter</td>
</tr>
<tr>
<td>Temperature log</td>
<td>Sensitech TempTale</td>
</tr>
</tbody>
</table>

Only the requirement NF12 was satisfied by the implementation, as shown in Figure 35. However, some other showcases in addition to the three mentioned as the requirements were also implemented, and they were accepted by the related stakeholders.

![Figure 35 Implementation of DOC quality application prototype](image)

The data source of the datasets used in the application prototype implementation was configured with a scheduled refresh mechanism, so the end users of the application always see the report based on the latest data.
Chapter 10 Verification and Validation

This chapter describes the verification and validation of the Production Data Platform prototype. A series of verification and validation was carried out after the implementation.

10.1 Verification

Verification is “an attempt to ensure that the product is built correctly [5].” The verification of this system implementation focused on only the system as a whole since it is assumed that the implementation of each component has been verified. The assumption comes from the fact that the components are implemented with the products from the Microsoft Azure cloud platform, and each product was verified by Microsoft.

The verification of the system relied on the integration test. One hackathon was held as the integration test after the implementation of the PDP prototype. During the hackathon, all the implementation of the components and the processes were verified. All the roles were also role-played to mimic the real scenario.

10.2 Validation

Validation is “an attempt to ensure that the right product is built [5].” It can be carried out by checking if the requirements are fulfilled by the system design and implementation.

The design and implementation were progressed with agile-like methodology: having a plan and review meeting every two weeks. In the meeting, the stakeholders reviewed the latest development of the system and made sure the related requirements were satisfied. There was another standalone meeting that aims to validate the whole system after the implementation. It was found out that some non-functional requirements were not satisfied yet, and they became the future works. The validation result is shown in Table 13.

Table 13 Validation of system requirements

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Priority</th>
<th>Status</th>
<th>Future action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR11, FR06, FR08, NF01, NF02, NF03, NF04, NF06, NF12</td>
<td>High</td>
<td>Designed and implemented</td>
<td>–</td>
</tr>
<tr>
<td>FR12, FR04, FR07, FR13, FR09, NF07, NF14</td>
<td>Medium</td>
<td>Designed and implemented</td>
<td>–</td>
</tr>
<tr>
<td>NF11, NF13</td>
<td>Medium</td>
<td>Not accomplished</td>
<td>Continue the design and implementation based on the stakeholders’ latest interests</td>
</tr>
<tr>
<td>FR10, NF05</td>
<td>Low</td>
<td>Designed and implemented</td>
<td>–</td>
</tr>
<tr>
<td>NF09, NF10</td>
<td>Low</td>
<td>Not accomplished</td>
<td>Design and implement the requirements</td>
</tr>
</tbody>
</table>
Chapter 11  Conclusions

This chapter elaborates the achieved results and the added value to the stakeholders. The future works are also described in this chapter.

11.1 Results

The results of this project are the design and implementation of a system called Production Data Platform, which aims to contain the files aggregated from different data sources and provide the files to data consumers for data-driven applications development.

The design contains several components that fulfill a subset of the functional and non-functional requirements. Besides, the roles and the flows regarding the usage of the system are also given. Data Engineers, which do not exist yet in the company, are the most important role in this system. Some usages of interfaces between components are designed with some user actions because of the requirement that the system implementation should avoid customization of standard products. The user actions make two components work together.

The implementation is based on the products from Microsoft Azure cloud platform because of the requirement that the system must be running on a cloud platform. The implementation was verified with stakeholders by conducting a hackathon to analyze production data. It was also validated by stakeholders over time and a dedicated meeting.

In addition, a DOC quality application prototype was built for verifying and demonstrating the system prototype. The application prototype was also verified and validated with the stakeholders from the layer production team.

The project results provide one part of the basis for data-driven applications and can be combined with the ongoing BI project results. In this way, more comprehensive analyses can be carried out in the future. The business stakeholders can think out of the box, think big, and think with the help from other existing resources.

Since the project results make data analyses able to combine multiple datasets from different systems, the stakeholders can think out of the box and not be limited by their imagination. Due to the scalability from the cloud-based implementation, the result can facilitate large-scale automated analyses, so the stakeholders can think big. Because of the high interoperability nature of the cloud-based implementation, the stakeholders can think with the help from other existing resources and combine heterogeneous resources together to realize a data-driven application.

11.2 Future work

There are three main things needed to do after the project finishes: satisfying remaining requirements, combining with the ongoing BI project, and improving the system.

Two non-functional requirements with low priority that were not satisfied with the design and implementation need to be satisfied. First, the system must support data replication. Second, the system must support high availability. The fulfillment of these two requirements will lead to a more robust system to meet the mission-critical analysis scenarios.

Besides, the two low-priority non-functional requirements about the showcases of this systems were not satisfied during the project. Instead, some other showcases were implemented and accepted by the related stakeholders. Therefore, the design and implementation of these two requirements depend on the interests of the related stakeholders.

Moreover, the project results can be combined with the ongoing BI project. This project scope was affected by the ongoing BI project to focus on the semi-structured and unstructured data collected...
during the production chain. The reason was that the ongoing BI project focused on the structured data, including the structured data collected during the production chain. Therefore, combining the results from these two projects will create a more comprehensive infrastructure for data-driven applications. Azure Data Catalog is used in both projects, and the instances need to be merged, then Data Analysts can have the single data catalog to know all available structured, semi-structured, and unstructured data for analyses.

Besides, the system can be improved by replacing the product used to realize the design of the Central Storage component with other better candidate product in the future. Microsoft Azure introduced the Azure Data Lake Store Gen2 during the project. It has more comprehensive features for storing the files for analytics usage. However, it is still in the preview phase after the project.
Chapter 12 Project Management

This chapter describes how the project was carried out from the project management perspective.

12.1 Introduction

The project was carried out with the project management approach defined by the company, which consists of four phases: project initiation, project planning, project execution, and project closure. A document called “Project Total Document” for the project management consists of the information of those four project phases. It was maintained throughout the project period. The following sections describe the project management of this project.

12.2 Work Breakdown Structure (WBS)

The work breakdown structure (WBS) containing several work packages was defined during the project initiation phase (as shown in Figure 36):

- Project Initiation: the activities about developing this WBS, analyzing the problem, and defining the project scope.
- Domain Analysis: the activities about understanding the relevant items of this project.
- System Analysis: the activities about analyzing the system requirements.
- System Design and Implementation: the activities about designing the system, studying the technologies for implementation, and implementing the system prototype.
- Application Prototype: the activities about acquiring the relevant data for building the data-driven application prototype, analyzing the acquired data, and designing and implementing the application prototype.
- Project Closure: the activities about finalizing the documentation of the system and transferring the knowledge.
Figure 36 WBS of this project
12.3 Project Planning

The project plan was formulated based on the defined WBS described in the previous section and the period from 8 January 2018 to 31 October 2018. However, some issues led to the change of the project plan during the project execution phase. The project change was confirmed with the stakeholders. The final project plan is shown by using a Gantt chart in Figure 37.

![Figure 37 Final project plan](image)

The initial risk analysis was also carried out in the project planning phase, and it evolved over the project execution period. Please see Section 5.2 for more detail.

12.4 Project Execution

The project execution followed the project plan with monthly project steering group (PSG) meetings to monitor the progress and direction. The execution of the design and implementation phases consists of two parts: the main system and the application prototype used to verify the system.

For the main system, an agile-like methodology was applied to carry out the design and implementation phases. In the methodology, the requirements were handled based on their priority. One plan and review meeting was held every two weeks to make sure the design and implementation were still on the right track.

For the application prototype, a review meeting was held around every month to make sure the results were correct. The review meeting was held not so frequently as the review meetings of the main system because the application prototype was not the focus but only a mean to demonstrate the system.

The plan of meetings (as shown in Figure 38) during the design and implementation phases was organized and sent to the related stakeholders. Each type of meeting has the defined stakeholders as attendees and the interval of meetings. The plan was confirmed with relevant stakeholders.

Besides, there were other ad-hoc meetings to discuss different topics throughout the whole project period.
Figure 38 Meeting plan for the design and implementation phases
Chapter 13  Project Retrospective

This chapter finalizes this report with the reflection on the project from the author’s point of view.

13.1  Reflection

Besides contributing to the company through the project, the project for me was also an excellent opportunity to learn or improve some skills by doing, and my hard and soft skills have been improved during the ten months. For the hard skills, I have gained some knowledge of data science and cloud platforms. For the soft skills, my communication and project management skills have been improved. In addition, I also gained the knowledge of animal breeding as a bonus.

The project started with the focus of a data analysis of the data collected during the production chain, which required the knowledge of data science. I tried to obtain the knowledge from different sources, including the data scientists from the company and some online resources. Then I applied the knowledge to carry out the preliminary data analysis successfully. However, the challenge was to surprise the stakeholders with unknown insights.

In order to satisfy one of the most important requirements about implementation: implementation with standards products on a cloud platform, I learned some products from Microsoft Azure, one of the most popular cloud platforms nowadays. Besides, I also learned how to build a solution with only standard products and no programming. It was a very different experience for me.

I also improved my communication skill over the project. During the meetings in the project, I practiced acquiring relevant information by asking open questions, and I benefited from the practices. However, the communications with the business stakeholders were still a challenge for me at the beginning since they talked in business language instead of the technical language. I realized that using examples, especially real cases, to explain technical things was effective for making them understand. In that way, I retained their interests in the project.

Last but not least, I also received a big improvement in project management skill through some pitfalls. After following the project plan for two months, I had the first delay on one task. The situation became worse after a period of time. The mistake I made was that I did not give each executable task a clear definition of done. Besides, I did not track of project execution regularly. Later, I signaled the situation and corrected the mistakes, and finally, I went back the right track of the project execution. I will keep this project management lesson in mind to make my future projects right.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU/e</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>SAI</td>
<td>Stan Ackermans Institute</td>
</tr>
<tr>
<td>PDEng</td>
<td>Professional Doctorate in Engineering</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>DOC</td>
<td>Day-Old Chick</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>BI</td>
<td>Business Intelligence</td>
</tr>
<tr>
<td>PL</td>
<td>Pure Line, a generation in layer hen production</td>
</tr>
<tr>
<td>GPS</td>
<td>Grandparent Stock, a generation in layer hen production</td>
</tr>
<tr>
<td>PS</td>
<td>Parent Stock, a generation in layer hen production</td>
</tr>
<tr>
<td>BU</td>
<td>Business Unit</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer, an architectural style</td>
</tr>
<tr>
<td>R</td>
<td>A programming language for statistical computing</td>
</tr>
<tr>
<td>CAFCR</td>
<td>A model for system architecting</td>
</tr>
<tr>
<td>HOT</td>
<td>Hatchability of Transferred, a layer hen quality indicator</td>
</tr>
<tr>
<td>PDP</td>
<td>Production Data Platform</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>ADLS</td>
<td>Azure Data Lake Store, a cloud-based storage product aiming to support big data analytic workloads</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement, a commitment between a service provider and a client</td>
</tr>
</tbody>
</table>
Bibliography


Appendix A: Requirements Analysis Result

Figure 39 Requirements analysis result – part one
Figure 40 Requirements analysis result – part two
About the Author

Wei-Sheng Chen received his bachelor's degree in Business Administration from the Faculty of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan in 2006. Then he received his master's degree in Business Administration from the Faculty of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan in 2008. His final thesis project was "GroupNet System Design for Supporting Ubiquitous Learning."

After his graduation, he started working as a software developer for HyWeb Technology and then joined Chunghwa Telecom as a software engineer for six years. He specializes in web systems development and also has experience in native mobile app development. He is enthusiastic about software architecture and implementation.