DACTyL

Athanasios Papakostopoulos
September 2013

DACTyL

Towards providing the missing link between clinical and telehealth data

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

Partners		
	PHILIPS Company	Eindhoven University of Technology
Steering Group	Rian Wouters Helen Schonenberg Ad Aerts	
Date	September 2013	

Contact Address	Eindhoven University of Technology Department of Mathematics and Computer Science MF 7.090, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands +31402474334
Published by	Eindhoven University of Technology Stan Ackermans Institute
Printed by	Eindhoven University of Technology *UniversiteitsDrukkerij*
ISBN	978-90-444-1240-6

Abstract

This document conveys the findings of the Data Analytics, Clinical, Telehealth, Link (DACTyL) project. This nine-month project started at January 2013 and was conducted at Philips Research in the Care Management Solution group and as part of the Data Analysis for Home Healthcare (DA4HH) project.
The DA4HH charter is to perform and support retrospective analyses of data from Home Healthcare products, such as Motiva telehealth. These studies will provide valid insights in actual clinical aspects, usage and behavior of installed products and services. The insights will help to improve service offerings, create clinical algorithms for better outcome, and validate and substantiate claims on efficacy and cost-effectiveness.
The current DACTyL project aims at developing and implementing an architecture and infrastructure to meet the most demanding need from Motiva telehealth customers on return on investment (ROI). These customers are hospitals that offer Motiva telehealth to their patients. In order to provide the Motiva service cost-effectively, they need to have insight into the actual cost, benefit and resource utilization when it comes to Motiva deployment compared to their usual routine care. Additional stakeholders for these ROI-related data are Motiva customer consultants and research scientists from Philips for strengthening their messaging and service deliveries to arrive at better patient care.
The challenge of the current project is that the Motiva data and the hospital data need to be coupled, though they reside in different data silos. In theory a hospital has all the available data at its disposal to answer the return on investment question but in practice it lacks the infrastructure to combine different data sets and the analytical skills to transform the data into meaningful insights. Additionally, every hospital stores data in a different way determined by the available IT infrastructure. The design of a generic system that is able to couple hospital and Motiva data has to deal with the heterogeneity and variety in systems and standards in this domain.
During the domain analysis we identified that a hospital interface engine is the sys-tem responsible for data transfer within a hospital. We found out that hospitals use different clinical vocabularies to categorize hospitalization events. The ICD-10 and SNOMED vocabularies are most often used. The HL7 CDA document is the state of the art standard for clinical data exchange. This standard is not widely supported in the Netherlands and countries that use it create country-specific adaptations. The IHE standardization body creates documents that specify how data should be exchanged in a clinical environment.
We identified five different roles interacting with the system the data contributor, the data integrator, the data presenter, the data analyst and the data researcher. Following that, we grouped the requirements in three different areas of interest connecting, linking and presenting. We implemented three layers to meet the requirements from each area.
The connection layer supports a push communication strategy that decouples the DACTyL from the hospitals infrastructure and leaves the initiative for data contributions to the hospital. The linking layer offers storage areas, databases and data ware-houses, and the mechanism to transfer data between these areas. The presentation layer gives access to the stored data. The access is realized through business intelligence reports.
We used a hospital interface engine to send our test data set to DACTyL. The system successfully accepted the data and made the necessary connections between the Motiva and the hospital data sets. For the first time the end user was able to re-construct a complete time-line containing telehealth and hospitalization events and thus see the “big picture” about the patient. As future work we plan to test DACTyL in a hospital environment. This infrastruc-

ture is a stepping stone towards a system that can integrate different sources of data with data coming from Philips telehealth products.

Keywords telehealth, hospital information systems, Motiva, data warehouse, HL7

Preferred reference A. Papakostopoulos, DACTyL: Towards providing the missing link between clinical and telehealth data. Eindhoven University of Technology, SAI Technical Report, September, 2013, ISBN 978-90-444-1240-6

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

Foreword

Within the DA4HH project led by Steffen Pauws (OOTI alumnus) we created our Data Analytics Framework (DAF). It demonstrates the possibilities of a Business Intelligence solution that allows users (including Philips Motiva Telehealth customers) to create dashboards and reports of Telehealth data, in this case, Motiva data from different locations in a fast, easy and dynamic way. However, one of the major open issues was how to get answers to analysis studies that require hospital data, such as Return on Investment reports, and how to link Motiva telehealth data with clinical outcomes for patient monitoring and research purposes. Since we did not have all the knowledge required, Rian Wouters (OOTI alumnus), came up with the idea to write an OOTI proposal with the goal to investigate this problem and the technologies that can be applied to solve it, and to extent the proof of concept implementation.

We had a few interviews with the OOTI candidates, and we decided that Athanasios Papakostopoulos would be our man (luckily we soon discovered we can call him Thanos). Thanos demonstrated his eagerness by asking for preparatory documents and information far before the formal start of the project. And he would need this eagerness, because it was not an easy assignment, especially in the beginning, where he still needed to do a lot of scoping to get the project plan set.

In the early stage of the assignment he discovered that there is no such thing as a straightforward HL7 interface. There is a whole world of 3-letter abbreviations, different standard versions, file types, architectures, government institutions, healthcare institutions, medical companies, and standardization bodies that together shape the domain of exchanging data between healthcare IT systems based on HL7 standards. In the meantime the proof of concept requirements needed to be defined. It appeared to be quite a challenge to define the requirements within this huge domain of telemonitoring, hospital systems, data warehousing, dimensional modeling, etc. The decisions about how the architecture should look like did not make it any easier, but he found his way through the jungle of standards, frameworks and technologies while being goal-driven and pragmatic when necessary. As if the project was not challenging enough already, it required the use of commercial complex specialist tools that were new to us and he had to learn by self-education and self-dedication.

The work of Thanos resulted in the report that is now in front of you, a working system, and a better understanding of the domain not only for him, but certainly also for us and definitely for others as well. The system is ready for release locally at a Philips customer site or globally at Philips telehealth data centers. In particular, after demonstration to a Philips customer, the customer enthusiasm directly resulted in a follow-up to install, configure and use the system at the customer site. That is the very next thing that we will do. In other words, Thanos' work will lead to direct business impact and improved customer value! In addition, the system and its architectural design were presented to the Principal Architect of the Philips Hospital to Home Innovation Business; Thanos gave him food for thought regarding the business strategy on data management and platforms.

Thanos was a bit worried that the assignment would not provide him enough programming and design tasks for a final OOTI project. By his pragmatism and perseverance, he proved himself wrong.

A few months prior to the end of his assignment, Thanos was offered a job at Philips Healthcare that he accepted. We thank him for the pleasant cooperation of the last 9 months and wish him all the best with the beginning of this new episode of his life in the Netherlands.

Helen Schonenberg & Rian Wouters
September 2013

Preface

This document forms the technical report of Data Analytics, Clinical, Telehealth, Link (DACTyL). This project describes the architecture, design and implementation of a proof of concept analytical system capable of combining external clinical data from hospitals and telehealth data from Philips products in order to provide insights on the data to the end users.

Non-technical readers can read Chapters 1 and 2 to understand the problem and Chapter 4 to find information about the domain. Technical readers should read Chapters 7-9 to understand how the system is designed. The conclusions are summarized in Chapter 12.

The project was carried out by Athanasios Papakostopoulos from the Stan Ackermans Institute Software Technology program of Eindhoven University of Technology. This nine-month project is the author's final project and concludes the two-year Professional Doctorate in Engineering (PDEng) program, known also by its Dutch name Ontwerpers Opleiding Technische Informatica (OOTI). The project was carried out at Philips Research, which is the industrial partner for this study, in Eindhoven.

Athanasios Papakostopoulos
September 2013

Acknowledgements

A number of people contributed through the duration of this project. First and foremost I would like to thank Helen Schonenberg and Rian Wouters my supervisors from Philips Research. On our weekly update meeting they provided guidance and support in the development effort of this project. Helen's feedback and comments had a huge impact in the creation of this document. She had the courage and patience to perform multiple iterations over the chapters and give detailed recommendations.

I would also like to thank Ad Aerts my TU/e supervisor and General Director of Software Technology program for his valuable contribution and support during this project. He provided valuable feedback for the creation of this report and useful advises for project and expectation management. Additionally he gave me the opportunity to present my work in the TU/e Alumni Day 2013.

All the OOTI coaches should be thanked for improving my technical and professional skills in this two year program. Special thanks should go to Maggy De Wert, the OOTI secretary, for being always there for the OOTIs.

I also want to thank the remaining members of the team Steffen Pauws, Jorn op den Buijs and Hossein Nassabi for providing a friendly working environment and relaxing talks during coffee breaks.

A number of "unknown" people contributed to my project by answering my questions in technical forums. I should therefore thank the community of the Oracle Technical Network and the community of the Stack Overflow forum.

Last but not least I would like to thank family and loved ones for their support.

September 2013

Executive Summary

This document conveys the findings of the Data Analytics, Clinical, Telehealth, Link (DACTyL) project. This nine-month project started at January 2013 and was conducted at Philips Research in the Care Management Solution group as part of the Data Analysis for Home Healthcare (DA4HH) project.

The current DACTyL project aims at developing and implementing an architecture and infrastructure to meet the most demanding need from Motiva telehealth customers on return on investment (ROI). These customers are hospitals that offer Motiva telehealth to their patients. To provide the Motiva service cost-effectively, they need to have insight into the actual cost, benefit and resource utilization for Motiva deployment compared to their usual care. Additional stakeholders for these ROI-related data are Motiva customer consultants and research scientists from Philips for strengthening their messaging and service deliveries to arrive at better patient care.

The findings of this project can be grouped in two categories: the domain analysis and the implementation findings. Coupling telehealth data with clinical outcomes is an innovative use case. Analytical systems are already in use in the healthcare domain but according to our literature research there is no system that relates telehealth and clinical data in a large scale.

Hospitals use different IT systems to store data that are potentially interesting for our use case. It is therefore not feasible to predict where data is located at a particular hospital, and how it can be accessed. The greatest common factor between hospital-IT layouts is the interface engine. This system facilitates the data transfer within a hospital and can support communication with external systems. We propose that the communication between DACTyL and the hospital is realized through the interface engine.

Standardization bodies such as the HL7 and the IHE provide standards for interoperability and data transfer across different systems. The adoption level of the standards varies per country. We propose further research to identify if the adoption of an existing standard or a specification of a new standard will benefit DACTyL.

For the envisioned system, we propose a three-layered architecture based on a push communication strategy. The layers we identified are the connection, linking and presenting layer. The connection layer receives, transforms and saves data contributed by hospitals. The linking layer offers storage areas and a mechanism to link Motiva and hospital data. The presentation layer offers access to the data to the potential end users.

To validate our design, we implemented a proof of concept version of DACTyL. This version successfully integrated a data set containing hospitalization data contributed by a hospital interface engine with Motiva data. The system was able to produce reports that compare hospitalizations between Motiva and usual care patients. The end user can evaluate the ROI from the difference in hospitalizations between Motiva and usual care patients. The demo of the system was well received by a customer (hospital) and led to the definition of new case study that will be implemented in the hospital for further evaluation.

We believe that a system capable to couple hospital and telehealth data will not only meet the client requirements on ROI but it will also provide Philips with unique insights and opportunities to further develop prediction models that will lead to better patient care.

Table of Contents

Foreword i

Preface iii

Acknowledgements v

Executive Summary vii

Table of Contents ix

List of Figures xiii

List of Tables xv

1. Introduction 1

1.1 Context *1*

1.2 Project Scope and Goals *2*

1.3 Outline *3*

2. Problem Analysis 5

2.1 Context *5*

2.2 Roadmaps *6*

3. Stakeholder Analysis 7

3.1 Introduction *7*

4. Domain Analysis 11

4.1 Introduction *11*

4.2 Hospital Information Systems *11*

4.3 Healthcare Standards *12*

4.3.1. Health Level 7 (HL7) 12

4.3.2. Integrating Healthcare Enterprise (IHE) 14

4.4 Clinical Vocabularies *14*

4.5 Business Intelligence *15*

4.6 Analytical Systems *16*

4.6.1. Data warehouse and Dimensional modeling 16

4.6.2. Extract Transform Load 18

4.6.3. OLAP cubes 19

5. Feasibility Analysis 21

5.1 Introduction *21*

5.2 Experiment: Communication between DACTyL and Interface Engine *21*

5.3 Risks *22*
5.4 Issues *23*

6. System Requirements 25
6.1 Introduction *25*
6.2 Roles interacting with DACTyL *25*
6.3 Project use case scenario *26*
6.4 Functional Requirements *27*
6.4.1. Connecting Requirements 28
6.4.2. Linking Requirements 28
6.4.3. Presenting Requirements 28
6.4.4. Return on investment report 29
6.4.5. Patient report 29
6.5 Constraints *30*
6.6 Nonfunctional requirements *30*

7. System Architecture 32
7.1 Introduction *32*
7.2 Data Aggregation Architectures *32*
7.2.1. Third Generation Integration 32
7.2.2. Second Generation Integration 33
7.2.3. Conclusions 34
7.3 Refined Decisions *34*
7.3.1. Data Extraction 34
7.3.2. Transport 34
7.3.3. Input interpretation 35
7.3.4. Transformation-Normalization 35
7.3.5. Summary of Refined Decisions 36
7.4 Overview of the system *37*

8. System Design 38
8.1 Introduction *38*
8.2 Component Diagram *38*
8.2.1. Connection Layer 38
8.2.2. Linking Layer 39
8.2.3. Presentation Layer 40

9. Implementation 41
9.1 Introduction *41*
9.1.1. Introduction to Eclipse Modeling Framework 41
9.1.2. Introduction to Teneo 41
9.1.3. Introduction to Oracle Data Integrator 41
9.1.4. Introduction to Oracle Business Intelligence Enterprise Edition 42
9.1.5. Tooling layout 43
9.2 Implementation of the input model *44*
9.3 Implementation of the connection layer *47*

9.4 Implementation of the linking layer *51*
9.4.1. Implementation of the landing area 51
9.4.2. Implementation of the Data Warehouse bus 51
9.4.3. Implementation of the Hospitalization data mart 52
9.4.4. Implementation of the ETL process 54
9.4.5. The “load_dim_patient” interface 54
9.4.6. The “load_dim_diagnosis_group” 55
9.4.7. The “load_diagnosis_group_bridge” 55
9.4.8. The “load_fact_final_hospitalization” interface 55

9.5 Implementation of the presentation layer *55*
9.5.1. Implementation of the OLAP cube 55
9.5.2. Implementation of the data analyst reports 57

10. Verification & Validation 59

10.1 Introduction *59*

10.2 Verification *59*
10.2.1. Connection layer 59
10.2.2. Linking layer 59
10.2.3. Presentation layer 59

10.3 Validation *60*

11. Deployment 63

11.1 Introduction *63*
11.1.1. Implementation deployment 63
11.1.2. Proposed system deployment 64

12. Conclusions 65

12.1 Results *65*
12.1.1. Domain analysis 65
12.1.2. Implementation 65

12.2 Limitations *66*

12.3 Lessons Learned *67*

12.4 Future Work *67*

13. Project Management 69

13.1 Introduction *69*

13.2 Process *69*
13.2.1. Planning and Scheduling 69
13.2.2. Communicating with supervisors 69
13.2.3. Acceptance Control 70
13.2.4. Configuration Management 70

13.3 Work breakdown structure *70*

13.4 Project Retrospective *71*
13.4.1. Strong Points 71
13.4.2. Improvement Points 71

Glossary 73

Bibliography 75

About the Authors 77

List of Figures

Figure 1 – Philips Research inovation history [1] 1
Figure 2 – Project and Product stakeholders.. 8
Figure 3 – Communication with Interface Engine [9]...12
Figure 4 – HL7 RIM Model..13
Figure 5 – Doctor visit modeled in RIM..13
Figure 6 – A particular version of the HL7 CDA document................................14
Figure 7 – Business Intelligence levels..15
Figure 8 – Star schema 17
Figure 9 – A visual represetnation of the data warehouse bus with conformed dimensions 18
Figure 10 – The main steps of ETL [11] 19
Figure 11 – OLAP cube with 3 dimensions..20
Figure 12 – Channels in Mirth Connect..21
Figure 13 – The status of the risks before and after the experiment....................23
Figure 14 – Areas of interest for DACTyL in red and the data sources in blue....25
Figure 15 – Identified roles. The data contributor provides data. The data researcher and analyst receive information and limited data access. The data integrator and presenter configure DACTyL. 26
Figure 16 – Use cases for this version of DACTyL..27
Figure 17 – Transforming different input instances to an instance of the harmonization model prior to saving 36
Figure 18 – Overview of DACTyL...37
Figure 19 – Component Diagram 38
Figure 20 – Presentation (left), business (middle) and physical (right) models...43
Figure 21 – Creating a new report for the data analyst..43
Figure 22 – Tooling layout in DACTyL...44
Figure 23 – Design of the input model 46
Figure 24 – Serialized instance of the input model...46
Figure 25 – Package diagram from components to classes..................................47
Figure 26 – Class diagram showing the factory and the participating interfaces..48
Figure 27 – The parser and the saver handler..48
Figure 28 – Class diagram with Pipe, Request and Handler................................49
Figure 29 –Source code to save an instance of the persistance model.................49
Figure 30 – A Request handled by the Handlers 50
Figure 31 – Saving the data 50
Figure 32 – Conformed dimensions in the data warehouse bus...........................52
Figure 33 – The data warehouse bus and the data marts 52
Figure 34 – Hospitalization data mart foreign keys...53
Figure 35 – The hospitalization data mart 54
Figure 36 – Hierarchy for the diagnosis dimension in the business model...........56
Figure 37 – Hospitalization fact table in the business model 56
Figure 38 – Return on investment report..57
Figure 39 – Patient report...57
Figure 40 – Deployment diagram...63
Figure 41 – Project timeline 70

List of Tables

Table 1 – Stakeholder Overview 7
Table 2 – Information Providers... 8
Table 3 – Example of ICD-10 Hierarchy...15
Table 4 – Analytical vs Operational system [9]....................................16
Table 5 – Simplified example of ICD-10 as a Dimension table 17
Table 6 – Simplified report on hostitalizations......................................20
Table 7 – Risks identified in the beginning [9].....................................22
Table 7 – Connecting Requirements...28
Table 8 – Linking Requirements 28
Table 9 – Presenting Requirements 28
Table 10 – Return on investment report 29
Table 11 – Patient report 29
Table 12 – Constraints...30
Table 13 – Nonfunctional Requirements..30
Table 15 – Hospitalization events data set..44
Table 16 – Hospitalization events data set..45
Table 18 – Summary of the hospitalization data mart 52
Table 18 – Connecting Requirements...60
Table 19 – Connecting Requirements...60
Table 20 – Connecting Requirements...60
Table 21 – Linking Requirements 60
Table 22 – Linking Requirements 60
Table 23 – Linking Requirements 61
Table 24 – Presenting Requirements 61
Table 25 – Presenting Requirements 61
Table 26 – Presenting Requirements 61

1.Introduction

1.1 Context

Philips Research is part of the Philips Corporation. As mentioned in the mission statement, the company aims to "introduce meaningful innovations that improve people's lives [1]." They provide technology options for innovations in the area of health and well-being, targeted at both developed and emerging markets. Philips Research division works on everything from spotting trends and ideation to proof of concept and – where needed – first-of-a-kind product development. The history of Philips Research covers a period of more than 120 years as can be seen in Figure 1.

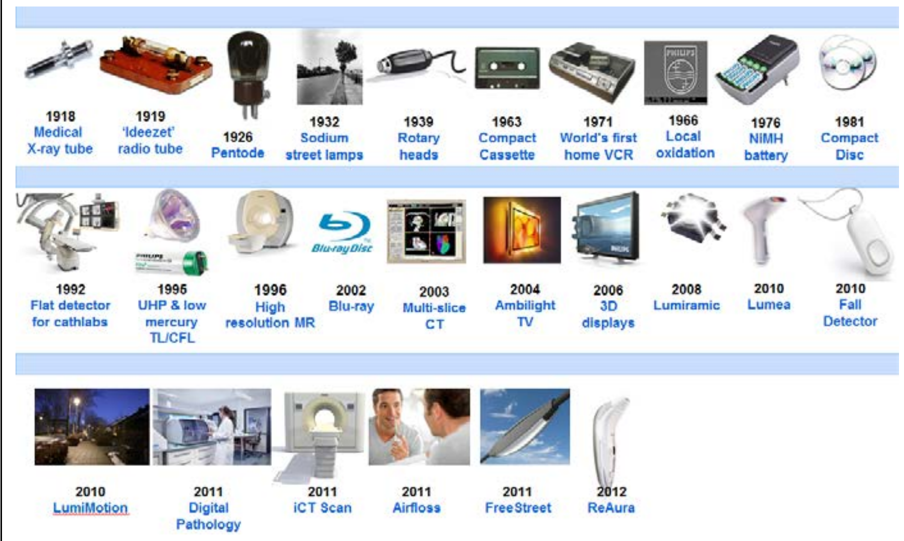


Figure 1 – Philips Research inovation history [1]

The ever increasing cost of healthcare is a modern challenge that affects governments, insurance companies and care providers. Hence, a major challenge in healthcare is to more efficiently provide high-quality care for an increasing number of patients using limited financial and human resources.

One of the actions towards the direction of cost reduction is the increased use of telehealth. Philips Research is actively involved in innovations in the telehealth domain. This domain covers a family of products and services that address health related needs of people in a decentralized way. In Telehealth Systems (TS) the caregiver is geographically separated from the care consumer and the treatment is being individually tailored to the patient's needs. This patient-centered concept of bringing the care from the hospital to the patient at home is expected to result in cost-reduction as it aims to reduce the number of hospitalizations.

In the area of remote patient monitoring, one section of TS, Philips offers the Motiva telehealth system. Using innovative measurement devices, such as weighing scales, blood pressure monitors and glucose meters enhanced with the Bluetooth technology, the vital signs of a patient could be transmitted to a set-top box. That device acts as a medical gateway, as it maintains a connection with a back-end subsystem that collects the health status information of the patient. The operator of the back-end, the so-called "Motiva nurse," has a good overview of the patient's health. The nurse can intervene with the aim to stabilize the patient within a safe zone and prevent the hospitalization of the patient. To improve the efficiency of the system, the developers of the Motiva system introduced the Care Plan, an integral set of messages, reminders, surveys and videos that are used to stimulate the patient into maintaining a healthier life-style. The Motiva system is managed by the Hospital to Home (H2H) business of Philips Healthcare.

One of the challenges the Motiva team is facing is to meet the customer requirements for more elaborate reporting and data access and interaction. Currently reporting for Motiva is done in an ad-hoc way. Motiva engineers manually create queries per customer and execute queries against a replica of the Motiva database. Then, they aggregate the results in reports which are sent by email to the clients. This approach will become infeasible when scaling up the business.

In this spirit, the Care Management Solution group within Philips Research initiated the Data Analysis for Home Healthcare (DA4HH) as a research project to perform data analysis on data coming from home healthcare products with the aim to improve service delivery. At the same time, the project also works on an infrastructure, the data analytics framework (DAF), to offer this type of analysis on an ongoing base. DAF introduced a scalable solution that is able to meet customer reporting requirements, based on Data Warehousing and Business Intelligence (BI) (see Section 4.5). This approach should not only satisfy the customer's requirements, but also allow the data to be used by Philips internally to improve service delivery. Last year the group released a first version of DAF. This analytical system was able to produce dynamic reports from Motiva data.

Currently, telehealth data and clinical outcomes are not linked. The missing link is an infrastructure that is able to combine and store Motiva data and clinical data for reporting and analysis. Even though hospitals have all the necessary data at their disposal to perform the reporting and analysis on their own, they still lack the infrastructure to do so. This link is necessary to meet one of the most urgent customer reporting requirements: a report on the return on invest (ROI) for Motiva. In other words, does Motiva bring a reduction in healthcare cost?

1.2 Project Scope and Goals

The goal of this project was to investigate possible designs for an analytical system that enables external data sources to submit data, combines them with Motiva data and generates specific reports for different stakeholders. The data sources that are relevant for this project are hospital information systems. As an exploratory project it aimed to identify the components that play an important role and document the relevant challenges and limitations.

The Data Analytics, Telehealth, Clinical, Link (DACTyL) system is implemented as a proof of concept to validate the design. This system further extends the features of DAF.

Recapitulating, the goals of the project were:

- Investigate the systems that hospitals use to store patient data and produce taxonomy of these systems.
- Investigate standards used for data transfer.
- Investigate which technology solution can be used to gather data from a hospital information system.
- Design and implement the components that will realize the communication with the hospital information system.
- Design and implement a storage area that will store the incoming data.
- Integrate hospital and Motiva data.
- Create reports that quantify the Return on Investment (ROI) from the telehealth service.

As the time budget for this project is fixed and the scope of such a system is vast, the following issues were kept out of scope since the beginning in order to increase the feasibility of this project.

The out-of-scope technical issues are:

- Anonymity and pseudonymity of incoming data[1].
- Secure communication with the hospitals.
- Secure storage of incoming data.
- Financial soundness of the produced ROI report.
- Legal and ethical aspects of the project

1.3 Outline

Chapter 2 gives an example for our problem and analyses the difficulties. The design criteria that apply are mentioned here.

Chapter 3 introduces the stakeholders and their relationships.

Chapter 4 explains concepts and technologies that are relevant for this project. It begins with an explanation of the fundamental systems in the healthcare domain. Following that, an explanation of health care standards and clinical vocabularies is given. Next the technology and usage of data warehousing are explained.

Chapter 5 documents experiments that took place in the early stages of the project. These experiments contributed to risk identification and assessment. A list of the identified risks can be found in this chapter.

Chapter 6 documents the functional and non-functional requirements of this project. The functional requirements are grouped in three different areas of focus. Additionally the end user reports that the system must produce are documented here.

Chapter 7 documents the system architecture. The chapter starts by enumerating and explaining alternative data aggregation architectures and validating them against the non-functional requirements of Chapter 6. The strategic decisions are documented here.

Chapter 8 describes layers and the components of the system.

Chapter 9 documents the realization of the design. Implementation issues are described here. Also simplifications used to produce the proof of concept deliverable are mentioned in this chapter. The reader can find here class and sequence diagrams and data model diagrams.

Chapter 10 documents how the system is tested and offers a map between the implementation and the requirements.

Chapter 11 presents the current deployment diagram of the system and describes our proposal for the real life deployment.

Chapter 12 documents the conclusions and the future work.

Chapter 13 documents the project management activities. The timeline of the project can be found here.

■

[1] Section 9.2 describes in details the data set used as input. This data set can lead to patient identification.

2.Problem Analysis

2.1 Context

To describe the problem we are going to use a simple example. Mr. Orestis is a heart failure patient who uses the Motiva system at home daily. On Monday and Tuesday he interacts with the systems and submits data. On Wednesday he is transported to the hospital due to an emergency situation. He stays hospitalized for two days and returns back home. The data from Monday and Tuesday are gathered by the Motiva system and the data during the hospitalization are gathered by the hospital information system. No system can reproduce the complete timeline of the patient data by combining the Motiva and the hospital data because the data reside in different systems that do not share information. This project describes how the gap between these datasets can be bridged. Bridging the gap means that there is an infrastructure capable to aggregate the data and combine the appropriately. This infrastructure can then give answers to questions such as the return on investment.

To enable communication with the new data sources the existing data analytics framework (DAF) needed a new extension. DAF linked so far only Motiva data sources. There is a substantial difference between the DAF use case and the DACTyL use case. In the DAF use case the database schema of Motiva is the same across installations and proprietary to Philips. This is the ideal case for an analytical system: identical and stable database schemas with fully know data semantics. The landscape is completely different for hospital information systems. It is unknown what data are available in a particular hospital. This depends on the IT infrastructure present in the hospital and can be different per case. Since hospital information systems have been and are being developed by competitors of Philips, it is highly unlikely that the details about the systems and their data models will ever be shared.

This heterogeneity of hospital systems was the source of complexity for this project. Every hospital is a different use case with particular systems that have certain features. The different data models introduce interoperability issues (see Section 4.3) that need to be dealt with before systems can share their data. It is nevertheless noticeable that Care Delivery Organizations (CDO's) such as hospitals and clinics are starting to make steps towards systems that exchange information [1] [2] [3].

We identify the following design criteria to be relevant to DACTyL

1. Complexity: The DACTyL requires design techniques and technologies that come from the software engineering domain and the data warehousing domain. Apart from the technologies and the tools that come along, it requires knowledge on the domain of IT and information exchange in healthcare. The DACTyL in a real life scenario would require a software engineer, a data warehousing-ETL (see Section 4.6) specialist and a BI specialist. Last but no least it requires consultation from a healthcare IT domain expert. Thus the complexity is not related with algorithmic complexity but with the integration of different technologies.
2. Impact: The system can be used to quantify the impact of Motiva with respect to hospitalizations. This can be then used as a strong selling point for the Motiva system which can lead to increased sales to hospitals. Additionally the system makes available for usage a combined data set of hospital and telehealth data. This creates new opportunities for Philips Research to develop advanced prediction models.
3. Elegance: The design for DACTyL (see Chapter 8) decomposes the system in layers which have clear interfaces between them and can be individually developed. We believe that the design is adaptable to the heterogeneous environment of healthcare IT and we consider this as a proof or elegance.

We identify the following design criteria to be not so relevant for DACTyL

1. Realization: The aspects of total cost of ownership were kept out of scope for this project. The tooling that is used for the solution was prescribed as a constraint (see Section 6.5) and no alternatives where investigated.
2. Genericity: This project is focused on a specific use case concerning the return on investment. The same principles can be applied for other data but this does not mean that the system is able to support any data set out of the box.

The inventiveness design criteria may or may not apply depending on the view of the reader. On one hand analytical systems have existed already for years. The technology these systems use such as data warehouses, ETL tools and OLAP cubes (see Chapter 4), is now mature and well established. Big vendors like Oracle and Microsoft offer powerful toolsets to create these systems. Also in the domain of healthcare analytical systems already exist. In most cases these systems are purpose built for only one hospital. On the other hand we were unable to locate examples of systems that integrate telehealth and hospital data. This can be justified by the fact that telehealth is still not used in a large scale. The use case is therefore quite innovative. The communication components developed in this project make use of web services which is also a mature and well establish technology. We also made use of model driven development techniques to specify data models that play a role in our system.

The work on DACTyL provided answers to the following questions:
1. What kind of data are we interested in adding to the system?
2. Where do these data reside?
3. How can these data reach our analytical system?
4. How can we interpret the foreign data?
5. How can we store them?
6. How can we combine the new data with the Motiva data?
7. How can we present all available data and enable combined reports?

2.2 Roadmaps

Philips aims to deploy telehealth on a large scale across Europe. The Motiva system plays an important role in this effort. To do this, it is essential that customer requirement with respect to reporting and data access are met. Furthermore, to be competitive in the market, it is vital to perform continuous product and service improvement and stay ahead of the competition. A solid framework for data analytics is needed to do this. This does not hold only for Motiva, but also for other home healthcare products.

■

3.Stakeholder Analysis

3.1 Introduction

This chapter introduces the stakeholders for this project. They are distinguished according to their interest and Table 1 presents an overview. The product stakeholders are interested in the features of DACTyL and some may also be interested in documentation artifacts and design decisions. The project stakeholders are additionally interested in the process that produced DACTyL, methodologies applied and time management.

Table 1 – Stakeholder Overview

Name	Interest	Description
Helen Schonenberg	Project, Product	Philips Research R&D Scientist, project member DA4HH
Rian Wouters	Project, Product	Philips Research Software Architect, (former) project member DA4HH
Ad Aerts	Project	General Director of Software Technology program
Steffen Pauws	Product	Philips Research Senior Scientist, project leader DA4HH
Christoph Westerteicher	Product	Philips Healthcare Business Director H2H/Telehealth International
Richard Stubbs	Product	WSD Program Manager, WSD Legacy Manager
David Barrett	Product	Nurse Lecturer in Telehealth University of Hull
NHS	Product	Payer, potential end user of ROI analysis
Insurance Companies	Product	Payer, potential end user of ROI analysis
Hospitals	Product	Data provider and potential end user
Philips Healthcare	Product	Business that develops, maintains and provides Motiva

The stakeholders can be also grouped according to their influence on the requirements of the project (see Figure 2). The immediate stakeholders are involved directly and they have the biggest influence on requirements and end goals. The intermediate stakeholders are not directly involved in the requirements of this project. Nevertheless their requirements are communicated through Helen Schonenberg and Rian Wouters who are immediate stakeholders. They have an interest in the features of the resulting system. The last group contains potential future stakeholders who may influence the system in future releases.

Helen Schonenberg and Rian Wouters interviewed a number of stakeholders the previous year and produced a requirements document for an analytical system designed to meet the reporting requirements of Motiva. Following that they developed a system that provides reporting for Motiva as mentioned in the introduction chapter.

The following list provides additional details on the immediate stakeholders:

- Helen Schonenberg
    - Interest: domain analysis, system design, system implementation, data analysis, future work
    - Input on: use cases, requirements, planning, system architecture, data warehouse design, creation of end user reports
    - Acceptance criteria: System features, technical report
- Rian Wouters
    - Interest: domain analysis, system design, system implementation
    - Input on: use cases, requirements, system architecture, system implementation
    - Acceptance criteria: System features, technical report
- Ad Aerts
    - Interest: proof of candidates' design skills
    - Input on: planning, requirements prioritization
    - Acceptance criteria: System design, project management, technical report

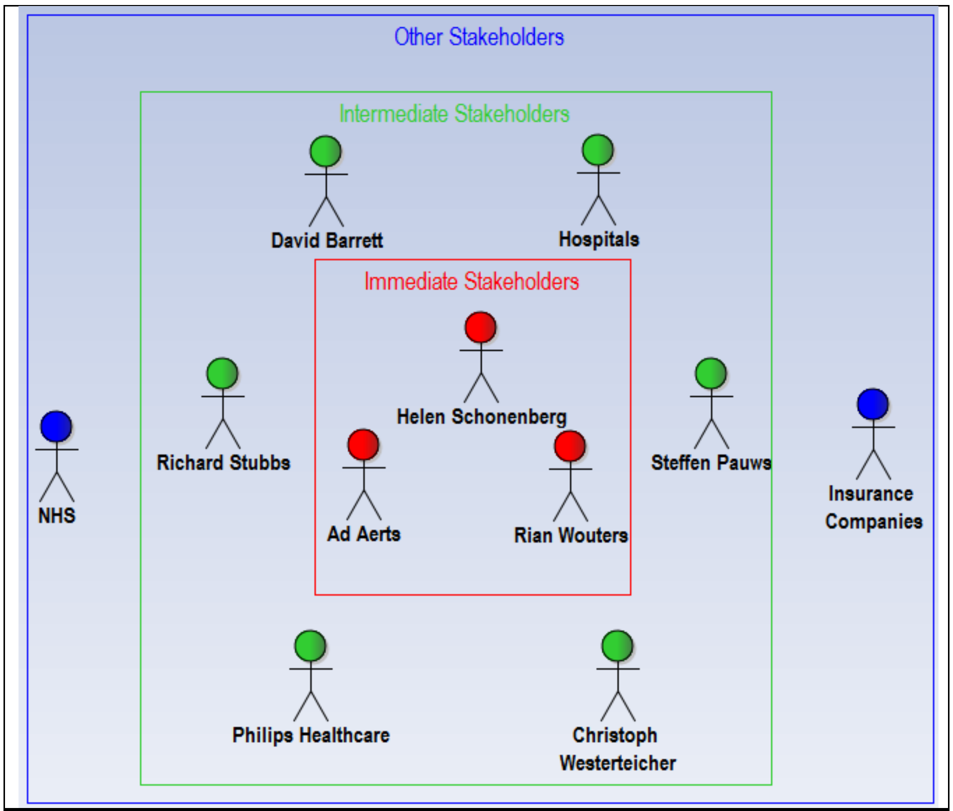


Figure 2 – Project and Product stakeholders

Apart from the stakeholders the project is influenced by other people who with their expertise played the role of external consultants. Table 2 documents the information providers.

Table 2 – Information Providers

Name	Position	Expertise
Charalampos Xanthopoulakis	Philips Research	HL7, NHS
Mohammed Asim	Philips Research	HL7
Nicolaas Arvid	Philips Research	Hospital information systems, interface engines

Nick Ebbers	Philips	Technical specialist Motiva
Johan Gustav Belika	University of Valencia	Disease surveillance systems
Zorg & ICT 2013	Epic, Siemens, Mc Kes-son, e-Novation, Chip-soft, CSC	Hospital Information systems vendors

▪

4.Domain Analysis

4.1 Introduction

As described in Chapter 2, collecting and analyzing data from healthcare systems is a known challenge. In this chapter we briefly introduce components and domain concepts that have a direct relationship with the project. We touch upon hospital information systems (Section 4.2), healthcare standards (Section 4.3), clinical vocabularies (Section 4.4) and analytical systems (Section 4.5).

4.2 Hospital Information Systems

Traditionally data in the healthcare domain were kept in paper-based records. The introduction of information systems changed the way data are stored. The term **Electronic Medical Record** (EMR) was introduced to mark the change from paper to computer-based data records. An EMR is a computerized medical record created in a CDO, such as a hospital or physician's office. The data in the EMR is the legal record of what happened to the patient at the CDO. EMRs tend to be a part of a local stand-alone health information system that allows storage, retrieval and modification of records.

Even though the term refers to the data record, in practice, it is often used to identify the system that contains the record. The EMR systems are used by care delivery organizations to perform various tasks and assist in different ways in the care delivery process. The EMR system is an application environment composed of multiple components. The core of the system is the clinical data repository. This area permanently stores all available data. The data model used for storage is designed by the system vendor. These systems are designed as stand-alone systems and their main goal is to substitute paper-based data keeping. The EMR system plays an important role for this project because it is the source location of the clinical data we aim to use [4].

Another important domain term is **Electronic Health Record** (EHR). The term describes a record in digital format that is theoretically capable of being shared across different CDO’s. Once again the term is often used to describe the system that contains the record. An EHR system may include data such as demographics, medical history, medication and allergies, immunization status, laboratory test results, radiology images, vital signs, personal statistics such as age and weight, and billing information. The sharing of information supported by the system can occur by way of network-connected enterprise-wide information systems and other information networks or exchanges [5].

The terms EMR and EHR are often used interchangeably. Technically, there is a distinction, but it is one that is been blurred by common usage. At a minimum, EMR systems replicate all aspects of paper charting. They are designed to facilitate all the documentation, lab results, visit notes, diagnostic test results, insurance information, demographics, health histories, medication information, and more. EHR systems, on the other hand, are essentially advanced EMR systems with the functionality for greater electronic exchange; that is, they may be able to follow patients from practice to practice and allow for things like data exchange and messaging between physicians [6].

A common scenario for CDOs is that over a period of time different information systems have been purchased or developed in house to address the ever evolving needs of healthcare. This results to a fragmented landscape of department-specific stand-alone systems. The communication gap between these systems is bridged by another utility system usually called **communication broker** or **interface engine**. An interface engine is a CDO's "telephone exchange" for clinical data, ensuring that information passes smoothly and quickly from one hospital system to another. It is designed to simplify the creation and management of interfaces between separate applications and systems within an organization. Interface engines undertake messaging between systems, and normally manage any mapping, translation and data modification necessary to ensure the effective exchange of data around the organization. Rather than connecting all systems to each other individually, a highly complex, time consuming and unreliable process, an interface engine acts as the intermediary for all messaging between hospital systems, as illustrated in Figure 3. The interface engine is an important system for this project because it is able to create a communication path between two systems that want to exchange data.

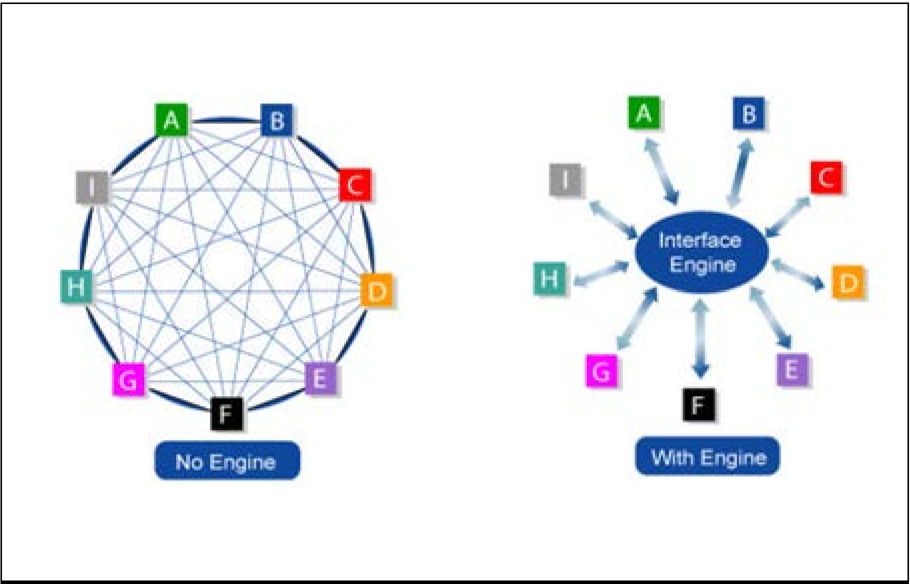


Figure 3 – Communication with Interface Engine [9]

4.3 *Healthcare Standards*

As soon as two systems need to exchange data the concepts of syntactic and semantic interoperability become important. When two systems successfully exchange data they are exhibiting **syntactic interoperability**. The receiving system is able to accept the incoming data because they arrive in the appropriate form according to a set of predefined rules. **Semantic interoperability** extends the syntactic one. The receiving system is able to automatically interpret the incoming message and use the information contained in an appropriate way. The difference is demonstrated in the following example: Willem from the Netherlands asks, in Dutch, Orestis from Greece, what is the time. The two "systems" exhibit syntactic interoperability since Orestis ear successfully accepts the information packaged in the air waves. Unfortunately Orestis cannot interpret the message. When Willem asks again in English Orestis responds. At this point the two "systems" achieve semantic interoperability since they used the English language as a standard.

4.3.1. Health Level 7 (HL7)

The task of standardization in healthcare is done by various organizations. The leading position belongs to **Health Level 7** (HL7) [10]. HL7 is an U.S.-based, ANSI-accredited health information standards development organization. Its specifications are mostly for application-level messaging among hospital information systems (HIS). Other recent areas of interest include the structure and content of clinical documents and decision support. There are two major working versions of HL7 standards, version 2 and version 3 in addition to some minor ones. The 2.4 version is by far the most widely implemented standard in health informatics worldwide. The main goal of HL7 v2 was to standardize messaging between HIS and achieve syntactic interoperability. Thus there is no guarantee for semantic interoperability when using HL7 v2.4 because there is no well-defined underlying information model.

The third version of HL7 is still aimed primarily at defining application messages, but now using a well-defined information model, the **Reference Information Model** (RIM) shown in Figure 4. The core classes of RIM are shown in blue and the first level of subclasses is shown in white. While assembling messages, the content schemas are derived by a restriction process starting from the RIM, further constrained by **domain information models** (DIM), **restricted message information models** (RMIM) and **common message element types** (CMET). The process ends with forming **hierarchical message definitions** (HMD) and the generated message schemas are represented as XML documents.

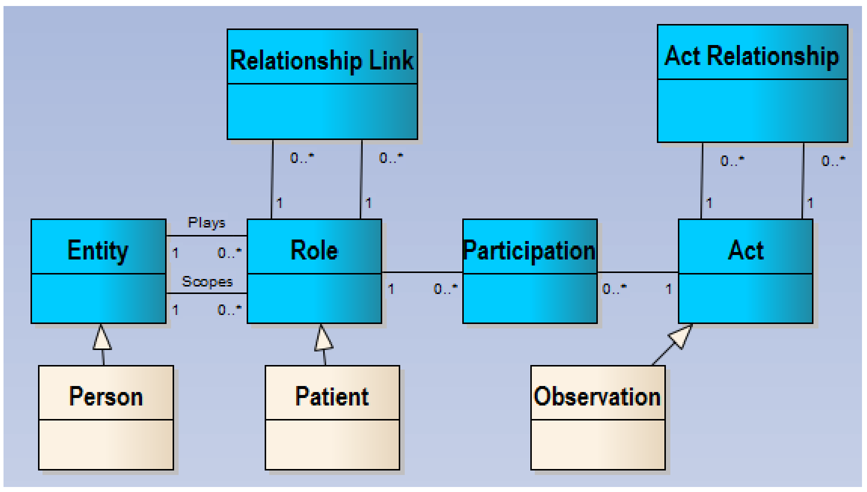


Figure 4 – HL7 RIM Model

To demonstrate how the RIM models a real life event, consider the scenario of a patient visit to a doctor where the patient's pulse rate is measured. Patient Willem is an instance of the class Person and plays the role of a diabetes patient. The doctor visit is an instance of class Act and the pulse measurement is an instance of class Observation. The patient is linked to the visit by an instance of the class Participation and the pulse measurement is linked to the visit though an instance of class Act Relationship. The object diagram in Figure 5 shows the relationships.

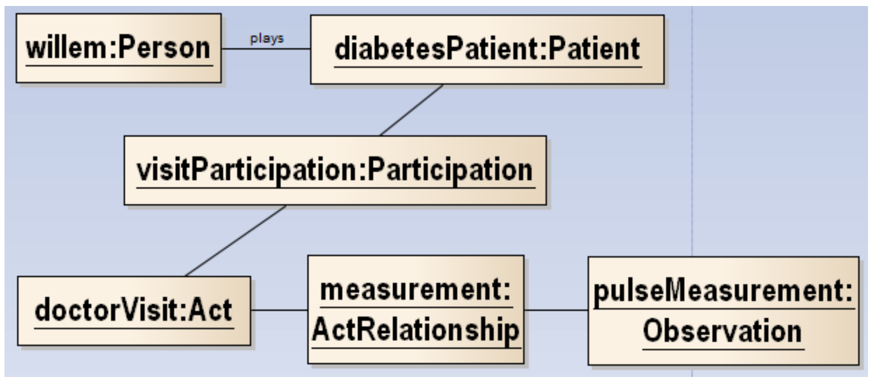
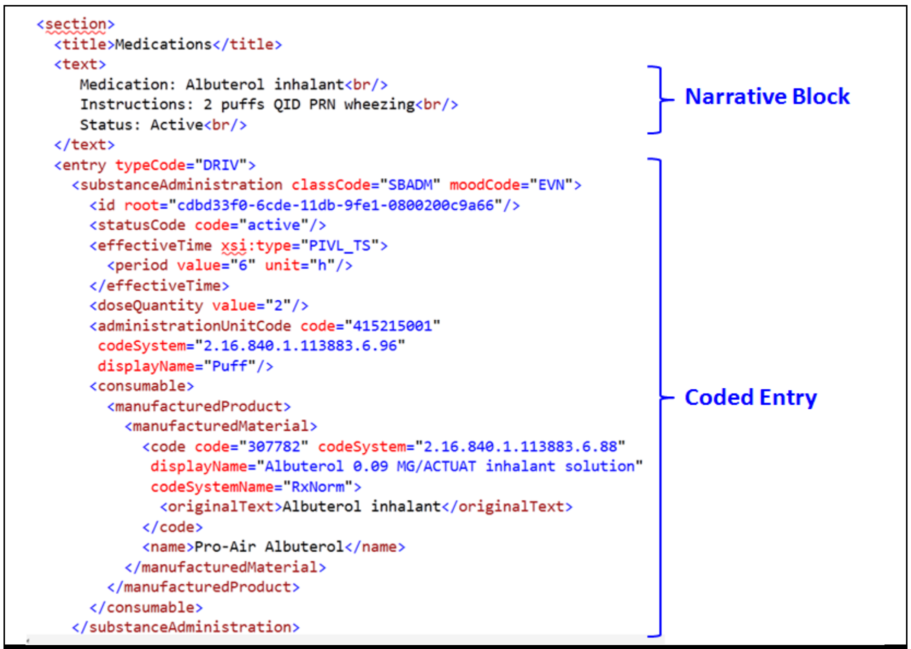

willem:Person
plays
diabetesPatient:Patient
visitParticipation:Participation
doctorVisit:Act
measurement: ActRelationship
pulseMeasurement: Observation

Figure 5 – Doctor visit modeled in RIM

One of the standards that HL7 has produced is the **Clinical Document Architecture** (CDA). CDA is an XML-based markup standard intended to specify the encoding, structure and semantics of clinical documents for exchange. An example is shown in Figure 6. Its semantics is derived from the HL7 RIM and uses the HL7 Version 3 Data Types, which are also part of the RIM. CDA document content is intended to be human-readable and supporting narrative text, yet still having some structure and allow for medical coding to represent concepts in a computable manner. CDA is HL7's proposed way to achieve semantic interoperability. From a software point of view the RIM and the CDA document are abstract domain models. They are published in the form of UML diagrams by HL7. The CDA specification is intentionally abstract so that the implementors can define their own flavor of the CDA doucment which serves their domain specific needs. Different organizations use different implementations of the CDA specification such as the **Continuity of Care Document** (CCD). These implementations are restrictions of the original abstract model. CDA is important for this project as because because it is the state of the art standard design to provide semantic interoperability [10].



```
<section>
  <title>Medications</title>
  <text>
     Medication: Albuterol inhalant<br/>
     Instructions: 2 puffs QID PRN wheezing<br/>
     Status: Active<br/>
  </text>
  <entry typeCode="DRIV">
    <substanceAdministration classCode="SBADM" moodCode="EVN">
      <id root="cdbd33f0-6cde-11db-9fe1-0800200c9a66"/>
      <statusCode code="active"/>
      <effectiveTime xsi:type="PIVL_TS">
        <period value="6" unit="h"/>
      </effectiveTime>
      <doseQuantity value="2"/>
      <administrationUnitCode code="415215001"
       codeSystem="2.16.840.1.113883.6.96"
       displayName="Puff"/>
      <consumable>
        <manufacturedProduct>
          <manufacturedMaterial>
            <code code="307782" codeSystem="2.16.840.1.113883.6.88"
             displayName="Albuterol 0.09 MG/ACTUAT inhalant solution"
             codeSystemName="RxNorm">
              <originalText>Albuterol inhalant</originalText>
            </code>
            <name>Pro-Air Albuterol</name>
          </manufacturedMaterial>
        </manufacturedProduct>
      </consumable>
    </substanceAdministration>
```

Figure 6 – A particular version of the HL7 CDA document

4.3.2. Integrating Healthcare Enterprise (IHE)

Another important standardization organization is **Integrating the Healthcare Enterprise** (IHE) [11]. IHE is an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information. IHE promotes the coordinated use of established standards such as HL7 to address specific clinical needs in support of optimal patient care. Systems developed in accordance with IHE have better communication, are easier to implement, and enable care providers to use information more effectively. IHE Profiles organize and leverage the integration capabilities that can be achieved by coordinated implementation of communication standards, such as HL7, W3C and security standards. They provide precise definitions of how standards can be implemented to meet specific clinical needs. The IHE profiles are relevant for this project as they are specifications for data exchange [8].

4.4 Clinical Vocabularies

Medicine is one of the few domains where extensive domain knowledge is defined through controlled vocabulary. Clinical vocabularies are in fact another type of healthcare standard. They are published and maintained by organizations and usually have a specific area of focus.

The vocabulary used in this project is the 10th revision of **International Classification of Diseases** (ICD-10). ICD was originally published by World Health Organization for classifying and coding of mortality cases [12]. Other uses include establishing a common naming and description of diseases and collection of comparable data for epidemiologic and healthcare management studies. The vocabulary is organized in three hierarchical levels: Chapters, blocks and codes. An example can be seen in Table 3. ICD-10 is relevant for this project as it is used to relate a hospitalization event with a particular disease. This clinical vocabulary is also visible in the report that the system produces.

Table 3 – Example of ICD-10 Hierarchy

Chapter	Block	Code
Diseases of the circulatory system	Chronic rheumatic heart diseases	I05.0 Mitral stenosis
		I05.2 Mitral stenosis with insufficiency
	Hypertensive diseases	I11.0 Hypertensive heart disease with (congestive) heart failure
		I12.0 Hypertensive renal disease with renal failure

4.5 Business Intelligence

Business Intelligence (BI) is the process of transforming data to useful information for decision makers. There are different levels in business intelligence as shown in Figure 7. The higher levels of BI require additional tooling support such as statistical tools. BI makes use of analytical systems to transform the data to insights.

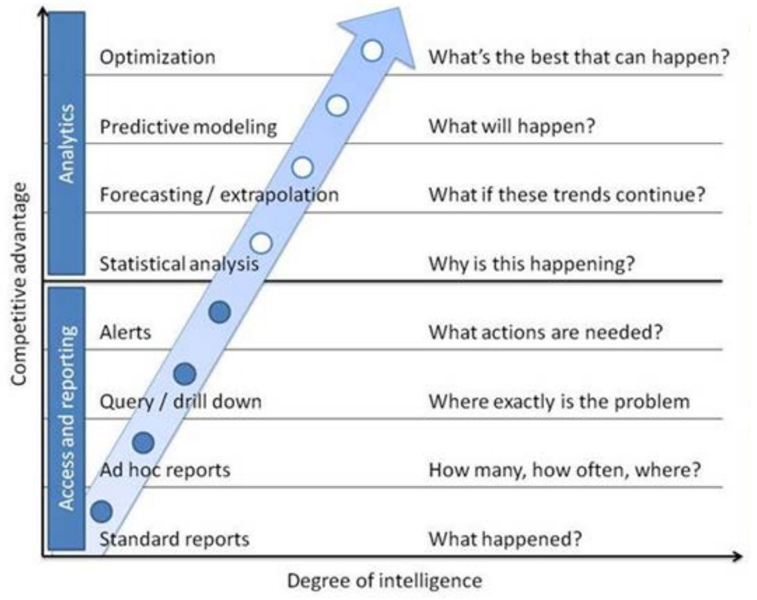


Figure 7 – Business Intelligence levels

4.6 Analytical Systems

This section provides an introduction to analytical systems and the technologies used to realize them. The main terms explained here are used later in this document.

Analytical Systems enable the evaluation of business processes such as hospitalizations. The differences between an analytical system and an operational system are summarized in Table 4. The users of such a system are decision makers. They interact with the system to gain insights about business processes. Based on the information gathered from the system they can take actions to improve the business process. Since this project is developed in a healthcare related environment, we will attempt to make an analogy between the human body and an analytical system.

Table 4 – Analytical vs Operational system [9]

	Operational	Analytical
Purpose	Execution of business process	Measurements of a business process
Primary interaction Style	Insert, Update, Query, Delete	Query
Scope of Interaction	Individual transaction	Aggregated transactions
Query Patterns	Predictable and stable	Unpredictable and changing
Temporal Focus	Current	Current and historic
Design Optimization	Update concurrency	High-performance query
Design Principle	Entity-relationship (ER) design in third normal form (3NF)	Dimensional design (Star schema or Cube)
Also known as	Transaction System On Line Transaction Processing (OLTP) System Source System	Data Warehouse System On Line Analytical Processing System (OLAP) Data Mart

4.6.1. Data warehouse and Dimensional modeling

The data that the analytical system uses are stored in **Data Warehouses** (DW). A DW is a collection of tables in a database system. The DW is the body of the analytical system. According to [10] a DW is designed using a dimensional design also termed as **star schema**. A dimensional data model is a particular way to structure data in a database, optimized for querying. The database tables in a dimensional model can be grouped in two categories: fact and dimension tables. Dimensional models are also called cubes or star schemes due to their appearance (see Figure 8).

Fact tables are the core of the star schema as shown in Figure 8. They capture facts about an event for a particular business process such as a hospitalization event. They contain measurements and foreign keys to dimensions. The foreign keys to dimension tables add context to the particular event.

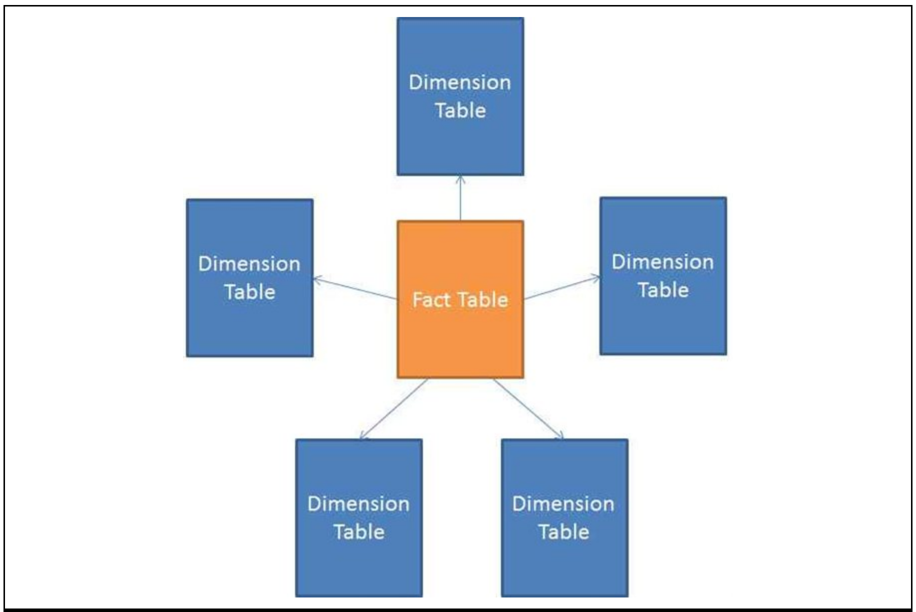


Figure 8 – Star schema

Dimension tables define a context space. Every row in a dimension table defines a point in that context space. For example we can model the ICD-10 vocabulary as a dimension. This vocabulary defines the space of diseases. The points of this context space are the ICD-10 codes. Every disease is represented by a row in the dimension table. As we discussed earlier, this vocabulary is organized in three hierarchical levels. This hierarchy is also physically present in the dimension table. It is achieved by duplicating dimension attributes as shown in Table 5. This table shows that dimensions are de-normalized tables.

Table 5 – Simplified example of ICD-10 as a Dimension table

Chapter	Block	Code
Diseases of the circulatory system	Chronic rheumatic heart diseases	I05.0 Mitral stenosis
Diseases of the circulatory system	Chronic rheumatic heart diseases	I05.2 Mitral stenosis with insufficiency
Diseases of the circulatory system	Hypertensive diseases	I11.0 Hypertensive heart disease with (congestive) heart failure
Diseases of the circulatory system	Hypertensive diseases	I12.0 Hypertensive renal disease with renal failure

A collection of fact and dimension tables that focuses on a particular business event is often termed as a **data mart**. For example, for DACTyL we developed a new data mart (see Section 9.4.3) that captures hospitalization events. Analytical systems can combine data coming from different data marts and thus provide unique insights into relations between these events. One approach to combine the data marts is applying the bus architecture described in [10]. The bus architecture relies on the concept of conformed dimensions. These dimension tables are shared across multiple data marts. Figure 9 shows a data warehouse bus. This "data bus" contains amongst others, three conformed dimensions the "Patient", "Calendar" and "Location". These conformed dimensions are referenced by foreign key relationships from fact tables contained in two different data marts "Hospitalizations" and "Motiva Tasks". According to [10] this situation looks like a data bus and the data marts are attached to the bus due to the foreign key relationships with the conformed dimensions.

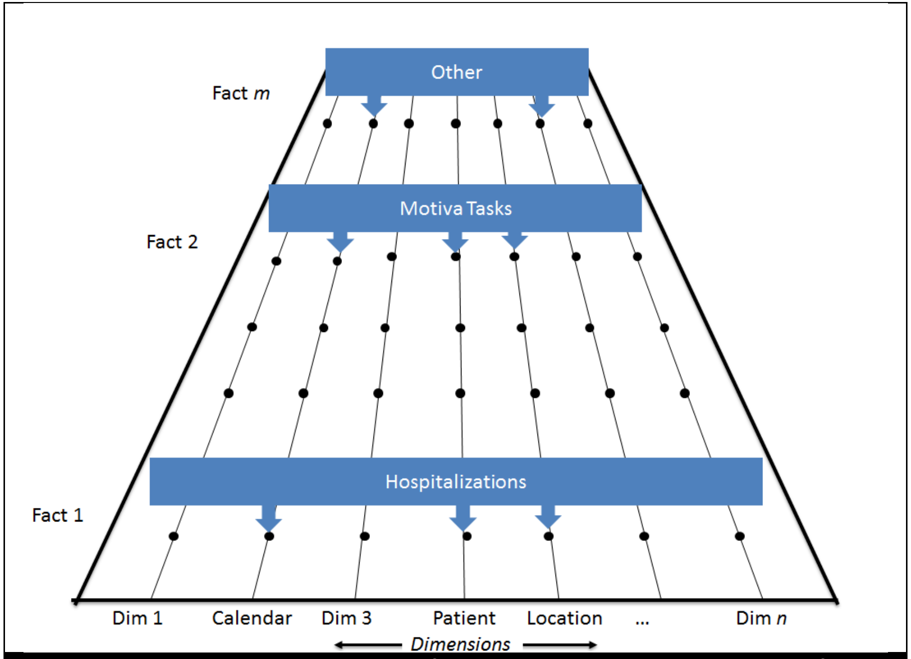


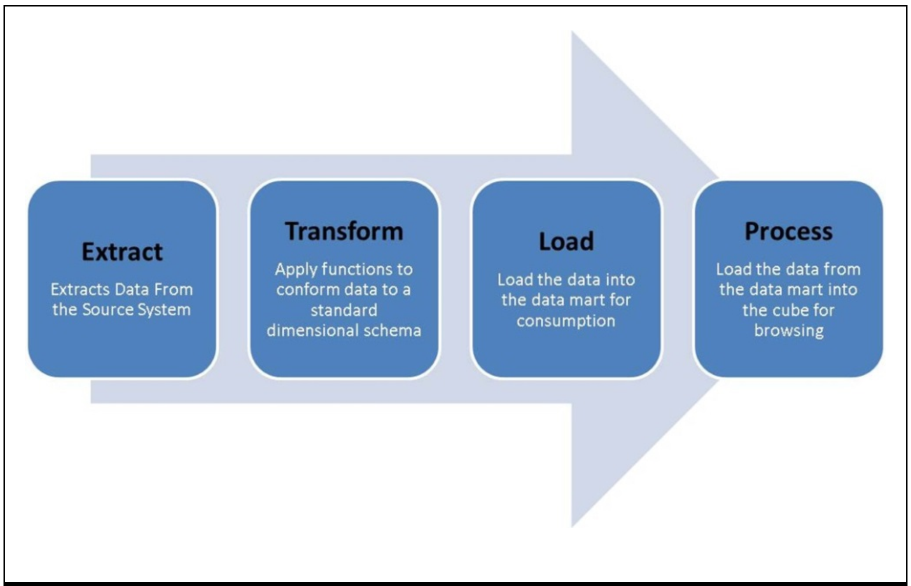
Figure 9 – A visual represetnation of the data warehouse bus with conformed dimensions

The reporting requirements for particular business events are the main driving force for the dimensional design. The structuring of data in dimensions and facts simplifies and improves the performance of the SQL queries needed to produce reports.

4.6.2. Extract Transform Load

What makes analytical systems powerful is their ability to aggregate and combine data from different sources. These sources can be operational systems, files and other streams of data. All data are stored in the DW. The process of data collection and storage is termed **Extract Transform Load** (ETL)[2]. Depending on the case, the ETL can be a collection of SQL scripts, bash scripts or any combination of techniques that extracts data from one location and delivers them to a different location. Software vendors provide dedicated tooling that facilitates the creation and maintenance of ETL processes. The ETL process should be considered as the heart and the circulatory system of an analytical system because it pumps new data into the data warehouse and guides the data flow. Figure 10 visualizes the different steps involved in ETL. Additionally it shows what comes after the ETL which is discussed in the next section.

[2] Oracle also uses the term ELT for one of their tooling solutions.


Extract
Extracts Data From the Source System
Transform
Apply functions to conform data to a standard dimensional schema
Load
Load the data into the data mart for consumption
Process
Load the data from the data mart into the cube for browsing
Figure 10 – The main steps of ETL [11]

4.6.3. OLAP cubes

Since we identified the body and the heart we are still missing the brain. The brain of an analytical system is the **Online Analytical Processing** (OLAP) cube. This cube is a data structure that indexes the available data from the fact tables of the DW. Figure 11 shows an example of a cube with three dimensions; calendar, patient and diagnosis. The OLAP cube is an *n*-dimensional cube in the general case but it is easier to visualize an example with three dimensions.

Every intersection of three dimension points is a fact, represented by a row in a fact table. Figure 11 could contain facts about patient hospitalizations. If we request from the cube data for patient “Orestis” with diagnosis code “I12.0” at calendar date “11-02-2012” the cube will return a single fact row with the hospitalization event if such a row exists. With this in mind the cube can be seen as an alternative representation of a fact table. What makes the cube more complicated is that it stores all data from every available fact table.

Things become more interesting when the cube receives a request with fewer dimension points. As an example consider the top left cube in Figure 11. The cube returns all data that have ‘Orestis’ as a dimension point. It treats the missing dimension points for calendar and diagnosis as wildcards. This is termed “slicing”.

The bottom case shows a more advanced operation that is related with the dimension levels in a hierarchy. The dimension is a physical database table with a number of columns. Any dimension column attribute can play the role of a dimension point. For instance, the columns in the row for date “11-02-2012” may also capture the year, month and day in different representations, as well as day number, week number, day of the week, and so on. In the example, the cube receives a query with dimension points “gender=M,” “chapter=Diseases of the circulatory system” and “month=April 2008.” The “month=April 2008” attribute certainly exists in a number of dimension rows in the calendar dimension. The same holds for the other attributes also. The result, as can be seen in Figure 11, is a smaller cube and this is termed “dicing.”

If the user edited the request and instead of “month=April 2008” specified a dimension point “year=2008,” he would receive a more general result as an answer and this is termed “drilling up.” If he specified “calendar week=15” he would get a more detailed result and this is termed “drilling down.”

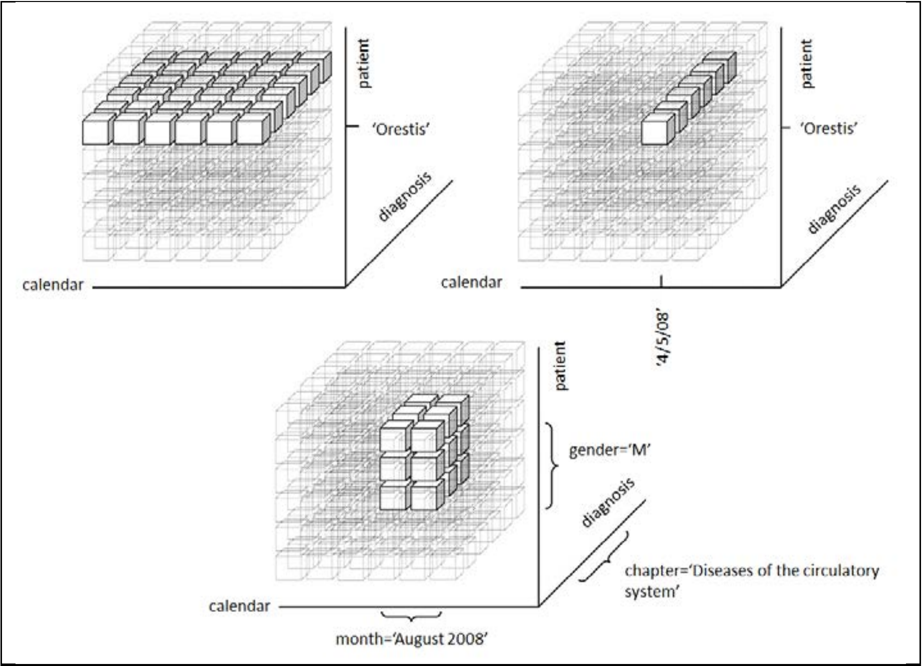


Figure 11 – OLAP cube with 3 dimensions

Table 6 shows what an end user report could look like. In this example hospitalization events are measured. The column "Hospitalizations" contains the aggregated result. In Table 6 the measurement "hospitalizations" is summarized. For other use cases the average, minimum, maximum, of a measurement can be used.

Table 6 – Simplified report on hostitalizations

Time	Diagnosis	Patient	Hospitalizations
August 2008	Diseases of the circulatory system	M	120
September 2008	Diseases of the circulatory system	M	100

Software vendors provide tools that simplify the creation of OLAP cubes out of DW. The cube acts as an interface between the reports used by the user and the actual DW. This is how software vendors sell dash boarding tools that can work on top of any physical DW. The OLAP cube requires upfront configuration and then it is intelligent enough to translate the user actions to DW specific queries.

■

5.Feasibility Analysis

5.1 Introduction

In this chapter we explain the experiment that validated our findings from the domain analysis. This experiment is a composition of different smaller experiments but we choose to present it as one complete experiment. Next, we explain the risks identified in the beginning of the project and show what the impact of the experiment on the risks was.

5.2 Experiment: Communication between DACTyL and Interface Engine

To validate the findings of domain analysis the following scenario was tested: A hospital and DACTyL need to communicate. The hospital was simulated by a file containing the hospital data and an interface engine. The file and the interface engine represent the hospital IT infrastructure. The interface engine will read the file and send the data to DACTyL via http. DACTyL will be simulated by a web server exposing a public interface. We choose to use a hospital interface engine as it was the greatest common factor between different hospital-IT layouts.

We used the **Mirth Connect** (MC) interface engine [12]. MC is an open source interface engine. It enables users to create channels between endpoints. The endpoints can be files, databases and web services. Additionally, it is also possible to define transformations within a channel. The transformations enable hospitals to extract data from one system adapt them and deliver them to another system in the appropriate format. The architecture of a channel in MC is shown is Figure 12.

For this experiment the source endpoint is the file that contains the hospital data. The target endpoint is a web service published by DACTyL. We need to create and configure a channel between the two endpoints and specify the transformations. The web service expects an xml document as an input with a particular structure. The channel must therefore transform the data from the file to a proper xml document and send it.

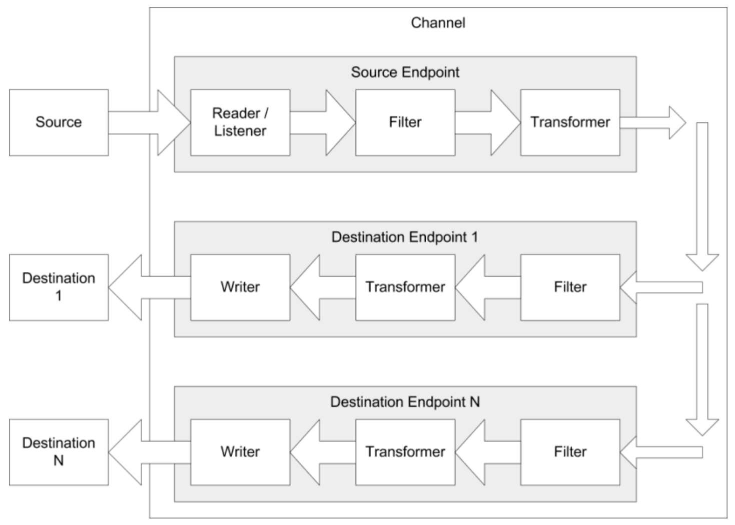


Figure 12 – Channels in Mirth Connect

After the channel creation and setup we were able to achieve the desired behavior. The data that were located in a file reached the web server in the form of an xml document.

Conclusions

1. The interface engine can connect the hospital information system with an external system through a web service.
2. The interface engine is capable of restructuring the hospital data according to an input format.
3. The interface engine requires a onetime configuration to periodically send data to the external system.

This experiment is important for a number of reasons. First it provided hands on experience with the features of a mainstream interface engine. Based on the experiments with MC we were able to understand how data flow is managed in a hospital environment. Second we proved that a hospital can adapt the data to a particular format by configuring his interface engine. This is important because it leads to the conclusion that the choice of the input model for DACTyL is not bound by the exporting features of the hospital information system. The interface engine has the necessary features to extend the exporting capabilities of a hospital information system.

5.3 *Risks*

The initial risks identified for DACTyL are shown in Table 7 sorted on impact.

Table 7 – Risks identified in the beginning [9]

ID	Risk Name	Potential impact on Project success Low/High	Potential likelihood of Occurrence Low/High	Mitigation plan
R2	Changing project scope	H	H	Prioritize requirements, agree on realistic goals
R4	Lack of ODI experience	H	H	Create small experiments, build simple ETL scenarios
R5	Impact of hospitalization updates	H	H	Create small experiments early
R6	Complexity of integration with Motiva DW	H	H	Migrate to production system early
R8	Lack of data warehouse experience	H	H	Create simple data marts early, stick to documented approaches
R9	Lack of time	H	H	Drop requirements
R1	No access to real hospital information system	L	H	Simulate system using open source alternatives
R3	Lack of real clinical data	L	H	Use legacy data from Philips and/or fabricate more data using data

				generators keep track of assumptions
R7	Lack of experience with BI tools	L	H	Create small experiments and simple reports

The feasibility experiment enabled the validation of a number of assumptions and helped in developing some experience with the related tools. Figure 13 shows the effect of the experiments on the risks.

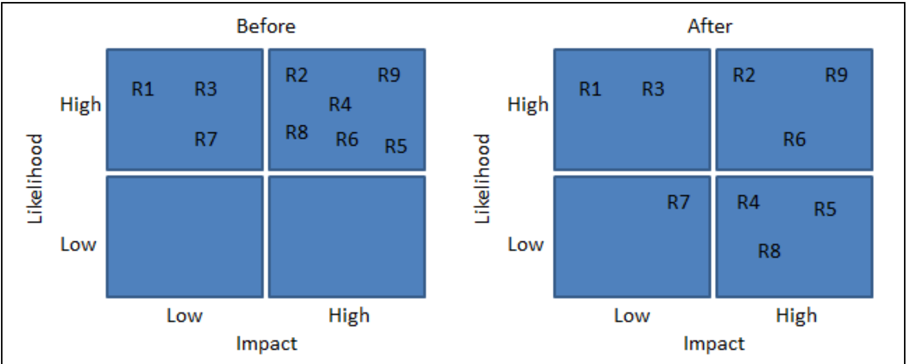


Figure 13 – The status of the risks before and after the experiment

5.4 *Issues*

As mentioned in the introduction chapter a number of issues were kept intentionally out of scope for this project. These issues can be grouped in two categories:

1. Security issues: The security issues cover requirements that have to do with secure transfer and storage of data. Additionally because we are dealing with sensitive patient data requirements for anonymity and pseudonymity may arise in the future.
2. Financial soundness of the reports: Calculating a sound return on investment report from a Motiva service requires combination of many different data sets. In this project we are only looking on hospitalization events and from these events we try to deduct the effect of the Motiva service.

-

6. System Requirements

6.1 Introduction

This chapter describes the requirements for DACTyL. The main sources of requirements are the immediate stakeholders from Philips. Additionally Philips already has a requirements document that specifies the reporting needs for the Motiva product. Requirements from this document combined with the feedback from the immediate stakeholders led to the specification of the functional and non-functional requirements. The functional requirements are grouped in three different areas of interest: connecting, linking and presenting. The connecting area covers requirements related to data transportation from external systems to DACTyL. The linking area covers requirements that assure the linking of different information sets. The presenting area covers requirements related to presentation of the available information. Figure 14 shows the three different areas of interest depicted in red and the data sources in blue.

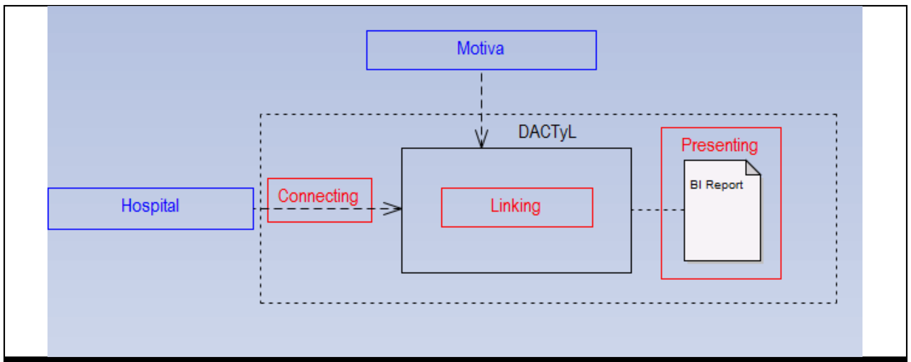


Figure 14 – Areas of interest for DACTyL in red and the data sources in blue

6.2 Roles interacting with DACTyL

During the early stages of the project five different roles were identified to interact with the envisioned system. These roles can be performed by people or systems depending on the case. The following section explains the roles and provides one general user story as an example.

- **Data Analyst**: The Data Analyst wants to analyze the available data periodically using a set of reports. He uses always the same methods to perform his analysis. A real life example can be a finance manager from a hospital who has to calculate the ROI from the Motiva telehealth service every quarter.
  - "*I as a Data Analyst want to use a dynamic report to find the data I am looking for.*"

- **Data Researcher**: The Data Researcher also wants to analyze the data but he does this an ad hoc basis using different methods. A real life example can be a researcher in Philips who wants to test his new prediction model with the available data in the DACTyL.
  - "*I as a Data Researcher want to use the available data to validate my prediction models.*"

- **Data Contributor**: A hospital who wants to contribute data periodically. The hospital has knowledge of the local data models and access to the data. The role can be played by a person or a system at a hospital.
  - "*I as a Data Contributor want to submit part of my data to DACTyL to make them available for analysis.*"

The complexity and configuration needs of such a system lead to the need to keep two configuration roles under scope. These roles understand parts of the DACTyL and they can modify its behavior to fulfill new needs of the end users.

- **Data Integrator**: Maintains and upgrades DACTyL. Implements the process that loads data to the data warehouse. He wants to create new data marts to meet upcoming reporting requirements.
    - *"I as a Data Integrator want to create and load data marts with the available hospital data to enable new reporting capabilities."*

- **Data Presenter**: Configures the reporting tools that provide access to the data warehouse. Creates reports for the Data Analyst.
    - *"I as a Data Presenter want to create dynamic reports to fulfill the requirements of the Data Analyst."*

The data presenter can also be a data analyst. The end user can be provided with already configured reports but additionally can be granted rights to interact with DACTyL and create his own reports. An overview of the identified roles is presented in Figure 15.

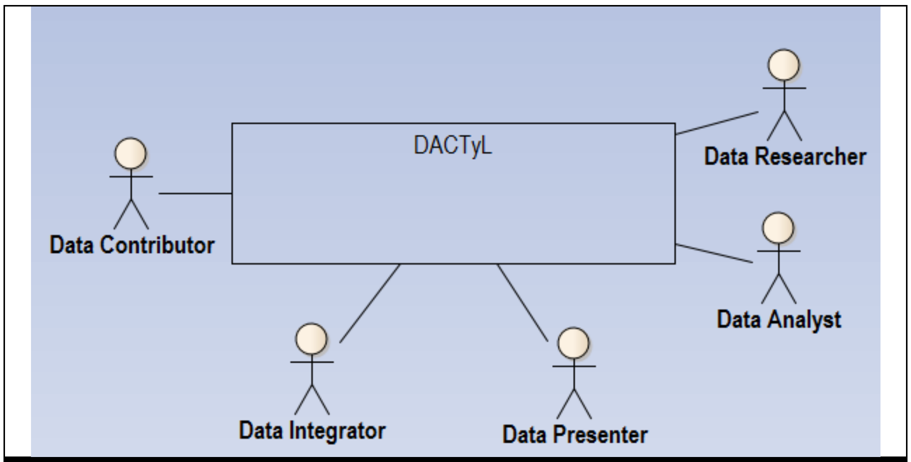


Figure 15 – Identified roles. The data contributor provides data. The data researcher and analyst receive information and limited data access. The data integrator and presenter configure DACTyL.

Every role can be treated as a requirements pool. This means that every user story generates a number of requirements that need to be met. Meeting these requirements realizes the user story. The main roles that we focused in this version of DACTyL are the data contributor and the data analyst.

6.3 *Project use case scenario*

This section describes the use case that was used for this version of DACTyL.

A hospital that provides the Motiva telehealth system to the patients wants to identify what is the return on investment from the telehealth service. As a first approach the hospital decides to monitor the number of hospitalizations for Motiva patients. The aim is to identify if the telehealth service has a measurable impact on the patient population.

The hospital executives know the expected number of hospitalizations for a patient population with a particular disease from literature. They would like to compare the expected number against the real one and deduct how many hospitalizations are avoided due to the healthcare service. Currently they are unable to do so because a patient record in the hospital system is not related to a patient record in the telehealth system.

The hospital agrees to contribute a set of patient data from the information system periodically. In return the hospital expects a system that is capable of processing the contributed data, merging them with Motiva data and producing interactive reports. The hospital is interested in two types of reports: the return on investment report and the patient report (see Sections 6.4.4 and 6.4.5).

In this use case a hospital IT system plays the role of data contributor. The role of the data analyst can be played by a finance manager who interacts with the reports produced by the system.

Figure 16 decomposes the project use case into smaller use cases.

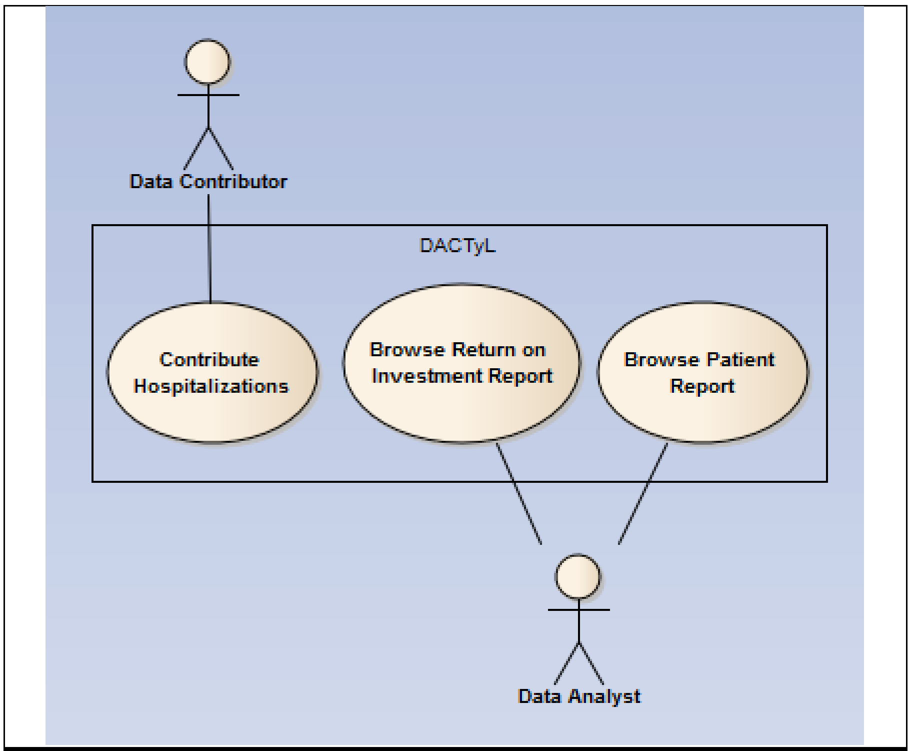

Data Contributor
DACTyL
Contribute Hospitalizations
Browse Return on Investment Report
Browse Patient Report
Data Analyst

Figure 16 – Use cases for this version of DACTyL

6.4 *Functional Requirements*

The MoSCoW [13] system is used to prioritize the requirements.

- M - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.
- S - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.
- C - COULD: Describes a requirement that is considered desirable but not necessary. This will be included if time and resources permit.
- W - WON'T: Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future.

To identify the requirements we arranged meetings with the immediate stakeholders from Philips. During these meeting the stakeholders described the existing system and introduced the use case in the previous section. Additionally they provided a data set from the Hull hospital in UK which would simulate the hospital data. This data set contains hospitalization events about Motiva patients.

6.4.1. Connecting Requirements

The connecting requirements describe the need to receive data from external systems. They are related to the contribute data use case and the data contributor.

Table 8 – Connecting Requirements

Id	Name	Priority	Description
C1	Input Data	M	The input for the DACTyL system shall be • Patient demographics • Admission date • Discharge date • Admission reason • Diagnosis • Mortality date • Patient unique identifier on the source system
C2	HL7 CCD Compatibility	S	The input for the DACTyL system shall be a valid HL7 v3 CCD document.
C3	Operational Source Location	M	The DACTyL system shall accept input data from operational sources that are physically situated in a different network.
C4	Updates	M	The DACTyL system shall accept input data from operational sources periodically.

6.4.2. Linking Requirements

The linking requirements describe the need to store the incoming data in such a way that information about the same patient is connected. They are related with the link data use case.

Table 9 – Linking Requirements

Id	Name	Priority	Description
L1	Unique DACTyL specific patient identifier	M	The DACTyL system shall assign a unique DACTyL specific identifier to every patient it contains.
L2	Data Linking	M	The DACTyL system shall link all available data about a specific patient.
L3	Data Linking Update	M	The DACTyL system shall link additional data about a specific patient when they become available.

- L1: The system must keep one unique code for every patient known to the system so that it can relate data from different sources to the same person. Additionally this DACTyL specific code hides any personal identification code.
- L2: The linked data are necessary to enable the combined reporting features.

6.4.3. Presenting Requirements

The presenting requirements describe the need to make the data available to the end users through reports. They are related with the use patient report and use return on investment report use case.

Table 10 – Presenting Requirements

Id	Name	Priority	Description

P1	Combined reporting	M	The DACTyL system shall produce a report that contains data from Motiva and from a clinical system for a specific patient.
P2	Return on in-vest report	M	This report compares real versus expected hospitalizations. See 6.4.4
P3	Patient report	M	This report combines hospitalizations and motiva tasks for See 6.4.5
P4	Access Re-strictions	C	The DACTyL system shall provide a custom level of detail according to the data analyst access rights.

- P4: The roles of the data analyst can be played by different entities such as doctors, nurses or hospital executives. The access to the data and the level of available detail should be tailored to protect the privacy of the data. This is an important requirement for later more mature versions of the system.

6.4.4. Return on investment report

The return on investment report is composed of dimensions and facts. The dimensions that the report must provide are documented in Table 10.

Table 11 – Return on investment report

Column Name	Description
Time	❖ All History ➢ Year ▪ Quarter • Month
Diagnosis	❖ All Diagnosis ➢ ICD-10 Chapter ▪ ICD-10 Block • ICD-10 Code
Age Group	❖ All Age Groups ➢ Age Group
Service	❖ All Services ➢ Service
Gender	❖ All Genders ➢ Gender
Admission Type	❖ All Admission Types ➢ Admission Type
Number of Hospitalizations	Sum
Total Length of Stay	Sum
Expected Hospitalizations	Constant

6.4.5. Patient report

The patient report is composed of dimensions and facts. The dimensions that the report must provide are documented in Table 11.

Table 12 – Patient report

Column Name	Description
Time	❖ All History ➢ Year ▪ Quarter

	• Month
Patient	❖ All Patients ➢ Patient
Number of Hospital-izations	Sum
Motiva Tasks	Sum

6.5 *Constraints*

Constraints describe the technology that must be re-used for the data warehousing part. As a reminder, DACTyL extends an existing prototype thus the same technology stack must be used. The constraints come from the immediate stakeholders.

Table 13 – Constraints

Id	**Name**	**Priority**	**Description**
CON1	Reuse of the Motiva data mart	M	The DACTyL system shall reuse the Motiva data mart which is already implemented
CON2	Reuse of Oracle data integrator	M	The DACTyL system shall reuse the Oracle data integrator tool to manage the ETL process
CON3	Reuse of Oracle Business Intelligence	M	The DACTyL system shall reuse Oracle Business Intelligence tool to create the end user reports
CON4	Patient Data Transfer	W	Conform with HIPAA regulation
CON5	Patient Data Storage	W	Conform with HIPAA regulation

6.6 *Nonfunctional requirements*

The nonfunctional requirements describe the qualities of the system. Philips is interested in a system that could receive data from an increasing amount of data contributors. Based on this vision we decided that maintainability is important for the system. A maintainable system in our case and according to our view does not require additional development effort for every new data contributor. When the system needs to adapt to a new incoming data set the architecture should include mechanisms that reduce the necessary changes. We believe that these adaptations are unavoidable due to heterogeneity of the healthcare environment.

Table 14 – Nonfunctional Requirements

Id	**Name**	**Priority**	**Description**
NF1	Maintainability: New data contributor, same data set.	M	The DACTyL system shall require 0 new lines of code and 0 new ETL scripts to accept a new data contributor who contributes data using an already known data set.
NF2	Maintainability: New data contributor new data set.	M	The DACTyL system shall require 0 new ETL scripts to accept a new data contributor who contributes data using a new data set.
NF3	Maintainability: New data contributor	M	The DACTyL system shall require 0 changes in the presentation reports to accept a new data contributor.
NF4	Robustness in	M	The DACTyL system shall present correct

	Presentation		data to the data analyst.
NF5	No prior knowledge	M	The DACTyL system shall not require knowledge of the data contributor database schema.
NF6	Portability	M	The DACTyL system shall not be bound to a specific database. This does not imply plug n play behavior for different databases.
NF7	Internationalization	C	The DACTyL system shall be usable in more than 1 country.
NF8	Load on operational source	M	The DACTyL system shall not add processing load to the operational source when a data analyst interacts with a report

■

7. System Architecture

7.1 Introduction

This chapter documents the proposed architecture to meet the requirements described in Chapter 6. The chapter begins by enumerating architecture alternatives that were investigated and concludes with the overall architecture of the analytical system.

7.2 Data Aggregation Architectures

During the first month of the project the design space for data aggregation architectures in healthcare was explored. The problem that these architectures try to solve is common. Different data sets contain valuable information and the end user of the system is interested in the combined data. For DACTyL the end user expects a combined view of the hospital and Motiva data. This section summarizes the findings from online resources and articles.

The first analytical systems collected data only from one operational source. The necessary data were transferred from the operational source to the analytical system. This is termed **first generation data integration** and it is obviously not relevant for DACTyL. Soon the end users required data from different operational sources. To meet these requirements the analytical systems used ETL and DW as described in Chapter 4. This is termed as **second generation data integration**. In the healthcare domain the privacy and regulation issues make the use of second generation system difficult because these systems require the transfer of sensitive data to a centralized DW. To overcome this barrier **the third generation systems** were used. These systems do not need a central DW to operate. The following paragraphs further analyze the second and third generation systems and compare them against the nonfunctional requirements of DACTyL. We begin with a description of the state of the art third generation systems, then we describe why this approach is not feasible for us, and then we describe the second generation which is our final choice.

7.2.1. Third Generation Integration

To overcome privacy, regulations and security concerns the third generation systems operate without the need of a centralized DW. These systems are further categorized as federated databases, mediated query systems and peer to peer data sharing systems [12]. Since there is no centralized DW, the systems use, in one way or another, mediator components that translate the user data requests to source-location-specific queries and distribute them. For this analysis we consider operational sources to be databases. Three distinct groups of third generation systems are distinguished.

A **federated database system** interacts with a number of databases with different schemas. The mediation happens at the application level. Every application that wants to extract data from the federation has knowledge of all local schemas. When the user issues a query the application uses the knowledge of the local schemas and translation rules to translate the query accordingly. When a new database joins the federation all applications must be updated with the new schema and new translation rules have to be implemented.

A **mediated query system** relies on mediator components to translate application queries to local queries. This mediator layer provides an interface between the applications and the local schemas and thus enables easier application development [13]. The mediators contain the knowledge of the local schemas.

Peer to peer data sharing systems are based on communication between data sharing agents. The agents require custom made programming for every source database. The agent based systems can be reconfigured in a decentralized way. A central command station can change the programming of all agents so the overall system can change behavior completely and start aggregating different data by altering only one configuration.

Some systems overcome the need of custom made programming by using a third party mediator component [15]. The mediator component is provided by a company who has agreements with certain EMR vendors. All agents share the same programming and rely on the mediator component to translate the queries. Unfortunately this abstraction layer does not exist for the general case. Another well-known example is the Dutch National Switching Point (Landelijk Schakelpunt LSP) system, previously known as AORTA. This implementation can be categorized as the brain of a third generation system.

Even though the above systems are considered state of the art approaches in data integration, a validation against the requirements of this project is necessary to decide on the systems' suitability. We identify three major drawbacks of third generation systems for this project.

- The first drawback of these systems is the need for knowledge of the source database schemas. In cases where this is not necessary these systems rely on components by third parties that encapsulate this knowledge. According to the research done for this project these third party components do not exist for hospital information systems due to the variety of systems and vendors.
- The second drawback is the need for additional programming to accept a new source database. There is a difference between configuring a system to accept a new source and programming the system for the new source. In the first case the necessary effort to add a new source is always the same. When new programming is required, for example, to update the translation rules, the effort depends on the new source.
- The third drawback is the performance aspects of such a system. Since the queries run on the local transactional systems, in our case Motiva and the hospital systems, when the end user interacts with the application he adds processing load to the source database. This may cause performance issues. Additionally the results of the queries have to travel back to the end user.

The non-functional requirements **NF1**, **NF5** and **NF8** (see Chapter 6) are the reasons that make a third generation approach inappropriate for this project.

7.2.2. Second Generation Integration

Opposed to third generation systems where data remains in the source, second generation systems move data from the data source to a central location, a data warehouse. The organization of data (data models) in the data warehouse is optimized for analytical querying. In addition to the data warehouse, applications provide access to the data. The applications query the data warehouse and not the data sources. An example of a second generation system is the SPINE system used by the U.K. National Health Service (NHS), which keeps a central record of patient demographics and care summary.

For a second generation system the pull or push data collection strategy is an important decision. When the operational source initiates the communication, this is categorized as push and when the central system initiates the communication, this is categorized as pull. The push strategy gives total control over the communication to the hospital. This is also the recommended way of communication according to one of the IHE profiles named XDS [8]. A pull strategy may appear "too aggressive" to hospital IT managers and includes the danger of adding workload to the operational source at an inappropriate time.

7.2.3. Conclusions

A second generation push system fulfills **NF1**, **NF5** and **NF8** (see Chapter 6):

- NF1: The system expects a particular input from a data source. When a new data source is added who contributes an already known data set, the system requires no additional programming.
- NF5: The system contains no knowledge about the potential data sources and their database schemas.
- NF8: The interaction with the end user reports affects only the DW and not the data sources. The load related to data submission is managed by the data contributors. They can decide on the appropriate time to submit the data.

This approach forces data to travel from their original location to the analytical system. In the healthcare domain this approach has to overcome privacy related issues. These issues are out of scope for this project but they should be taken seriously in a real life system.

7.3 Refined Decisions

This section groups the design decisions in four different areas. The decisions are on the system level. Nevertheless after reading this section the reader should understand the boundaries of the system and the main building blocks. We connect the analysis with the requirements of Chapter 6.

Every analytical system makes certain design decisions in the following areas [13]:

- **Data Extraction**: Data extraction is the process of extracting the necessary data from the operational source.
- **Transport**: Transport is related with networking protocols, technologies and security considerations.
- **Transformation and Normalizations**: Transformation and Normalization refers to the process of harmonizing different data formats and coding systems.
- **Analysis Set Creation-Delivery**: Analysis Set Creation-Delivery is related to the processing of the integrated data and how they are presented to final users.

7.3.1. Data Extraction

As a push system DACTyL is by default passive. This means that the actual data extraction is external to DACTyL. The extraction is managed by the interface engine of the hospital as described in Chapter 5. The hospital has to configure the interface engine and initiate the communication with DACTyL. In a realistic scenario this configuration will be executed by a Philips service engineer in cooperation with the hospital IT department.

By keeping the data extraction external to DACTyL we do not require knowledge about the database schemas of the actual hospital information system. As mentioned earlier this knowledge belongs to Philips' competitors. Additionally we leave the initiative for connection to the hospital which is easier than requesting for permission to connect an external system directly to the hospital's systems.

7.3.2. Transport

DACTyL exposes a web service that is ready to receive incoming data from the hospital's interface engine. We choose web services since this is a widely used and well established technology. The protocol used to transfer the data is http and the data are xml documents packaged in soap messages. We used http[3] for prototyping since the security issues are not under the scope of this project.

7.3.3. Input interpretation

Apart from the network protocol and the technology used in the transport the input, the input needs to be interpreted. This means that DACTyL must be able to parse the xml document in this case and understand the semantics of the contents. This "intelligence" is the specification of the **input model**. When DACTyL receives the xml document it will try to interpret it according to the specification of the input model. The xml documents that DACTyL receives are valid serialized instances of the input model[4].

In an ideal case all data contributors would provide their data using the same xml document which can be interpreted by the same input model. In this case DACTyL would require only one input model. We consider this case unrealistic therefore we design DACTyL in a way that it can use different input models to interpret the incoming xml documents.

Other candidates for input models are

- An existing version of HL7 CDA document such as the CCD.
- A new version of HL7 CDA document

A decision for a version of HL7 CDA model requires the hospital to export his data as an xml document which is a valid instance of the HL7 CDA model. Even though the standard for HL7 CDA export can be supported in existing solutions according to system vendors, in practice it is not widely used[5]. This is due to the fact that hospitals do not exchange data thus they do not need exporting features that provide semantic interoperability.

Additionally it should not be neglected that countries that use the HL7 CDA model as a standard for interoperability, define their own country-specific versions, which is relevant due to the internationalization requirement **NF7**.

The decision for support of a particular input model should be taken in close cooperation with the hospitals that will contribute data. For this version of DACTyL we defined a custom input model that fulfills the **C1** requirement.

7.3.4. Transformation-Normalization

Equipped with the specification of the input models, the system is able to interpret data from different input models. This enables Philips to accept data in different input models and minimizes the need for additional development. For example Philips can agree to accept data using HL7 CDA version A with data contributors in the Netherlands and version B with data contributors in the U.K.

Regardless of the input model the incoming data need to be normalized. The normalization can differ per use case. One form of normalization is translating different units, for example kilos to pounds, or changing date formats. More complicated use cases may include mappings between different clinical vocabularies, for example from SNOMED codes to ICD-10 codes. Such mappings require interaction with additional systems usually called clinical vocabulary servers and are left out of scope in this project. Nevertheless the transformation-normalization is designed in a way that allows further extension.

[3] In case of https there is no noticeable difference in the overall system.
[4] The reader can see an input model and a serialized input model instance in Section 9.2 Figures 22 and 23.
[5] We visited the Zorg & ICT 2013 and interviewed the major hospital IT system vendors. They commented that the standard is not widely used.

Transformation in the context of this project is the process of transforming the instance of the input model to an instance of another model. We introduce the concept of the **harmonization model**. This model is a Philips-specific model and acts as interface between the data received and the saved data. The instance of the input model is translated in a step by step process to an instance of the harmonization model. This process includes normalization steps and additional business logic steps. For example, apart from the above mentioned normalization steps, we could mark the incoming data with a timestamp that records when DACTyL received the data.

The instance of the harmonization model is finally stored in a database. The harmonization model hides the details of the different input models from the rest of the system. Figure 17 shows this transformation prior to data saving. Continuing the example with the data contributors from different countries, we see that the system requires the addition of a new transformation to accept instances of a new input model and it can then reuse the rest of the components.

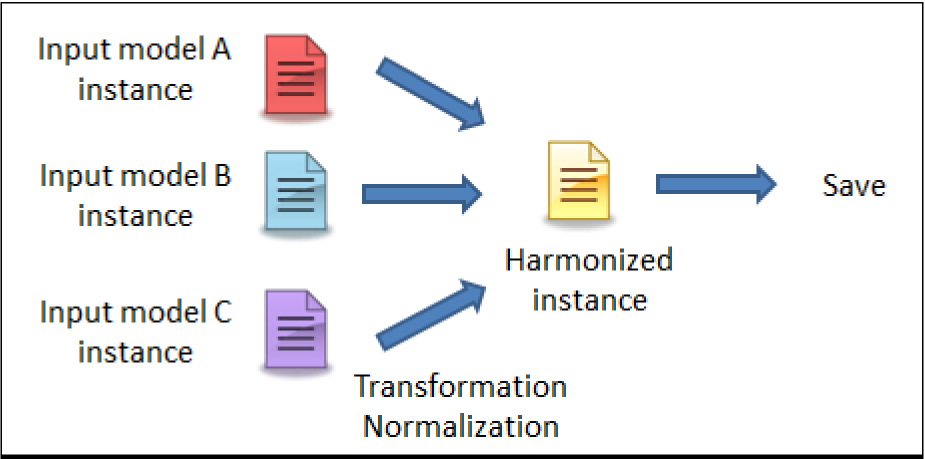


Figure 17 – Transforming different input instances to an instance of the harmonization model prior to saving

Analysis Set Creation-Delivery
The constraints of Chapter 6, **CON1-3,** provide the elements for the data warehousing part of the system. We decided to design a new data mart for the incoming data. This choice is justified because the incoming data are related to a new business process, in this case hospitalization. Since the Motiva data mart is already designed as a dimensional model we follow the data warehouse bus architecture according to [10] in order to combine the two data marts. The tooling used for the ETL process and the creation of the reports is defined by the constraints.

These design decisions are also related to the linking and presenting requirements of Chapter 6.

7.3.5. Summary of Refined Decisions

The following list summarizes the design decisions:

- Data Extraction: External to DACTyL. Implemented by the interface engine in the hospital.
- Transport: A web service receives xml documents packaged in soap messages over http. A custom made input format is used and special care is given for support of additional input formats.
- Transformation and Normalizations: A harmonization model is used to hide the details on the incoming models from the rest of the system. DACTyL stores instances of the harmonization model.
- Analysis Set Creation-Delivery: A new data mart for the hospitalization data is designed. The data warehouse bus architecture is used to combine the

hospitalization and Motiva data marts. The ETL tooling specified by the constraints is used to load the hospitalization data mart from the stored data. The BI tooling specified by the constraints is used to create the end user reports.

7.4 Overview of the system

Figure 18 shows the overview of the system and the environment where it operates. Starting from the bottom of the figure we see the hospital information systems and the hospital interface engine. The interface engine communicates with DACTyL through the web service. It transmits data packaged in an xml document.

The web service delivers the incoming data to the processing layer. There the input model is going to be transformed to the harmonized model. The last action of the processing layer is to store the data in the landing area.

A number of ETL processes periodically load the data marts contained in the data warehouse with new data coming from the landing area.

The system provides dashboards and reports that present the available data. These dashboards are accessed by the end users of the system. Additionally the data are available to other applications, which can query the system directly. The figure shows a number of possible end users, who can be seen as data analysts or data researchers.

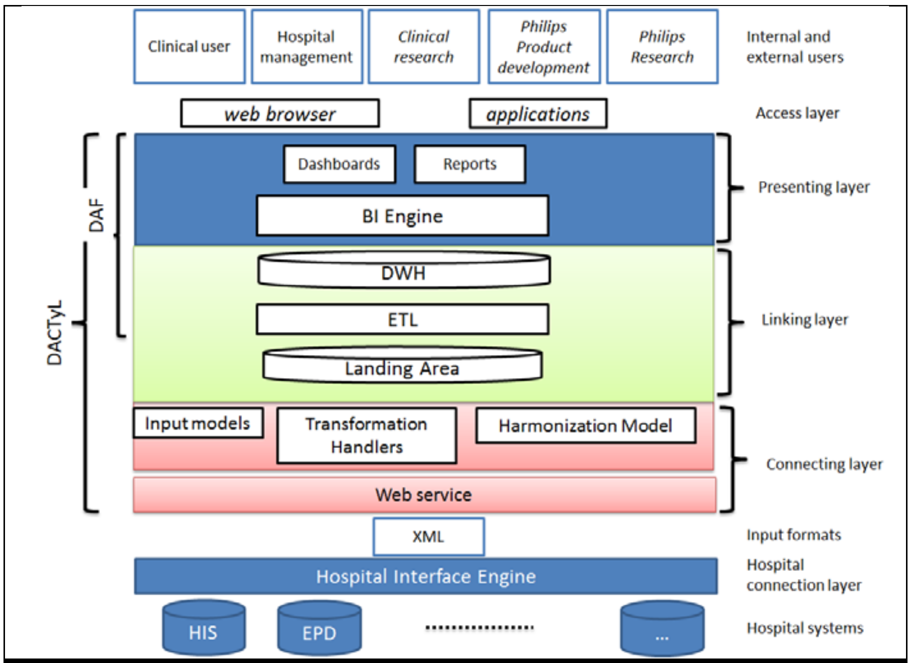


Figure 18 – Overview of DACTyL

■

8.System Design

8.1 *Introduction*

This chapter describes the layers and the components of DACTyL. Other design elements such as the design of the data marts and the design of the input model are described in the implementation chapter (Chapter 8) to keep it more cohesive for the reader.

8.2 *Component Diagram*

This section presents the component diagram in UML2 of DACTyL in Figure 19 and explains the purpose of each layer. The description is intentionally high level.

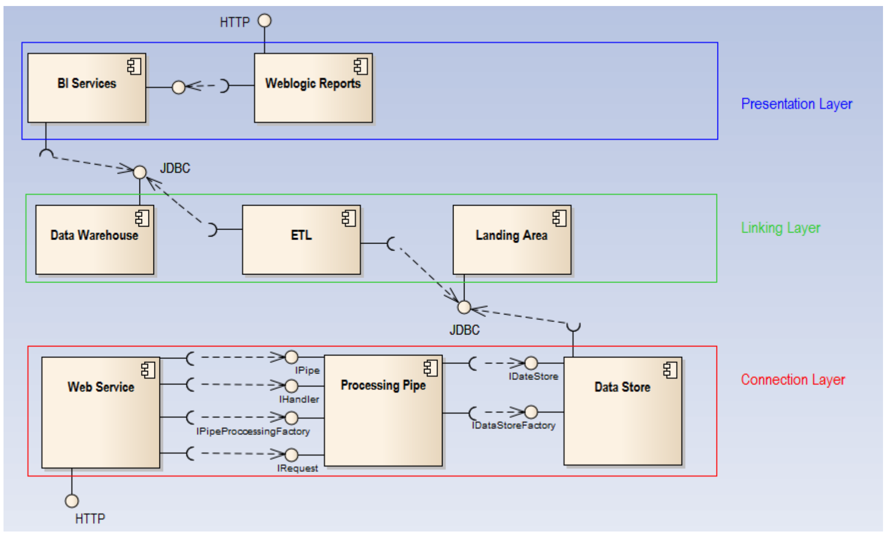


Figure 19 – Component Diagram

8.2.1. Connection Layer

The connection layer is responsible for receiving the incoming data, transforming them to the harmonization model and storing them to the landing area. This layer uses java classes and is physically hosted in a web container such as Apache Tomcat. The connection layer offers a URL, which can be used by the data contributor to submit data. This layer requires a connection with a database system. This database system is the landing area.

Web Service component
This component exposes a public URL to the operational sources. Every web service serves a particular input model. Additional input models can be supported by exposing new web services. The connection between the web service and the outside world is handled by the web container.

Processing Pipe component
The Processing Pipe component offers transformation handlers, which transform the instance of the input model to an instance of a harmonized model. Additionally it offers handlers that store the instance of the harmonized model.

Data Store component
The Data Store component offers interfaces are used by the handlers to implement the storage of the harmonized model instances.

8.2.2. Linking Layer

The linking layer offers two storage areas to the adjacent layers. The storage areas are the landing area and the data warehouse. The landing area is used by the connecting layer and the data warehouse by the presenting layer. Internally it handles the loading of the data warehouse with new data.

Landing Area

The landing area is a relational database that stores the incoming data in the schema specified by the harmonization model. It plays the role of the data archive. The existence of this database provides flexibility to create different data marts out of the same collected data.

As mentioned earlier the data marts are de-normalized databases specifically designed to answer a particular business question. Since the stakeholders will come up with new questions on a regular basis, it is highly probable that the existing data marts in the data warehouse will not be able to fulfill them. Thus the system must be able to create new data marts. The reader could argue that the new data marts could come from the existing ones and the landing area is not necessary as the data can be stored directly to a data mart. This is technically possible, under certain restrictions, but not advisable. The data warehousing best practices [9] suggest that the data marts should be created out of the operational data source.

Designing the landing area as a database is a design decision. Other feasible alternatives are text files and directories. First we believe that a database is easier to manage and scale. Second the ETL tool can take advantage of out-of-the-box features such as table journalization[6]. In combination with the frameworks used in the communicating area, the need for hand written persistence code is minimized effectively removing one of the database approach considerations.

The system keeps a separate database schema for every contributor. The communication components are responsible for routing the incoming data to the appropriate database. By keeping one isolated database per data contributor we avoid the single point of failure problem and make traceability or errors easier. Additionally the ETL tooling that we used (see Section 9.1.3) makes it very easy to execute the ETL process on identical physical schemas.

The landing area schema affects the complexity of the ETL design. A complicated harmonization model can generate a complicated landing area schema. This will result in even more complicated ETL procedures. The main design principle followed here is simplicity. The landing area schema has to be understood and used by the data integrator to create the ETL scripts.

ETL component

The ETL component orchestrates the data transfer between the landing area and the data warehouse. This component is realized by an ETL tool. The tool is able to translate the ETL design to SQL scripts. The SQL scripts are executed by the database engines involved in the ETL design. To connect to the database engines the tool uses the appropriate driver.

Data Warehouse component

The Data Warehouse stores the data marts. The data marts are dimensionally designed as explained in Section 4.5. The data marts are the data sources for the data analyst's reports. The ETL component periodically loads new data in the data marts.

[6] This feature can mark rows in the landing area as "already integrated". These rows still exist in the physical table but they will not be considered in the next ETL execution.

8.2.3. Presentation Layer

The presentation layer offers access to the data stored in the data warehouse. This access mechanism is used by the data analyst and the data researcher. The access can be realized through dashboards and reports or through direct queries against the OLAP cube. The presentation layer requires a connection with the data warehouse.

BI Services and Weblogic Reports components

These components are realized by tools and are responsible for creating the user reports and translating the user actions into SQL queries towards the data warehouse. These tools require configuration that specifies the location of the data warehouse, the data model and which tables are available for usage in the end user reports.

■

9.Implementation

9.1 Introduction

This chapter documents the implementation of the proof of concept system. Before the actual implementation section a small introduction is given to the frameworks and tools used for this version of the system.

9.1.1. Introduction to Eclipse Modeling Framework

Eclipse is a well-known platform used by numerous developers. One of the frameworks that it provides is the Eclipse Modeling Framework (EMF) [21]. EMF is a model driven development framework where the developer can design models and the framework can produce java source code that implements the model.

The models designed in EMF are based on the Ecore meta-model, which can be roughly described as a simplified version of the UML2 meta-model. The developer can create a model using classes, associations and attributes to describe a domain of interest. The code generator can read the model and generate the java code.

The framework also provides serialization and de-serialization utilities. This means that an instance of an EMF model, which is a number of java classes in the RAM memory, can be serialized in an xml file. This xml file can be then parsed to recreate the instance of the model in the memory of the computer.

In this project we design the input and harmonization models at configuration time using this framework. At runtime we use this framework to parse the input model and create a new instance of the harmonization model.

9.1.2. Introduction to Teneo

When we want to save a java object to a database there are a number of ways to achieve this. The traditional way is to write SQL and save the state of the object to the database. A more modern approach makes use of object relational mapping techniques such as JPA and Hibernate. With these frameworks there is no need to write SQL code in order to save an object to a database. Instead the developer must annotate the java source code with labels in the case of JPA or write a mapping file in the case of Hibernate.

Teneo [22] uses Hibernate [23] and is an object persistence framework that can automatically generate the Hibernate mapping file for an EMF model. This means that there is no need to create the database schema, to write the SQL code that saves the object or to write the Hibernate mapping file. Object relation mapping has a number of limitations as there is no one to one mapping from java to the relational database schema. For example, relational databases cannot model inheritance. How Teneo deals with such cases is specified in a configuration file.

We used Teneo to generate the landing area database schema out of the harmonization model at configuration time. At runtime we used Teneo to save the instance of the harmonization model to the landing area.

9.1.3. Introduction to Oracle Data Integrator

The Oracle Data Integrator (ODI) [24] is an ETL integration interface between data sources and targets. The process designed in ODI implements the ETL component. The following section will describe briefly some key concepts of the tool that are used in the project.

A physical schema in ODI describes the physical characteristics of both the data sources as the targets. It describes the type of database server (Oracle, DB2, SQL Server, Informix and Sybase), where the database server is located and the name of the database server. For a file as a data source or target, things such as the filename, the file type (XML, flat file) are described.

One physical schema or a group of structurally identical physical schemas that are in different physical locations can be identified as a single logical schema. The logical schema is resolved into one of the physical schemas at runtime for a given context (see next paragraph). In ODI all interfaces[7] that are developed are designed on top of the logical schema. The logical schema acts like a pointer to a physical schema.

Contexts are runtime configurations that map logical schemas with physical schemas. For example, in our use case we can execute the ETL process for context A which uses the landing area-physical schema A, and then run the same ETL process for context B which uses the landing area-physical schema B. In general, structurally equal physical data sources and data targets but with different content can be addressed in different contexts.

9.1.4. Introduction to Oracle Business Intelligence Enterprise Edition

Presentation components are realized by Oracle Business Intelligence Enterprise Edition [25] tool. This toolset is Oracle's state of the art response to BI requirements. According to the role analysis, the data presenter interacts with this toolset to design the reports for the data analyst.

The data presenter creates three different models using the Oracle BI Administrator tool, namely physical, logical and presentation models[8]. The physical model contains the physical database that is used to provide data to the reports. For this deliverable the physical model contains the Motiva and hospitalization data mart. The business model is the OLAP cube. This model binds the physical tables of the DW with the cube. In this model the levels of the dimensions and the measures of the fact tables are specified. The presentation model defines which fact tables and dimensions are available for the report and dashboard creation. Here the data presenter can change the names of tables and columns that will appear in the end user reports. Using these three models the OLAP cube has sufficient knowledge to translate the user interaction with the reports to SQL queries specific for the two data marts used in this project. Figure 20 shows the user interface for the OLAP cube configuration. On the right side we see the physical model, next to that is the business model and at the left side is the presentation model.

[7] Interface in ODI terminology stands for a data mapping between a source and a target database table.

[8] These models are completely unrelated with the input and harmonization models mentioned earlier. We simply follow the terminology of the tool.

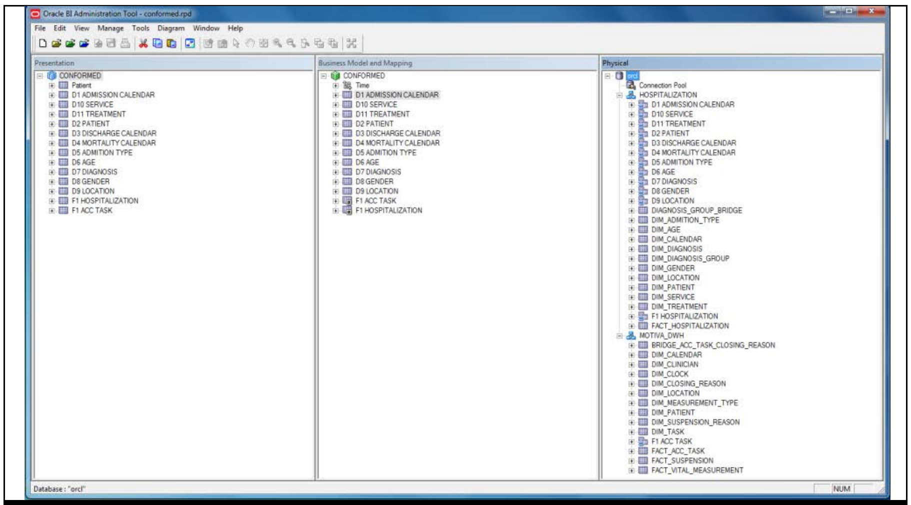
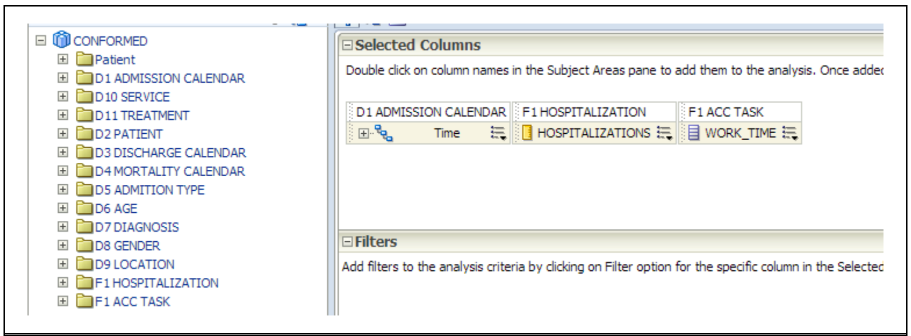


Figure 20 – Presentation (left), business (middle) and physical (right) models

The final step in this area of focus is the actual design of the end user report. The data presenter has every object in the presentation model at his disposal to compose the dynamic reports. Figure 21 shows this UI. The reports can have the form of spread-sheets with clickable elements that allow the drilling up and drilling down features. Additionally other user interface elements can be created such as graphs or dropdown menus. The elements of the presentation model in the left side of Figure 20 are the elements that can be used to create the end user reports and can be seen in Figure 21.



Double click on column names in the Subject Areas pane to add them to the analysis. Once addec

Add filters to the analysis criteria by clicking on Filter option for the specific column in the Selectec

Figure 21 – Creating a new report for the data analyst

In a real product deployment, users, roles and access rights are configured in the OBIEE tool to manage the access to the available data and the level of details visible. This means that the patient details are only available to their care providers. Other users for example hospital management or researchers have access to a subset of de-identified data. This configuration was out of the scope of this project.

9.1.5. Tooling layout

This section shows where each tool or technology is used in this project. Figure 22 shows the layers of the system "annotated" with the tools used to realize them.

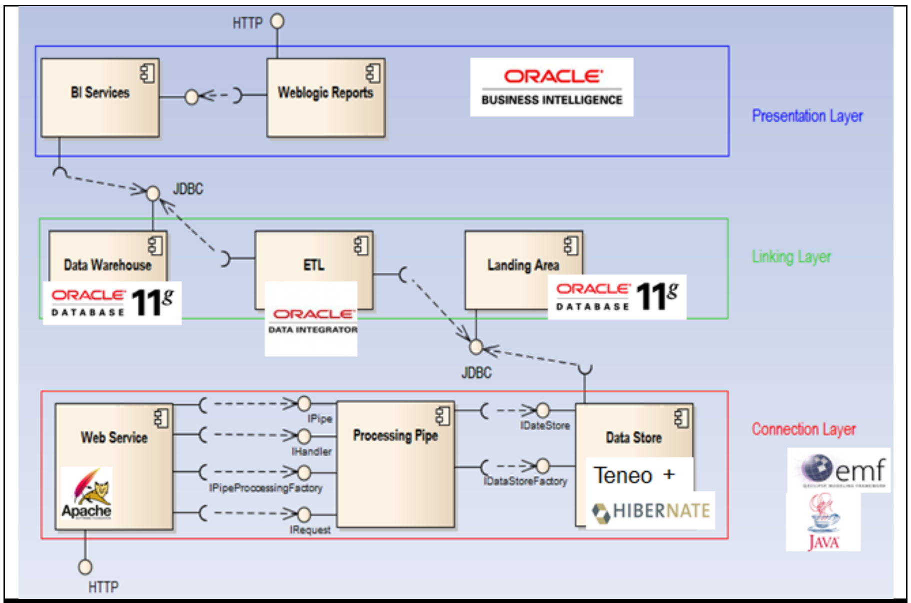


Figure 22 – Tooling layout in DACTyL

9.2 *Implementation of the input model*

Since we currently have no must requirement to support a particular input model the input model and the harmonization model are the same for this proof of concept system. This means that the transformation process of the input model does not result in a different model but in a model with additional values.

This section describes how the input model is implemented. To implement the input model we analyzed the data that the team had at their disposal for this version of the system. Since we built a proof of concept system we chose to create a purpose built input model capable of accommodating our available data. Table 15 shows the available data set and briefly explains every column.

Table 15 – Hospitalization events data set

Column name	Description	Used in the Input Model	Data example
Motiva Patient ID	The code of the patient in the Motiva system.	Yes	147
Link ID	Hospital internal code. Not relevant.	No	152
Before/After	Marks if the hospitalization occurred before or after the patient enrollment to Motiva.	Yes	Pre
Hospitalization ID	Unique identifier of the hospitalization event per hospital.	Yes	456
Date of Death	Self-explanatory	Yes	25-03-2011
Admission Date	Self-explanatory	Yes	20-03-2011
Discharge Date	Self-explanatory	Yes	22-03-2011
Admission Month	Self-explanatory	No	03
Admission Year	Self-explanatory	No	2011
Admission Method	Hospital specific	Yes	AE

	code that describes an admission method.		
Treatment Specification	Hospital specific code that describes the treatment.	Yes	GEMN
Consultant Code	Identifier of the doctor. Not relevant.	No	WALC
Diagnosis 1 to 8	ICD code for the diagnosis. A hospitalization event may be related with a number of diagnoses.	Yes	R07.4

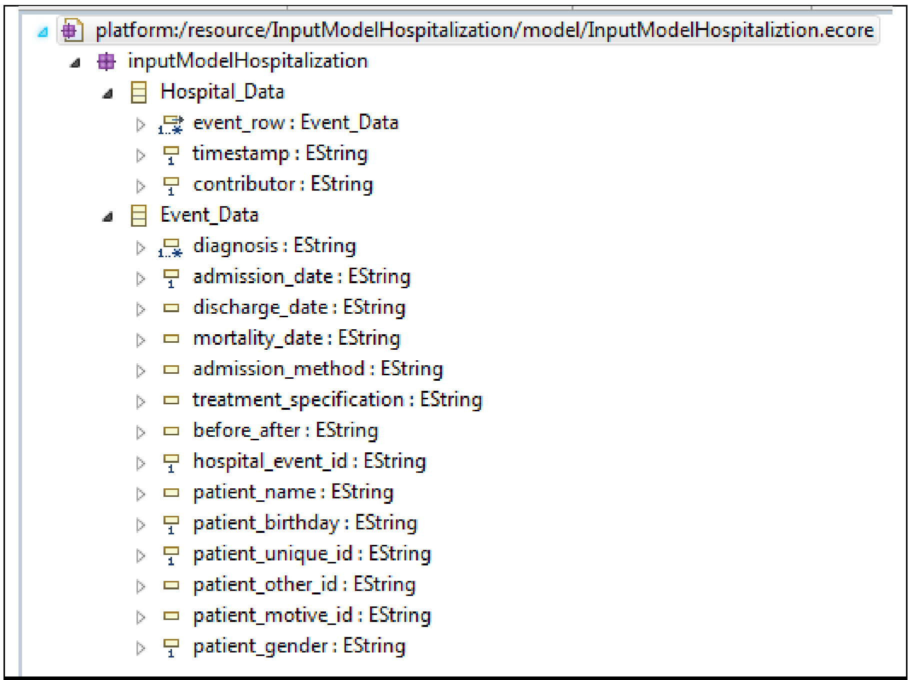
From the available data set we chose not to use the Motiva ID code even though it is an appropriate key to match the patients across the data marts. The Motiva ID is a Philips ID not known to the hospitals in the general case. This means that we expect that the hospitals will not be able to provide this number.

To create the input model we decided to include additional columns shown in Table 16. The patient name is added for presentation purposes. The unique patient ID in combination with the patient birth date is part of the key used to match the patients in the different data marts. The patient birth date is used to classify the hospitalization events in different age groups. The patient gender is used for the same reason. We expect that the data sent by the hospital contains at minimum the data to populate an instance of the input model.

Table 16 – Hospitalization events data set

Column name	Description	Used in the Input Model	Data example
Patient name	Self-explanatory	Yes	Orestis
Unique patient id	A number that uniquely identifies the person. For example, the BSN number.	Yes	1234365
Patient birth date	Self-explanatory	Yes	11-02-1986
Patient other id	Identifier of the patient. For future use cases.	Yes	ERT3455
Patient gender	Self-explanatory	Yes	M

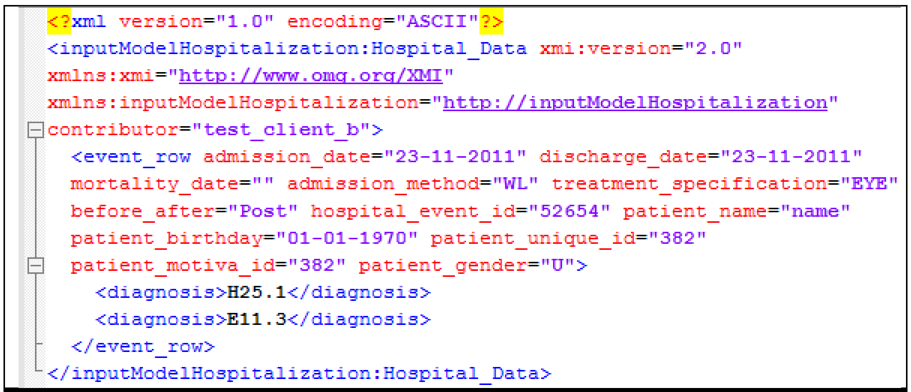
Using the EMF framework and the model editor we created the model shown in Figure 23. In this figure we see the Event_Data class. This class represents the hospitalization event and contains all the previously mentioned columns as attributes. The Hospital_Data class contains one or more hospitalization events and two additional attributes. The contributor attribute specifies the sender of the message and the timestamp attribute is added by the server when it receives the message in order to mark the time of message arrival in DACTyL. It is important here to point out that a number of attributes have a multiplicity 1 or 1..*. This means that these attributes are considered mandatory for a “savable” hospitalization event. The columns that will store these attributes in the landing area cannot be null.



```
platform:/resource/InputModelHospitalization/model/InputModelHospitaliztion.ecore
  inputModelHospitalization
    Hospital_Data
      event_row : Event_Data
      timestamp : EString
      contributor : EString
    Event_Data
      diagnosis : EString
      admission_date : EString
      discharge_date : EString
      mortality_date : EString
      admission_method : EString
      treatment_specification : EString
      before_after : EString
      hospital_event_id : EString
      patient_name : EString
      patient_birthday : EString
      patient_unique_id : EString
      patient_other_id : EString
      patient_motive_id : EString
      patient_gender : EString
```

Figure 23 – Design of the input model

A serialized instance of the model is an xml document. This document is the payload of the soap message that the web service receives. An example is given in Figure 24.



```xml
<?xml version="1.0" encoding="ASCII"?>
<inputModelHospitalization:Hospital_Data xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:inputModelHospitalization="http://inputModelHospitalization"
contributor="test_client_b">
  <event_row admission_date="23-11-2011" discharge_date="23-11-2011"
  mortality_date="" admission_method="WL" treatment_specification="EYE"
  before_after="Post" hospital_event_id="52654" patient_name="name"
  patient_birthday="01-01-1970" patient_unique_id="382"
  patient_motiva_id="382" patient_gender="U">
    <diagnosis>H25.1</diagnosis>
    <diagnosis>E11.3</diagnosis>
  </event_row>
</inputModelHospitalization:Hospital_Data>
```

Figure 24 – Serialized instance of the input model

The only difference between the input model instance and the harmonization instance is the timestamp attribute. The instance of the input model does not contain the timestamp attribute. This is added by DACTyL in a transformation step.

The next list summarizes the minimum necessary **assumptions** about the input data

1. Every hospitalization event is marked with a “hospital event id” which is unique for the data contributor. This ID enables the system to understand if the data refer to a new hospitalization or it is an update that contains additional data about a past hospitalization.
2. An update for a hospitalization event will be marked with the same “hospital event id” as the original one.
3. An update contains at least the previously submitted data and additional data about a hospitalization event.
4. Every hospitalization event contains a unique identifier for the patient.
5. Every hospitalization event contains the birth date of the patient.
6. The combination of the patient unique identifier and the birth date identifies a patient across data contributors.

9.3 *Implementation of the connection layer*

This section describes the implementation of the connection layer. This layer is responsible for parsing the incoming data and storing them in a changed form to the landing area. The class diagrams and the sequence diagrams in UML2 for the software components are presented here.

The classes that implement the connecting components receive the input from the data contributors and deliver an output to the landing area. Figure 25 shows the components and the classes that implement them. The communication between the components is based on java interfaces. The classes that implement the interfaces make extensive use of external frameworks. The interfaces, in contrast, carry no dependencies to external frameworks. In this way the implementation classes can be easily changed. The packaging of the classes separates the interfaces from the implementation.

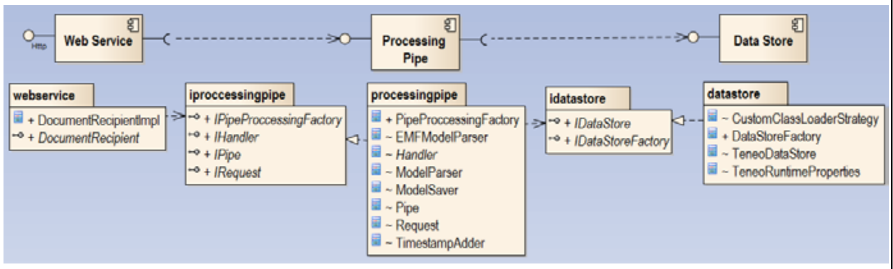


Figure 25 – Package diagram from components to classes

A web service is a normal class that provides a public function. When the web service is deployed in a web container such as Apache Tomcat a separate configuration file informs the web container that a new public function should be published. The web container takes care of the publishing actions for the new function. The container creates a new unique URL for this function which can be reached by an http request. When a data contributor sends a request to this URL, the web server finds the appropriate configuration file and loads the web service class in a new thread. Finally he calls the public function with the incoming data as an argument. Since every web service call runs in a different thread, the web server is able to service multiple concurrent calls to the same web service.

The web service in our case, knows what the expected input and harmonization model is. It performs a rather simple job. It assembles a processing pipe and puts the incoming data in the pipe. To create the necessary objects the web service uses a factory injected with the input and harmonization models. The web service uses the public interfaces of the processing pipe. This decouples the web service from the processing pipe implementation.

Figure 25 shows the concrete factory and the interfaces[9] of the classes that will implement the behavior. As expected, the pipe processing factory will create the objects. The "create" dependency lines from the factory towards the concrete classes and the concrete classes themselves are omitted in Figure 21 due to lack of space. The output model in Figure 26 is what we termed as harmonization model.

The injection of the factory with the input and the harmonization model is necessary because some transformation handlers are generic. By generic we mean that exactly the same handler implementation class can be used for different models.

[9] The differences in the font size of the interface operations are due to a bug in Enterprise Architect and carry no special meaning. All operations are equally important.

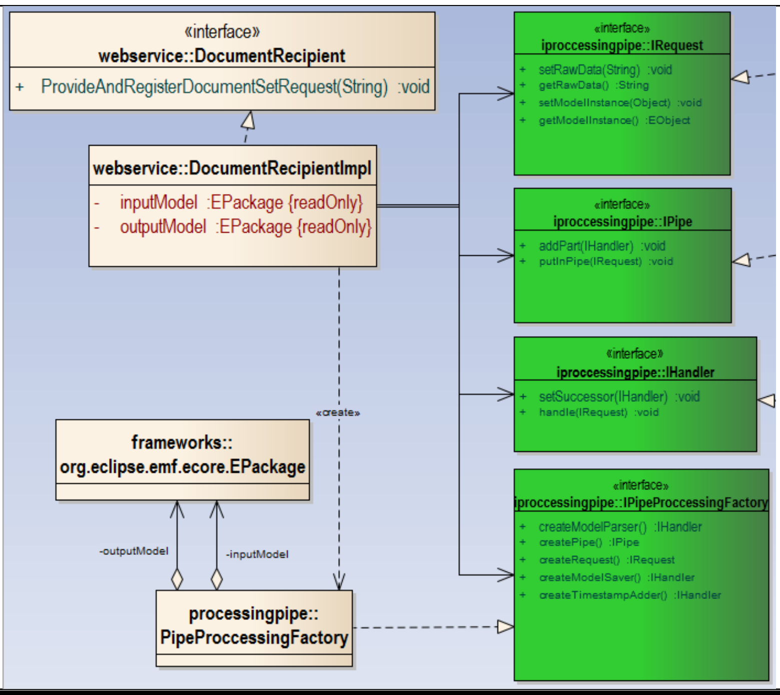

«interface»
webservice::DocumentRecipient
+ ProvideAndRegisterDocumentSetRequest(String) :void
webservice::DocumentRecipientImpl
- inputModel :EPackage {readOnly}
- outputModel :EPackage {readOnly}
«create»
frameworks::
org.eclipse.emf.ecore.EPackage
-outputModel
-inputModel
processingpipe::
PipeProccessingFactory
«interface»
iproccessingpipe::IRequest
+ setRawData(String) :void
+ getRawData() :String
+ setModelInstance(Object) :void
+ getModelInstance() :EObject
«interface»
iproccessingpipe::IPipe
+ addPart(IHandler) :void
+ putInPipe(IRequest) :void
«interface»
iproccessingpipe::IHandler
+ setSuccessor(IHandler) :void
+ handle(IRequest) :void
«interface»
iproccessingpipe::IPipeProccessingFactory
+ createModelParser() :IHandler
+ createPipe() :IPipe
+ createRequest() :IRequest
+ createModelSaver() :IHandler
+ createTimestampAdder() :IHandler

Figure 26 – Class diagram showing the factory and the participating interfaces

The transformation of the input model to the harmonization model is achieved in a number of steps. Figure 27 shows the Pipe, the Handler and the Request class. The Request class is a wrapper that contains data. The structure of the Handlers is inspired by the Chain of Responsibility pattern with one twist: every handler in the chain is responsible for the request and acts upon the request.

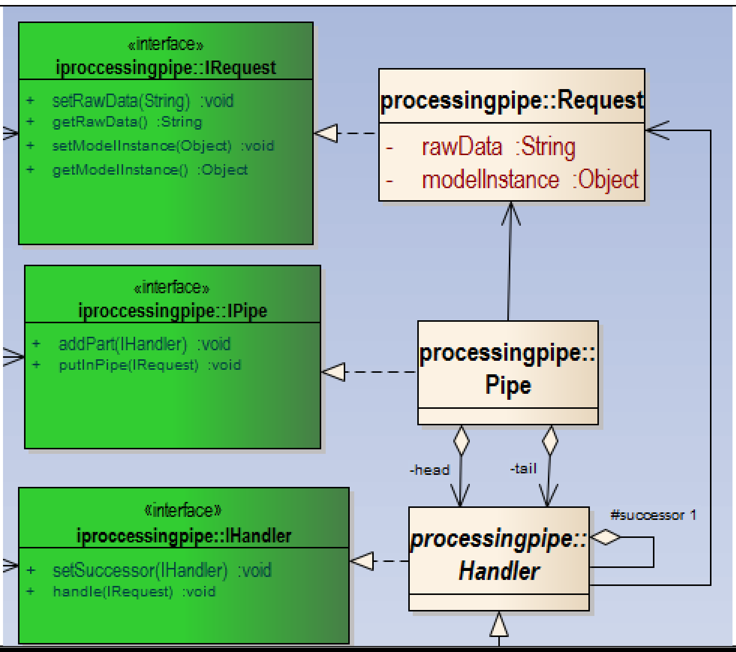


Figure 27 – The parser and the saver handler

Two types of handlers are always present in a Pipe: a parser and a saver handler. The parser handler parses the incoming data. This means that the parser will read an xml string and check if the string is valid according to the rules of the input model. At this point we want to stress the fact that this parser is generic and injected with the input model. The parser can parse any serialized EMF input model.

A saver stores the instance of the harmonized model to the landing area. When the request object reaches the saver the input model is considered to be transformed to the harmonization model. The saver makes use of the classes that implement the data store component. Figure 28 shows the handlers that are always present in the beginning and at the end of a processing pipe. We can also see that the saver creates the objects of the data store component through a factory and uses the public interfaces to invoke the behavior. This decouples the saver from the implementation of the data store and the Teneo framework which is used in this project.

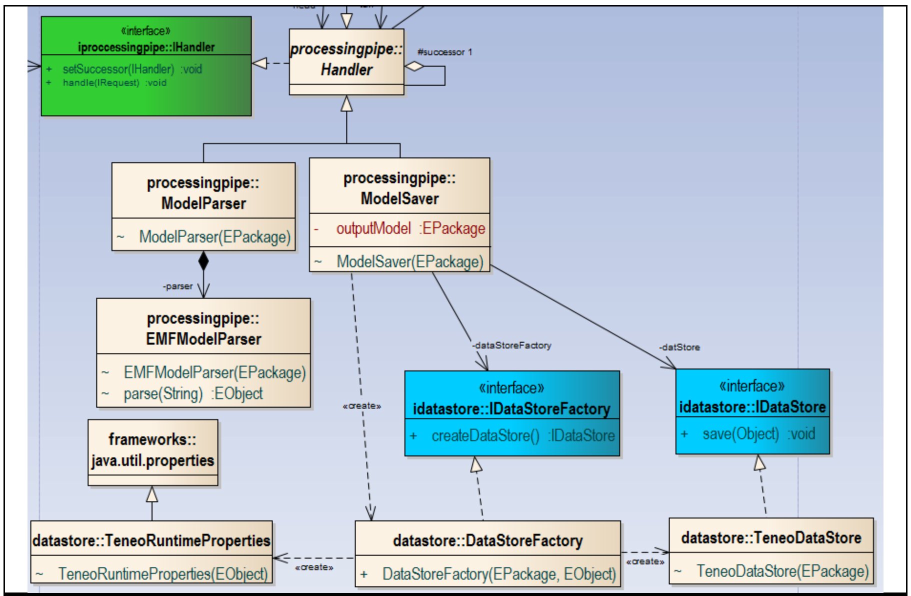
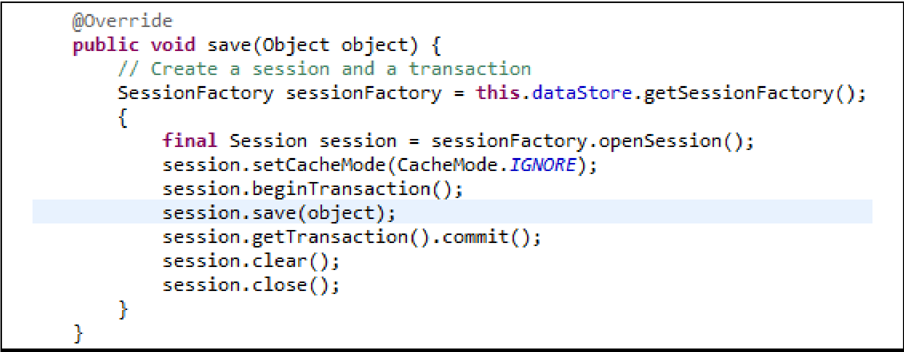

«interface»
iproccessingpipe::IHandler
+ setSuccessor(IHandler) :void
+ handle(IRequest) :void
processingpipe::Handler
#successor 1
processingpipe::ModelParser
~ ModelParser(EPackage)
processingpipe::ModelSaver
- outputModel :EPackage
~ ModelSaver(EPackage)
-parser
processingpipe::EMFModelParser
~ EMFModelParser(EPackage)
~ parse(String) :EObject
-dataStoreFactory
-datStore
«create»
«interface»
idatastore::IDataStoreFactory
+ createDataStore() :IDataStore
«interface»
idatastore::IDataStore
+ save(Object) :void
frameworks::java.util.properties
datastore::TeneoRuntimeProperties
~ TeneoRuntimeProperties(EObject)
«create»
datastore::DataStoreFactory
+ DataStoreFactory(EPackage, EObject)
«create»
datastore::TeneoDataStore
~ TeneoDataStore(EPackage)
Figure 28 – Class diagram with Pipe, Request and Handler

Figure 29 shows all the code that is needed to save a harmonization instance to the landing area. Hibernate will take care the creation of the SQL code using the mapping specification that Teneo generated out of the EMF model.


```
@Override
public void save(Object object) {
    // Create a session and a transaction
    SessionFactory sessionFactory = this.dataStore.getSessionFactory();
    {
        final Session session = sessionFactory.openSession();
        session.setCacheMode(CacheMode.IGNORE);
        session.beginTransaction();
        session.save(object);
        session.getTransaction().commit();
        session.clear();
        session.close();
    }
}
```
Figure 29 –Source code to save an instance of the persistance model

The handlers between the parser and the saver implement the business logic that transforms the input model to the harmonization model. This design enables the extension of the communication components so that DACTyL can interact and benefit from additional systems. It also enables every possible combination between input and harmonization models.

Possible additional systems are an Enterprise Master Person Index (EMPI) and/or a clinical vocabulary server (see 12.4). These additional systems are not part of this deliverable for two reasons: First the time budget for this project is limited and second the project is not mature enough to benefit from these systems. Nevertheless these future expansions were kept in mind during the design and implementation of the software components. The integration with such systems requires

1. The implementation of a new handler.

2. An update of the pipe processing factory so that the new handler becomes available to web services.

Figure 30 shows how new data coming from the data contributor are handled. The setup of the objects is omitted to gain space. In this figure we consider the pipe already properly set up. The incoming data are stored in a request object and the request is put in the pipe. The parser handler will parse the data contained in the request. If the parsing is successful the parser outputs an object which is the de serialized xml string thus a new java instance of the input model. Following that the instance is stored in the request and the request gets forwarded to the next handler in the pipe. In this case the next handler is a timestamp adder. This handler will act on the instance object contained in the request and add a timestamp.

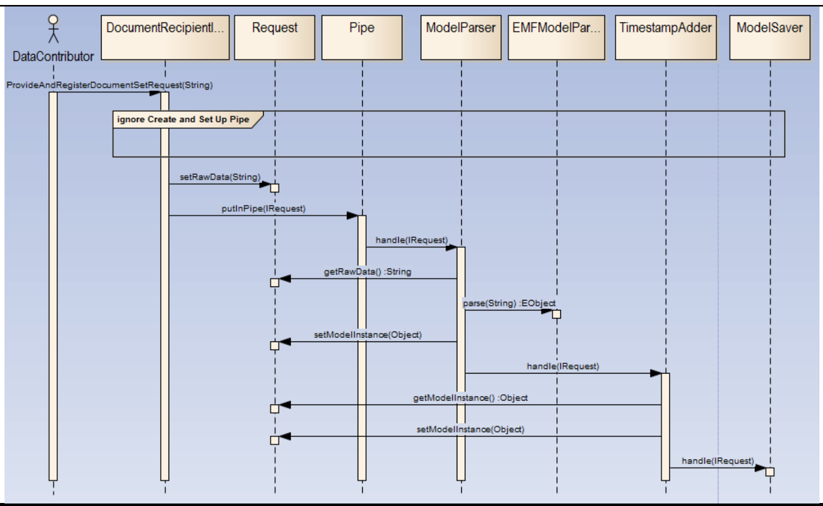


Figure 30 – A Request handled by the Handlers

The next handler is the saver. Figure 31 continues from where Figure 30 ended and shows how the saver handles the request. It will use the data store factory to create an instance of the data store and then ask from it to save the java instance of the model contained in the request. Again the object creation is omitted from the diagram.

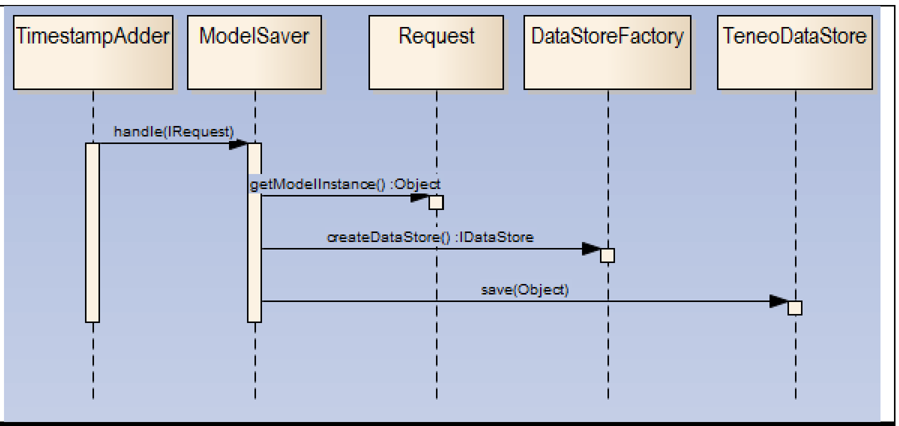


Figure 31 – Saving the data

The software components communicate through interfaces. The implementation of the interfaces uses the external frameworks. The following list summarizes the benefits and the drawbacks related to the selection of external frameworks according to our judgment:

- Benefits
  - EMF
    1. The usage of EMF to design the input and harmonization model creates a single point of reference for these models.

        2. There is no need for a hand written parser of the input.
        3. Open source models for healthcare standards can be easily reused.
    - Teneo
        1. No need for SQL code to create a database schema for the harmonization model.
        2. No need for SQL code to save an object to the landing area.
- Drawbacks
    - The external frameworks add dependencies to the project that need to be managed.
    - Since the landing area schema is generated out of the harmonization model the designer of this model should understand how the object relational mapping works.

9.4 Implementation of the linking layer

This section describes the tasks related to implementation of the linking layer. This layer is responsible for loading the data marts with data. Here we describe the data mart design, the landing area design and the ETL process.

9.4.1. Implementation of the landing area

The landing area is the database that stores instances of the harmonized model. In this version we used the Oracle 11g database for the landing area. The connection layer can store data in every database supported by Hibernate but since the Oracle database was already installed for the data warehousing part it made sense to reuse it for the landing area. The design of the database schema is generated by the harmonization model. Using the harmonization model we instruct Teneo to create a database schema capable of saving an instance of the harmonized model. Teneo reads the harmonization model and produces a Hibernate mapping file. Then it issues the appropriate SQL commands to create the table and establish the foreign key relationships.

We create one database user[10] per hospital. The connection layer uses the "contributor" attribute to realize where the data should be delivered. This means that if we receive data from hospital A and hospital B, the landing area contains two structurally identical databases, one for user A and one for user B. The databases are structurally identical because they are both generated from the harmonization model. As long as the data contributed by the hospitals can be accommodated by the harmonization model no change is necessary. This is why the design of the harmonization model plays an important role in a real case scenario. This model must be generic enough to cover Philips' data needs from the hospitals. In other words the harmonization model must be considered more stable than the input model.

9.4.2. Implementation of the Data Warehouse bus

The data warehouse bus is a collection of dimension tables that are referenced by fact tables belonging to different data marts. These are the conformed dimensions of the data warehouse that enable the drill across operations.

In order to better manage the conformed dimensions we create a dedicated database user. This user owns three tables, which are the three conformed dimensions in our system

1. DIM_CALENDAR: The time dimension.

[10] A database user in Oracle terms should be translated by the reader to a database schema. An Oracle database user can own tables and other database objects such as views, procedures and sequences.

2. DIM_LOCATION: A dimension that holds all locations that are contributing data.
3. DIM_PATIENT: A dimensions that holds all the patients in the system.

The DIM_LOCATION and DIM_CALENDAR dimensions originate from the Motiva data mart. The table structure and data are copied in the data warehouse bus user. There is only one addition in the DIM_CALENDAR dimension and this is the “null” row. This row represents the unknown date. It is necessary because many hospitalization events will not contain a discharge or a mortality date. These dimensions are set up at configuration time.

They do not change and they do not get updates from the incoming data, however new location can be added as the customer base grows. The patient dimension on the other hand, can receive new data from the hospitalization data mart and the Motiva data mart. To assign a unique system-wide identifier we use a sequence that produces number for the DIM_PATIENT.NID column. If an ETL process needs to add a new row to the patient dimension, it has to use this sequence to produce the primary key. Figure 32 shows the details of the conformed dimensions in the data warehouse bus.

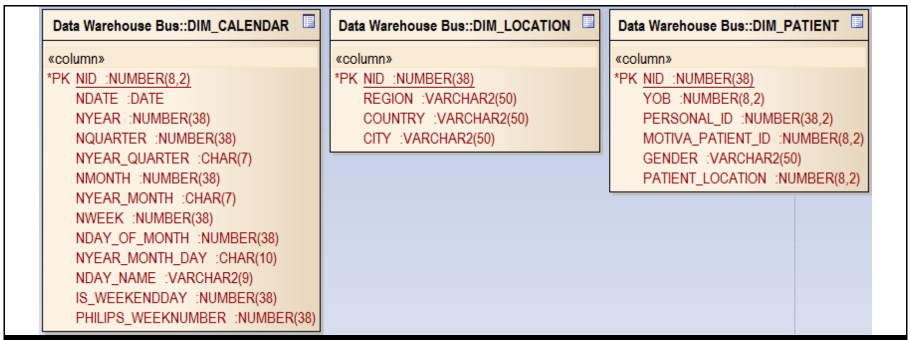


Figure 32 – Conformed dimensions in the data warehouse bus

The Motiva data mart, which was developed last year, and the hospitalization data mart developed in this project contain fact tables with foreign key relationships to these conformed dimensions. Figure 33 shows the relationship.

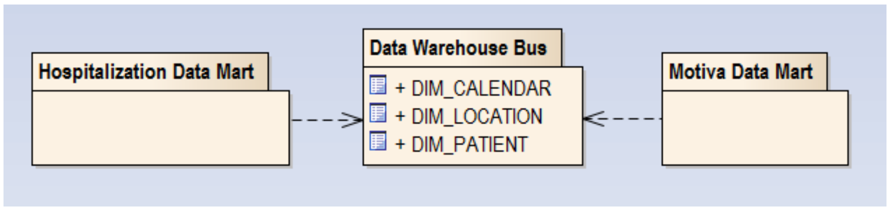


Figure 33 – The data warehouse bus and the data marts

9.4.3. Implementation of the Hospitalization data mart

The hospitalization data mart is a database that structures the hospital data in a particular way. The hospitalization fact table captures a business event, which in this case is the occurrence of a hospitalization. The measurement for this table is the Length of Stay (LOS) in days. The dimensions add context to the event and can provide answers to questions such as “where this event took place,” “who is the patient” and “what was the diagnosis.” The fact table is designed as an accumulating snapshot table according to [10]. The summary of the data mart is shown in Table 18.

Table 17 – Summary of the hospitalization data mart

Fact Table Type	Accumulating Snapshot

Measurements	Length of Stay in days
When Dimensions	DIM_CALENDAR
Who Dimensions	DIM_PATIENT DIM_AGE DIM_GENDER DIM_SERVICE
Where Dimensions	DIM_LOCATION
Why & How Dimensions	DIM_DIAGNOSIS DIM_DIAGNOSIS_GROUP DIM_ADMISSION_TYPE DIM_TREATMENT

Figure 34 shows the foreign key relationships of the fact table. The fact table has foreign key relationships with data mart specific dimensions and with the conformed dimensions in the data warehouse bus.

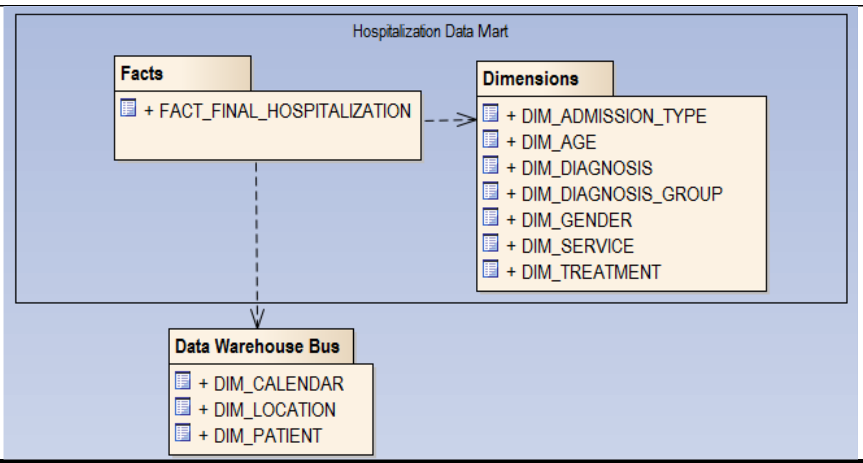


Figure 34 – Hospitalization data mart foreign keys

The fact table has a multirole relationship with the calendar dimension[11]. This practically means that the fact table has three columns; admission, discharge and mortality date that are foreign keys from the calendar dimension. This multirole relationship enables the fact table to mark the important timestamps for the lifespan of the event.

As an example consider the following scenario: At time t DACTyL receives data that contain an admission for a particular patient. A new row will be added in the fact table and the length of stay column will be null because there is no discharge date yet. At time t+i DACTyL receives data that contain a discharge date for the same event. The fact table row will be now updated with the new value and the length of stay will contain the length of the hospitalization. This revisit and update pattern classifies the table as an accumulating snapshot table. Figure 35 shows the details of the tables in the hospitalization data mart. We see the usage of the conformed dimensions and the multirole relationship with the calendar dimension.

A hospitalization event can be related with more than one diagnosis. This is why the fact table contains one foreign key to the diagnosis dimension for the primary diagnosis and one additional foreign key to the diagnosis group dimension. This table is used together with the diagnosis group bridge table to relate one hospitalization event with multiple diagnoses.

[11] The calendar dimension is said to be a role playing dimension.

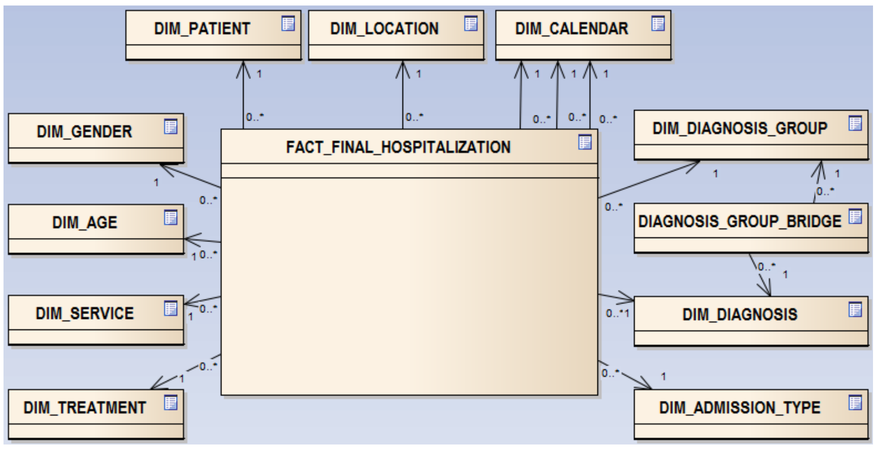


Figure 35 – The hospitalization data mart

Apart from the DIM_DIAGNOSIS_GROUP all the other dimensions of the hospitalization data mart are set up during the configuration time and they do not receive new updates from the incoming data. One dimension with particular interest is the DIM_DIAGNOSIS. To create this dimension the ICD-10 vocabulary is used. To populate the table we designed an ETL process to import the data published by world health organization.

The DIM_TREATMENT and DIM_ADMISSION_TYPE are case specific vocabularies. The codes found in these vocabularies are used in the data set that we used as input but, according to our research, they do not map to any standardized clinical vocabulary.

9.4.4. Implementation of the ETL process

In this section, we describe the ODI interfaces[12]. These interfaces perform the loading of the hospitalization data mart with data coming from the landing area. The designed interfaces are

1. load_dim_patient
2. load_dim_diagnosis_group
3. load_dim_diagnosis_group_bridge
4. load_fact_final_hospitalization

The following sections provide descriptions for the interfaces. The implementation details can be found in the Appendix A section 15.1.

9.4.5. The “load_dim_patient” interface

The “load_dim_patient” interface has to check if the data in the landing area contain new patients. The source table is the EVENT_DATA table in the landing area. The target table is the DIM_PATIENT table in the data warehouse bus. The implementation of the interface defines which columns play the role of the update key in the target table. The update key is used to define if the data on the source table already exist on the target table. If the update key already exists then the source data row will be treated as an update. Otherwise the source data row will be considered as data for a new patient and a new row will be added in the DIM_PATIENT row. The surrogate key for the new patient row will be generated by a sequence and will identify the patient across the system.

For this interface the update key of the target table is composed from three columns:

[12] Again at this point the words interface stands for a mapping between a source and a target database table

1. PATIENT_LOCATION: This column is populated using ODI specific code that returns the value of the current context at runtime.
2. YOB: This column is populated form the PATIENT_BIRTHDATE column in the source data.
3. PERSONAL_ID: This column is populated using the PATIENT_UNIQUE_ID column in the source data.

9.4.6. The "load_dim_diagnosis_group"

The "load_dim_diagnosis_group" interface is responsible for creating one new row for every new hospitalization. The source table is the EVENT_DATA table in the landing area. The target table is the DIM_DIAGNOSIS_GROUP table in the hospitalization data mart. This is a quite simple interface that creates a new table row per hospitalization event. There are only insert and no updates for this interface.

9.4.7. The "load_diagnosis_group_bridge"

The "load_diagnosis_group_bridge" interface connects primary keys from DIM_DIAGNOSIS and DIM_DIAGNOSIS_GROUP tables. The source table is the EVENT_DATA table in the landing area. The target table is the DIAGNOSIS_GROUP_BRIDGE table in the hospitalization data mart.

9.4.8. The "load_fact_final_hospitalization" interface

The "load_fact_final_hospitalization" interface is responsible for loading the fact table with new data. The source table is a joined table between the EVENT_DATA, EVENT_DATA_DIAGNOSIS and DIM_PATIENT tables. The target table is the FACT_FINAL_HOSPITALIZATION table. Additionally one lookup operation on a dimension table is required per foreign key contained in the fact table.

The join between the EVENT_DATA and the EVENT_DATA_DIAGNOSIS tables is necessary for the lookups that load the foreign key to the primary diagnosis and the foreign key to the diagnosis group. The join between the EVENT_DATA and the DIM_PATIENT is necessary for the lookups that load the foreign key to the gender and the foreign key to the age dimension.

Since this fact table is an accumulating snapshot fact table every row will be possibly revisited and updated. The update key for this table is composed from two columns:

1. LOCATION_NID: This column is populated using ODI specific code that returns the value of the current context at runtime.
2. EVENT_ID: This column is populated from the HOSPITAL_EVENT_ID column in the source data.

Every time this interface runs it checks if the combination of LOCATION_NID, EVENT_ID exists already in the fact table. If this is true the row of the fact table gets updated and in the opposite case a new row is inserted in the fact table.

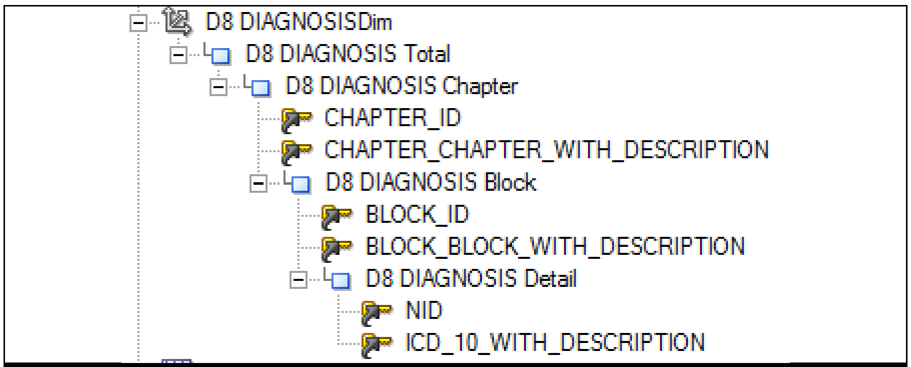
9.5 Implementation of the presentation layer

The presentation layer provides access to the data. The data analyst interacts with this layer to gain insights from the data. This access is realized through predefined reports such as the return on investment and patient report. Additionally applications can query the OLAP cube directly.

9.5.1. Implementation of the OLAP cube

To implement the OLAP cube we used the Oracle BI Administrator tool. The physical layer contains the data warehouse bus, the hospitalization data mart and the Motiva data mart tables.

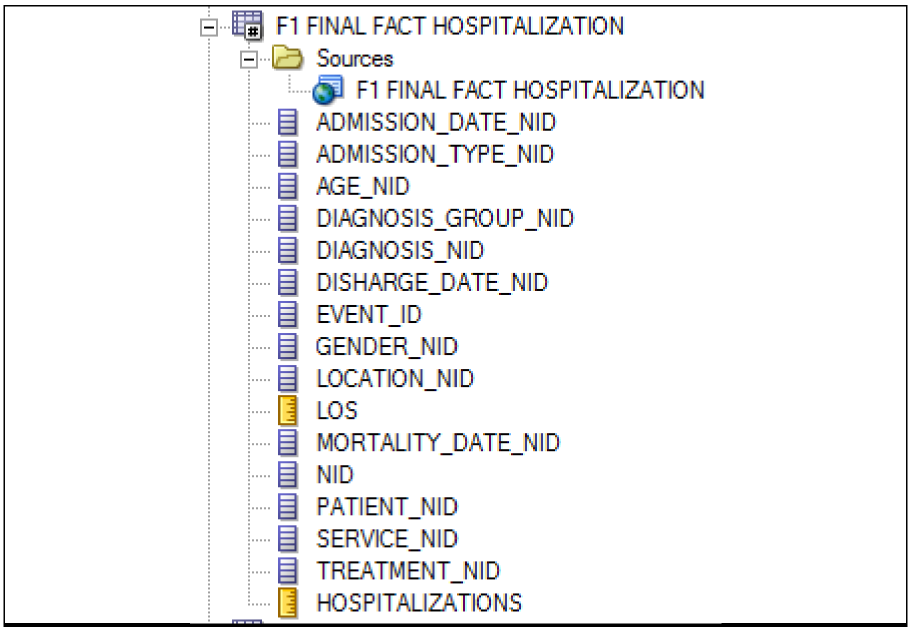
The business model contains all the dimension and fact tables. Additionally in the business model we specify the dimension hierarchies for every dimension. The dimension hierarchies are necessary to meet the reporting requirements. As an example Figure 36 shows the hierarchy definition for the diagnosis dimension. We see a top “Total” level and then the three levels of the ICD-10 hierarchy “Chapter”, “Block”, “Detail”. The “Detail” level corresponds to the actual ICD-10 code.



```
D8 DIAGNOSISDim
  D8 DIAGNOSIS Total
    D8 DIAGNOSIS Chapter
      CHAPTER_ID
      CHAPTER_CHAPTER_WITH_DESCRIPTION
      D8 DIAGNOSIS Block
        BLOCK_ID
        BLOCK_BLOCK_WITH_DESCRIPTION
        D8 DIAGNOSIS Detail
          NID
          ICD_10_WITH_DESCRIPTION
```

Figure 36 – Hierarchy for the diagnosis dimension in the business model

In the hospitalization fact table, we add one logical column that counts the number of table rows. This column does not affect the actual physical table. It exists only in the business model. This means that the OLAP cube treats this column as if it was a real physical column. With this addition the hospitalization fact table has two measures, the length of stay and the number of hospitalizations. The two measures are shown in yellow color in Figure 37.



```
F1 FINAL FACT HOSPITALIZATION
  Sources
    F1 FINAL FACT HOSPITALIZATION
  ADMISSION_DATE_NID
  ADMISSION_TYPE_NID
  AGE_NID
  DIAGNOSIS_GROUP_NID
  DIAGNOSIS_NID
  DISHARGE_DATE_NID
  EVENT_ID
  GENDER_NID
  LOCATION_NID
  LOS
  MORTALITY_DATE_NID
  NID
  PATIENT_NID
  SERVICE_NID
  TREATMENT_NID
  HOSPITALIZATIONS
```

Figure 37 – Hospitalization fact table in the business model

We reuse this approach to add logical columns wherever is necessary. For example we add a logical column on the fact table of the Motiva data mart to count the number of Motiva tasks. Since the need to count rows is a common thing in data warehouses some authors propose to add an additional physical column on the fact table which always contains the number one. We found the approach of logical columns more elegant for our case. Of course by choosing this approach we depend on a feature of the Oracle BI Administrator tool.

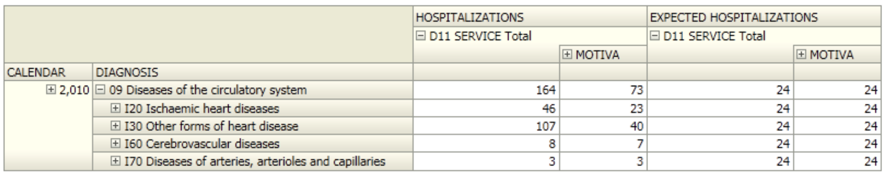
One additional task performed with this tool is column renaming. We change the physical names of the columns to user friendly text. This activity takes place on the presentation model or business model.

With the physical, business and presentation model specified the configuration of the OLAP cube is complete. The configuration is stored on a file which is used by the Weblogic server (see Chapter 11). Every element of the presentation model becomes available for usage by the data presenter.

9.5.2. Implementation of the data analyst reports

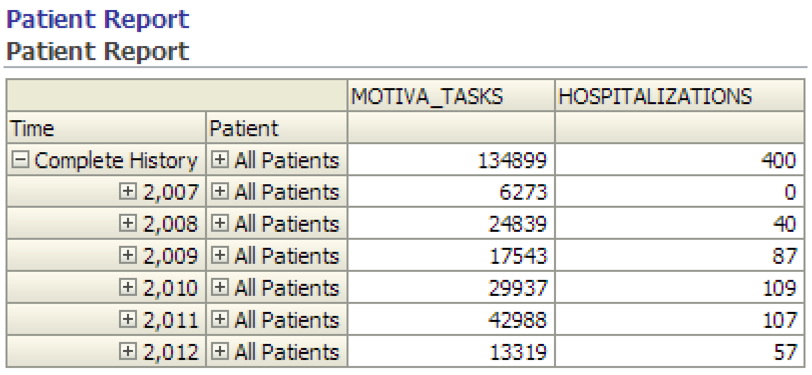
The last implementation activity for this project was the creation of the end user reports. This is the activity done by the data presenter. Oracle uses the term dash-boards to describe them. The dashboards are compositions of different elements such as interactive tables, user prompts and graphs.

To create the dashboards according to the presentation requirements we use the available elements from the presentation model. Figures 38 and 39 show the end user reports. Figure 38 shows only a few columns due to lack of space. The expected hospitalization value is a constant value based on the user input.



		HOSPITALIZATIONS		EXPECTED HOSPITALIZATIONS	
		⊟ D11 SERVICE Total		⊟ D11 SERVICE Total	
			⊞ MOTIVA		⊞ MOTIVA
CALENDAR	DIAGNOSIS				
⊞ 2,010	⊟ 09 Diseases of the circulatory system	164	73	24	24
	⊞ I20 Ischaemic heart diseases	46	23	24	24
	⊞ I30 Other forms of heart disease	107	40	24	24
	⊞ I60 Cerebrovascular diseases	8	7	24	24
	⊞ I70 Diseases of arteries, arterioles and capillaries	3	3	24	24

Figure 38 – Return on investment report



Patient Report
Patient Report

		MOTIVA_TASKS	HOSPITALIZATIONS
Time	Patient		
⊟ Complete History	⊞ All Patients	134899	400
⊞ 2,007	⊞ All Patients	6273	0
⊞ 2,008	⊞ All Patients	24839	40
⊞ 2,009	⊞ All Patients	17543	87
⊞ 2,010	⊞ All Patients	29937	109
⊞ 2,011	⊞ All Patients	42988	107
⊞ 2,012	⊞ All Patients	13319	57

Figure 39 – Patient report

A presentation requirement is met not only by the end user report. The report is only the tip of the iceberg. A simple example will clarify how a presentation requirement is met.

The return on investment report must contain a time column which can be used for drill down operations according to the requirements. To meet this simple requirement several implementation actions in different layers are required:

1. The input model accommodates the admission date for a hospitalization event.
2. The persistence model and the landing area design enable the storage of this date.
3. The design of hospitalization data mart contains a time dimension table properly loaded with data.
4. The design of the hospitalization data mart specifies that the hospitalizations fact table has a foreign key to the time dimension to mark the admission date.

5. The ETL interface that loads the hospitalization fact table uses the admission date stored in the landing area. It performs a lookup operation on the time dimension table to find the dimension key that corresponds to this date. This dimension key is stored in the hospitalization fact table.
6. The configuration of the OLAP cube specifies that the physical time dimension table is a logical dimension with a specified hierarchy. The specification of the hierarchy enables the drill down features. The logical time dimension table is made available to the data presenter by being part of the presentation model.
7. Finally the dashboard uses this time dimension in a report. This column offers drill down features and connects hospitalization events with admission dates.

■

10. Verification & Validation

10.1 Introduction

This chapter describes the verification and validation [26] for the DACTyL project. To clarify the terms we distinguish them as follows

- Verification: Are we building the product right?
- Validation: Are we building the right product?

10.2 Verification

This section describes how each layer was tested. Due to the different technologies involved we adjusted our approach per layer.

10.2.1. Connection layer

The connecting layer is composed of java classes. To test these classes we used the JUnit [27] framework. Whenever a connection with a database was necessary we used the Hyper SQL [27] database to keep the Oracle database clean from testing data. We followed a white box approach for the unit testing and a black box approach for the subsystem test.

- Unit test: Individual classes where tested in isolation.
- Subsystem test: To test the layer as a subsystem we submitted a set of data and successfully retrieved them from the landing area.

10.2.2. Linking layer

The ETL interfaces where tested using a black box approach due to the nature of the ODI tool. ODI provides an execution report when an interface executes successfully. The execution report contains the number of rows inserted, updated and deleted. We tested the ETL interfaces in two different levels

- Unit test: Every interface was executed in isolation. The expected number of inserts, updates and deletes was compared to the actual one reported by ODI. This testing method does not reveal unsuccessful lookup operations. We used SQL scripts to count the number of null values in the target table columns.
- Subsystem test: To test the complete ETL process we used the same method but instead of executing every ETL interface in isolation we executed an ETL scenario which is a sequence of ETL interfaces.

During testing with the Hull data set we found out that there is number of disease codes which could not be matched. Further investigation proved that these codes come from an addition to the ICD-10 vocabulary which is not included in the official version published by WHO. In the future work section we advise the inclusion of these additions as well.

10.2.3. Presentation layer

The BI Administrator tool which configures the OLAP cube offers a consistency check mechanism. Once the configuration is specified the tool can check if the configuration is consistent. This consistency checking mechanism does not ensure correctness of the OLAP configuration.

- System testing: The end user reports are in fact a black box testing tool for the complete system. Interacting with the reports reveals the errors in the OLAP cube configuration.

10.3 Validation

This section summarizes how the must requirements for this project are met.

Table 18 – Connecting Requirements

Id	Name	Priority	Description
C1	Input Data	M	The system input shall be: • Patient demographics • Admission date • Discharge date • Admission reason • Diagnosis • Mortality date • Patient unique identifier on the source system
Met By	The input data are accommodated by the input model and parsed by the parser handler. Sections 9.2 and 9.3 describe the details.		

Table 19 – Connecting Requirements

Id	Name	Priority	Description
C3	Operational Source Location	M	The system shall accept an input data from operational sources that are physically situated in a different network.
Met By	The requirement is met by the web service and the web container. The data contributor can send data from a different network. Section 9.3 describes the details.		

Table 20 – Connecting Requirements

Id	Name	Priority	Description
C4	Updates	M	The system shall accept input data from operational sources periodically.
Met By	The requirement is met by the web service and the web container. The data contributor can send data from a different network. Section 9.3 describes the details.		

Table 21 – Linking Requirements

Id	Name	Priority	Description
L1	Unique DAC-TyL specific patient identifier	M	The system shall assign a unique DACTyL specific identifier to every patient it contains.
Met By	The requirement is met by database sequence that generates surrogate keys for the DIM_PATIENT table and the ETL interface that loads the DIM_PATIENT dimension. Sections 9.4.2 and 9.4.5 describe the details.		

Table 22 – Linking Requirements

Id	Name	Priority	Description
L2	Data Linking	M	The system shall link all available data about a specific patient.
Met By	The requirement is met by the foreign key relationship between the fact tables that contain the data and the DIM_PATIENT table that uniquely identifies every patient. Sections 9.4.2 and 9.4.8 describe the details.		

Table 23 – Linking Requirements

Id	Name	Priority	Description
L3	Data Linking Update	M	The system shall link additional data about a specific patient when they become available.
Met By	The requirement is met by the design of the fact table as an accumulating snapshot and the foreign key relationship between the fact table that contains the data and the DIM_PATIENT table that uniquely identifies every patient. Sections 9.4.2, 9.4.3 and 9.4.8.		

Table 24 – Presenting Requirements

Id	Name	Priority	Description
P1	Combined reporting	M	The system shall produce a report that contains data from Motiva and from a clinical system for a specific patient.
Met By	The requirement is met by the conformed dimensions between the Motiva and the hospitalization data mart. Section 9.4.2 describes the details.		

Table 25 – Presenting Requirements

Id	Name	Priority	Description
P2	Return on investment report	M	See 6.2.4
Met By	The requirement is met by the return on investment report. Section 9.5.2 describes the details.		

Table 26 – Presenting Requirements

Id	Name	Priority	Description
P3	Patient report	M	See 6.2.5
Met By	The requirement is met by the patient report. Section 9.5.2 describes the details.		

■

11. Deployment

11.1 Introduction

This chapter describes the deployment of the implementation and the proposed deployment for future versions of the system.

11.1.1. Implementation deployment

Figure 40 shows how the proof of concept system is deployed. Everything is currently hosted on a pc running 64-bit windows 7. The connection layer uses the Apache Tomcat and the axis2 execution environments. Within axis2 is the web service XDR.aar that implements the layer described in Section 9.3. The service.xml is a deployment specification that connects the web service with the web container. The web service is deployed on the appropriate directory and executed every time the container receives data.

This device also hosts an instance of an Oracle 11g database. This database contains the landing area and the data warehouse with the data marts as described in subsections 9.4.1-9.4.3.

The ODI tool executes the loading scenario and loads the data warehouse with new data. The loading scenario is designed at configuration time and executed periodically by execution agents. The loading scenario executes the interfaces describe in subsections 9.4.5-9.4.8.

The BI administrator tool produces an OLAP cube configuration which is designed at configuration time and deployed on the web logic server as described in subsection 9.5.1. The reports and the dashboards that the end user uses are designed at configuration time and deployed on the web logic server as described in sub-section 9.5.2.

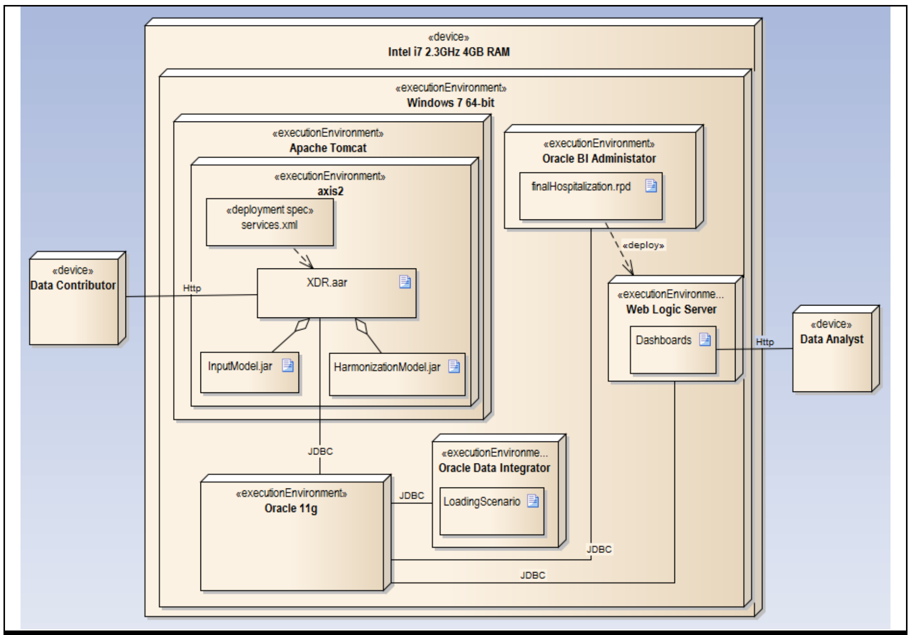


Figure 40 – Deployment diagram

11.1.2. Proposed system deployment

The current deployment of the system in one laptop makes it easily presentable to management and potential clinical partners. If the intension is to continue exploratory development we propose to transform the physical system to a virtual one. The snap-shotting features of the virtual machine make versioning the entire system easier.

For a realistic deployment we propose the following changes:

1. The connection layer which contains the Apache Tomcat and the web service should be deployed in a separate server. This server will receive the incoming data from the data contributors. Since it publishes a web service it requires special security configuration.
2. The server that contains the connection layer should also host the landing area. This means that the server should host an instance of Oracle or another general purpose DBMS system. The reason for this is that the web service responds after the data are successfully saved in the landing area. The web service will respond faster, if the landing area is also hosted in the same server.
3. A dedicated server should host the data warehouse. This will enable easier maintenance and faster responses from the data warehouse.
4. The ODI tool can connect to any remote database system. This means that the ODI tool can be simply hosted on the system of the data integrator. We propose that the master and work repositories for the ODI tool are hosted in the same database instance as the data warehouse. We also propose that the staging area used by the tool should be in the server that hosts the data warehouse. These recommendations are in line with the Oracle's suggested layout and increase the performance of the ETL scripts.
5. The same holds for the BI Administrator tool. It can be hosted on the system of the data integrator and connect to the remote databases.
6. The OBIEE environment which contains the web logic server and the end user dashboards should be deployed in a separate server. This server exposes the end user reports and special security configurations should apply here too.

■

12. Conclusions

This chapter summarizes the main results of the DACTyL project and discusses lessons learned and future work. The goal of this project was to investigate possible designs for an analytical system that enables external data sources to submit data, combines them with Motiva data and generates specific reports for different stakeholders.

12.1 Results

We group the results of this project in two main categories, the results from the domain analysis and the result from the design and implementation tasks.

12.1.1. Domain analysis

Hospitals use different systems to store patient data. These systems use proprietary data models. We cannot predict what data will be available at a particular hospital and in which format.

The greatest common factor between hospital-IT layouts is the interface engine. This system handles communication between different systems and has features that facilitate the data exchange. Hospitals use different interface engines but the list of supported features does not vary dramatically. The most interesting feature for our project is the connection to a remote web service. We propose that the communication between DACTyL and the hospital is realized through the interface engine.

Hospitals use managed vocabularies. These vocabularies offer a coding system. This project used the ICD-10 vocabulary to categorize hospitalization events because it was the coding system of the input data set. There are multiple coding systems used in practice. It is important that the coding system is interpreted by the receiving part in a data exchange scenario.

The HL7 CDA document is the state of the art standard that offers semantic interoperability. It is an abstract data model which does not specify concrete messages. According to hospital IT vendors the standard in still not widely supported. This is due to fact that hospitals do not share information across their organizations boarders. In countries where this data model is used, a country specific version is published and enforced to the communicating parties by an external, usually government related, organization. Further research is necessary to identify if the usage of particular version of the HL7 CDA such as the CCD or the specification of a new version would benefit the DACTyL project. This research should explore if the systems of the contributors can export data in these formats and if not what would be the cost of adapting their systems.

The IHE provides specification for the process of data exchange. If hospitals decide to create data sharing clusters it is probable that they will follow the specifications of IHE. Our envisioned system could play a role in such a cluster.

12.1.2. Implementation

To implement the system we defined our use case the roles that interact with the system and decomposed the requirements in three different areas. Apart from the obvious roles of the data contributor, who shares his data, and the data analyst, who uses the system to gain insights, we identified roles related to the configuration of the system. We conclude that due to the heterogeneity of the domain the system would require additional configurations and adaptations as more contributors want to share data and new reporting requirements come from the stakeholders. These roles change the behavior of the system and adapt it to new requirements. It is therefore important to keep them under scope.

DACTyL is composed of three different layers. We conclude that this decomposition is necessary to have clear separation of concerns within the system. Additionally this approach enables the parallel development of the different layers.

The design of the connection layer carried the weight of dealing with the heterogeneity. We conclude that this layer must provide connectivity to the data contributors, accept different input models and perform transformations on the received data. The software developed in this project for the connectivity layer establishes a solid starting point for further development. An alternative approach can be the usage and configuration of a commercial interface engine if alignment with Philips preferred technology stack is necessary.

The linking layer contains the data marts which store the user accessible data. We conclude that the data warehouse bus architecture is suitable for this use case. As the system grows new data marts will be necessary to meet stakeholders' requirements. The bus architecture is the proper mechanism to connect the new with the old data marts.

The presentation layer contains the OLAP cube and the end user reports. We conclude that this is the suitable approach to make the data warehouse available to end users. The end user reports can cover the requirements of the data analysts and the data researchers can create their own applications that directly query the OLAP cube.

We managed to meet our requirements and produce the required reports by implementing and testing the DACTyL system. The produced reports compare real versus expected hospitalizations. The difference can be used to deduct the return on investment.

We conclude that the tree-layered analytical system described in this document is a feasible approach, with respect to the functional and nonfunctional requirements, to realize the communication with hospital information systems and meet the reporting requirements of the stakeholders.

12.2 Limitations

A number of diagnosis codes were not matched because they follow the ICECI adaptation of the ICD-10 vocabulary. This adaptation adds additional codes to the official vocabulary. It is managed by a different standardization body, thus it is not part of the official vocabulary published by the WHO. To overcome this limitation we propose the inclusion of the ICECI adaptation.

One element that is currently missing from the Motiva data mart, that we reused, is the status of a Motiva patient. Once a patient is stored as a Motiva patient he always stays a Motiva patient. There is no information about patients who drop out. To overcome this limitation we propose further communication with the Motiva team to identify where can we find the patient status.

With the available input data set the system can measure and categorize the hospitalization events. To calculate the expected hospitalizations we need additional data. The missing data are the total population of patients and the expected re-admission percentage for a particular diagnosis. In this version of DACTyL these numbers can be provided as input from the data analyst through user prompts. According to our view such constant values are expert knowledge and should be always provided through input prompt. We would not include them in the input model since they are not related with an event.

12.3 Lessons Learned

Combining telehealth with clinical data is an innovative use case. Deploying a generic solution in a big scale to perform this analysis, on an ongoing manner, across different hospitals has not been endeavored before according to our domain research. This solution can bring unique insights and competitive advantage to Philips.

Information exchange in the healthcare domain is still in a developing state due to the heterogeneity of the domain. There are proposals on how it should be done but this does not necessarily mean that hospitals are ready to share information. There is no one-size-fits-all solution when it comes to communication with hospitals. A successful design must be able to adapt and reduce the propagation of change within the system.

The ETL tools work great on stable environments, where nothing changes. This is mainly due to the fact that the ETL process has dependencies on table and column names. The lesson to keep from this is that once the ETL is written any structural change on the database systems involved, brings the additional cost of updating the ETL.

12.4 Future Work

There are multiple future use cases and possible extensions. In this section we focus only on those which are closer to our project.

We have already mentioned that further research is necessary to investigate which standard will benefit DACTyL.

As mentioned earlier the ICD-10 vocabulary used in this project is one of the clinical vocabularies used in practice. Systems usually termed as **clinical vocabulary servers** or **terminology servers** handle the mappings between different vocabularies. This does not guarantee that there is always a one-to-one mapping between the terms described in the vocabularies. DACTyL could benefit from such a system if data contributors send data using different terminologies. To connect with such a system the connection layer should be extended with an additional handler.

One of the conformed dimensions used in DACTyL is the patient dimension. This physical table contains exactly one row for every patient known to the system. Such a table is also termed as a **master parson index** (MPI). In DACTyL this table is populated by an ETL process. The specification of the process makes sure that a unique identifier is generated for every new patient. Healthcare IT vendors offer dedicated systems for patient management which are termed as **enterprise master person indexes** (EMPI). These systems offer also a unique identifier for every patient and they can link together different MPIs and automatically identify duplicates. They offer advanced searching features which can use fuzzy logic to match patients. DACTyL could benefit from such a system in future cases when advanced patient management is needed.

As these lines are written, Philips arranged a use case study with a hospital in order to further test and evolve DACTyL. We believe that this is the best approach to further continue this project.

■

13. Project Management

13.1 Introduction

Project management proved to be a challenging task for the DACTyL project. Since this was an exploratory project the requirements were unstable and we had to adapt them as we learned more about the domain and the technologies around it. Most of the domain concepts that we encountered during this project are new for the team. This chapter documents the process and the methodologies followed.

13.2 Process

DACTyL is a system that combines pure software components, data warehouse design and implementation, ETL process design and BI report creation in a healthcare environment. Since most of the concepts were new to us in combination with the lack of concrete requirements and domain experts the use of a waterfall approach was impossible. We worked iteratively throughout the duration of the project setting realistic goals.

In the early stages of the project we combined domain analysis and experimentation. We designed small experiments to try out different approaches, evaluate risks and gain experience with the provided tools and systems (see Chapter 5). The experiment documented in Chapter 5 is itself a composition of a number of smaller experiments. During this period we combined the activities of reading documentation, online sources and articles, trying out open source healthcare systems and completing tutorials for the tools mentioned in the constraints (see Chapter 6).

The knowledge and experience gathered from these activities contributed in the specification and decomposition of the requirements (see Chapter 6) in the three areas namely connecting, linking and presenting. Following the requirements specification we designed the layers that meet every requirements set (see Chapters 7 and 8). We worked iteratively on every layer to improve the provided features and meet the deliverables of each month.

13.2.1. Planning and Scheduling

The deliverables for this project were planned in monthly bases. Every month contained a number of deliverables. The types of deliverables are: documentation chapters, completed experiments and system versions. The detailed planning is documented in the work breakdown structure section. At the end of every month the plan for the coming deliverables was revised and updated.

13.2.2. Communicating with supervisors

For the first two months of the project twice per week we had a progress update meeting with the two company supervisors. After February this was replaced with a weekly meeting every Tuesday. During the progress update meeting the following tasks was performed:

- Report on the progress of the deliverables for the current month.
- Presentation of intermediate results-findings.
- Decide if a drop or an addition in the deliverables is necessary.

Once per month during the project steering group meeting (PSGM) all the deliverables for the past month were presented. During a PSGM meeting the following tasks were performed:

- Quick revision of the previous PSGM meeting.
- Presentation of the current position in the project timeline.
- Presentation of the deliverables of the past month.

- Presentation of the tasks for the coming month.
- Discussion over the must tasks of the coming month.

13.2.3. Acceptance Control

For the different system versions after the presentation of the system demo the supervisors provided feedback on improvements and future features. This review meeting took place at least once per month during the PSGM but also during the presentation of the intermediate results in a progress update meeting.

The documentation chapters of the final report were forwarded to the supervisors for feedback and comments.

13.2.4. Configuration Management

The software components are version controlled using the SVN infrastructure of Philips Research. Before the migration to the "production" computer we developed the system in a virtual machine using Virtual Box [16]. This choice proved beneficial during the experimentation phase because we could easily try out different things and revert to a prior stable state.

13.3 Work breakdown structure

Figure 41 shows the time line of the project with the major milestones. The timeline shows only the first version of the requirements and architecture documents to pinpoint a beginning. These two chapters were updated after every new version of the system as we accumulated additional knowledge on the domain.

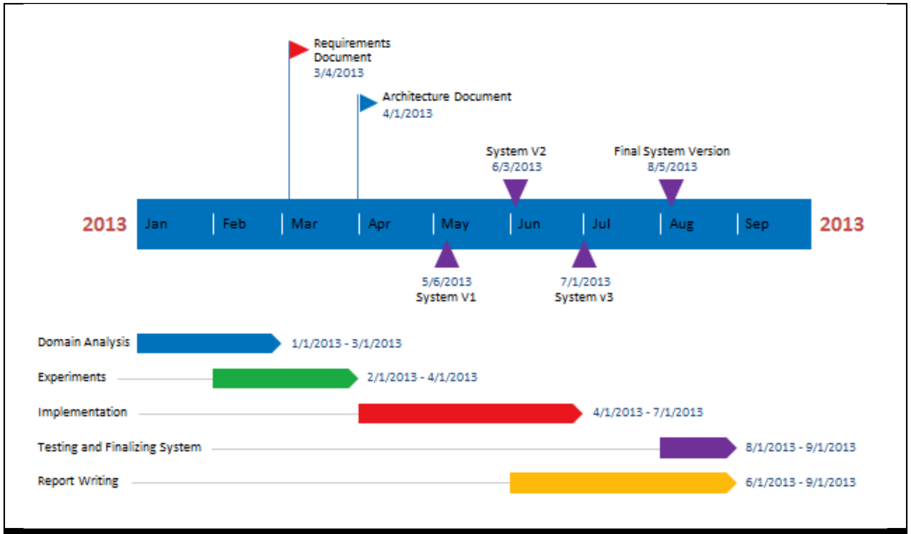


Figure 41 – Project timeline

The activities per month are described in the following lists. The percentages are approximations of the time spend on each activity. The day to day activities where managed using an online to do list.

- January
  - Analytical systems in healthcare 25%
  - Taxonomy of hospital information systems 50%
  - Healthcare standards 25%
- February
  - Tutorials on data warehouse implementation 20%
  - Experiments with interface engines 20%
  - Experiments with web services 10%
  - Specifying requirements 50%
- March
  - Specifying architecture 50%

    - Experiments with EMF models 25%
    - Experiments with Teneo 25%
- April
    - Implementing System version 1 100%
- May
    - Implementing System version 2 100%
- June
    - Implementing System version 3 60%
    - Report writing 40%
- July
    - Report writing 50%
    - Vacations 50%
- August
    - Report writing 50%
    - Testing and Finalizing system 50%

13.4 Project Retrospective

The past nine months can be described as an intense experience. This section contains the reflection of the author on the course of the DACTyL project. We enumerate a number of strong and improvement points.

13.4.1. Strong Points

- Exploring a huge domain. We believe that we singled out concepts of the domain that are relevant for this project and for the future versions.
- Documenting decisions. This report contains a detailed explanation of our design decisions. We explained our rational and also document alternatives wherever possible.
- Adopting technology. In this project we worked with different tools and had to learn and adapt fast.
- Working incrementally. We created a basic version and then build the prototype incrementally focusing on a different layer per iteration.
- Managing meetings. We managed to keep the meetings organized and within the estimated duration.

13.4.2. Improvement Points

- Locating domain experts. During this project we were unable to locate people that could give concrete black or white answers to our domain related questions. Usually the people we interviewed had either a too abstract knowledge of the domain or a too detailed which was unrelated to our goals. We could have allocated more time in searching for the right people than searching for online resources.
- Estimating documentation effort. We underestimated the documentation effort. We could have avoided this pitfall if the documentation effort had started earlier.

■

Glossary

Term	Description
BI	Business Intelligence
CCD	Continuity of Care Document
CDA	Clinical Document Architecture
CDO	Care Delivery Organization
DA4HH	Data Analytics for Home Healthcare
DACTyL	The name of the project. It is also used as the name of the developed system.
DAF	Data analysis framework
Data warehouse	A collection of systems and processes that enable reporting and data analysis.
Dimension	In a data warehousing context, Dimensions provide structured labeling information to otherwise unordered numeric measures. The dimension is a data set composed of individual, non-overlapping data elements. The primary functions of dimensions are threefold: to provide filtering, grouping and labeling.
Drill up/down	Drilling down stands for browsing information in a more detailed level. Drilling up is the opposite.
EHR	Electronic Health Record
EMR	Electronic Medical Record
Fact	In a data warehousing context, a fact is a value or measurement, which represents a fact about the managed entity or system.
Fact table	A database table that contains facts and references to dimensions.
H2H	Hospital 2 Home
HIS	Hospital Information Systems
HL7	Health Level Seven International
HL7 v3	The HL7 version3 standard
ICD-10	International classification of diseases version ten
IHE	Integrating Healthcare Enterprise
Level	In a data warehousing context, Dimension rows can be grouped in different levels of detail. For example a dimension for time could have the following levels: Year > Quarter > Month > Week > Day.
Motiva	Motiva is a secure, personalized healthcare platform that leverages consumer electronics and broadband to connect patients and their care providers, thereby enabling care models for patients in their homes.
Operational source	An external system such as a hospital information system.
RIM	Reference Information Model
TS	Telehealth systems
WHO	World Health Organization

Bibliography

[1] Philips, [Online]. Available: www.philips.com.

[2] D. F. Sittig, "A Survey of Informatics Platforms That Enable Distributed Comparative Effectiveness Research Using Multi-institutional Heterogeneous Clinical Data," *Medical Care,* vol. 50, 2012.

[3] C. Weng, "Using EHR's to integrate research with patient care: promises and challenges," [Online]. Available: http://www.jamia.bmj.com. [Accessed 2013].

[4] P. Groves, "The 'big data' revolution in healthcare," McKinsey & Company.

[5] T. G. T.D, "The Emergence of National Electronic Health Record. Architectures in the United States and Australia:Models, Costs and Questions," 2005.

[6] D. Garets, "Electronic Patient Records," [Online]. Available: http://www.providersedge.com/ehdocs/ehr_articles/Electronic_Patient_Records-EMRs_and_EHRs.pdf. [Accessed 2013].

[7] "Americam Medical Assosiation," [Online]. Available: www.ama-assn.org. [Accessed 2013].

[8] mcleodcg, [Online]. Available: www.mcleodcg.com.

[9] "Health Level Seven International," [Online]. Available: www.hl7.org. [Accessed 2013].

[10] "Integrating Healthcare Enterprise," [Online]. Available: http://www.ihe.net. [Accessed 2013].

[11] World Health Organization, "International Classification of Diseases (ICD)," [Online]. Available: http://www.who.int/classifications/icd/en/.

[12] C. Adamson, Star Schema The Complete Reference, McGraw Hill Professional.

[13] M. R. Ralph Kimball, The Data Warehouse Toolkit, Wiley.

[14] D. H. Chen, "OLAP Cubes in the SCSM Data Warehouse: OLAP Cube Processing," 3 February 2012. [Online]. Available: http://blogs.technet.com/b/servicemanager/archive/2012/02/03/olap-cubes-in-the-scsm-data-warehouse-olap-cube-processing.aspx. [Accessed 2013].

[15] "Mirth Connect," [Online]. Available: http://www.mirthcorp.com/products/mirth-connect. [Accessed 2013].

[16] International Institute of Business Analysis, A Guide to the Business Analysis Body of Knowledge, 2009.

[17] W. B. Lober, "Information System Architectures for Syndromic Surveillance".

[18] R. Domenig, "An Overview and Classification of Mediated Query Systems".

[19] J. G. Bellika, "Properties of a federated epidemiology query system," *International journal of medical informatics,* no. 76.

[20] Eclipse, "Eclipse Modeling Framework (EMF)," [Online]. Available: http://www.eclipse.org/modeling/emf/.

[21] "Teneo," [Online]. Available: http://wiki.eclipse.org/Teneo.

[22] Hibernate, "Hibernate," [Online]. Available: http://www.hibernate.org/.

[23] Oracle, "Oracle Data Integrator," [Online]. Available: http://www.oracle.com/technetwork/middleware/data-integrator/overview/index.html.

[24] Oracle, "Oracle Business Intelligence Enterprize Edition 11g," [Online]. Available: http://www.oracle.com/us/solutions/business-analytics/business-intelligence/enterprise-edition/overview/index.html.

[25] B. B.W., Software Risk Management, IEEE Computer Society Press, 1989.

[26] Oracle, "VirtualBox.org," Oracle, [Online]. Available: https://www.virtualbox.org/.

About the Authors

Athanasios Papakostopoulos received his Diploma in Computer Engineering and Informatics from the University of Patras, Greece in 2010. During his studies he focused on Software Technology and Computer Science. His main interests include parallel programming and algorithm analysis. His diploma thesis demonstrates how the concept of service oriented programming can be used to implement robotic algorithms which are agnostic to the underlying hardware.
In September 2011 he started working for Eindhoven University of Technology as a PDEng candidate for the Stan Ackermans Institute Software Technology Program. From January 2013 until September 2013 he worked at Philips Research on the project described in this report.