3TU.School for Technological Design

STAN ACKERMANS INSTITUTE

On-Site Customer Analytics and Reporting (OSCAR)

Panagiotis Thomaidis

September 2014

TU Delft
Delft University of Technology

TU/e Technische Universiteit Eindhoven University of Technology

UNIVERSITY OF TWENTE.

On-Site Customer Analytics and Reporting (OSCAR)

Panagiotis Thomaidis

September **2014**

On-Site Customer Analytics and Reporting (OSCAR)

A portable clinical data warehouse for the in-house linking of hospital and telehealth data

Panagiotis Thomaidis

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

Partners

Philips Research

Eindhoven University of Technology

Steering Group

Helen Schonenberg
Charalampos Xanthopoulakis
Mykola Pechenizkiy

Date September 2014

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	978-90-444-1316-8 A catalogue record is available from the Eindhoven University of Technology Library (Eindverslagen Stan Ackermans Instituut ; 2014/058)

Abstract

This document conveys the results of the On-Site Customer Analytics and Reporting (OSCAR) project. This nine-month project started on January 2014 and was conducted at Philips Research in the Chronic Disease Management group as part of the H2H Analytics Project.

Philips has access to telehealth data from their Philips Motiva telemonitoring and other services. Previous projects within Philips Research provided a data warehouse for Motiva data and a proof-of-concept (DACTyL) solution that demonstrated the linking of hospital and Motiva data and subsequent reporting. Severe limitations with the DACTyL solution resulted in the initiation of OSCAR. A very important one was the unwillingness of hospitals to share personal patient data outside their premises due to stringent privacy policies, while at the same time patient personal data is required in order to link the hospital data with the Motiva data. Equally important is the fact that DACTyL considered the use of only Motiva as a telehealth source and only a single input interface for the hospitals.

OSCAR was initiated to propose a suitable architecture and develop a prototype solution, in contrast to the proof-of-concept DACTyL, with the twofold aim to overcome the limitations of DACTyL in order to be deployed in a real-life hospital environment and to expand the scope to an extensible solution that can be used in the future for multiple telehealth services and multiple hospital environments.

In the course of the project, a software solution was designed and consequently deployed in the form of a virtual machine. The solution implements a data warehouse that links and hosts the collected hospital and telehealth data. Hospital data are collected with the use of a modular service oriented data collection component by exposing web services described in WSDL that accept configurable XML data messages. ETL processes propagate the data, link, and load it on the OSCAR data warehouse. Automated reporting is achieved using dashboards that provide insight into the data stored in the data warehouse. Furthermore, the linked data is available for export to Philips Research in de-identified format.

Keywords: telehealth, hospital information systems, Motiva, data warehouse, web services, virtualization

Preferred reference: P.Thomaidis, On-Site Customer Analytics and Reporting (OSCAR): A portable clinical data warehouse for the linking of hospital and telehealth data. Eindhoven University of Technology, SAI Technical Report, September, 2014. (978-90-444-1316-8)

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

Foreword

When it comes to research, it is commonplace that your project objectives are often vague or you begin your efforts based on – later proved – false assumptions. You are usually aware of the tools and technologies to employ, yet it is far from certain that your outcome shall live up to your expectations. Things look “easy” or “difficult” depending on the assumptions you make in the early beginning.

In the H2H business unit of Philips Research, we take pride in delivering innovation that matters to people’s lives, with a particular focus on Telehealth services and applications. Towards improving our inventions, we are constantly looking for the facts and figures that can allow us to evaluate the efficiency and efficacy of our remote patient health monitoring applications. In order to close the loop between patient hospitalizations and home tele-monitoring and investigate the Return On Investment for our services, we need to combine the patient clinical and telehealth data. It was fair to assume that such a linkage could easily take place at the comfort of an office at the High Tech Campus.

Hardly was that really the case. Stringent privacy and data protection laws expressly forbid the export of personal patient clinical data outside the hospital premises. This entailed that clinical and telehealth data analysis and linkage must take place on-site, notwithstanding the variability and complexity of Hospital IT Infrastructures. This rendered the deliverable of the DACTyL project non-deployable and gave rise to the OSCAR project in January 2014.

Panagiotis worked meticulously in order to derive a comprehensive software architecture that could fit into the variability of Hospital IT Infrastructures, perform the linkage in-house and generate the necessary reports and exports. Thinking out of the box, he proposed a design that welcomed changes; rather than enforcing an interconnection standard to Hospital IT, he suggested constructing the interfaces dynamically, based on the messaging formats favored by the Hospital administrators; instead of placing hardware equipment inside the Hospital server room, he implemented OSCAR, a configurable Virtual Machine that could fit in a USB stick and be directly deployed on the Hospital VMware infrastructure. His solution was ingeniously crafted, easily configurable, and seamlessly integrating into the Hospital IT infrastructures.

As a trainee for 9 months in Philips Research, Panagiotis lived up to my original expectations. A so-called “silent power”, he would be working quietly in his desk, drafting potential design alternatives on the whiteboard, asking the key questions, and proposing the appropriate solutions. As a graduate of the Electrical and Computer Engineering Faculty of the Aristotle University of Thessaloniki (AUTH), I was confident about his technical background and I appreciate the inquiring mind he developed in his OOTI program. I would only advise him to work on his presentation skills, as I can sense the stiff “engineer” attitude that AUTH graduates often have.

Upon writing these lines, we are considering the ways of taking his work even further, by aligning OSCAR with the Philips Healthcare Platform and performing benchmarking and qualitative analysis. His work would serve as a reliable basis to start from.

Charalampos Xanthopoulakis PDEng, Senior Software Designer, Philips Research
September 2014

Preface

This document is the technical report of the On-Site Customer Analytics and Reporting (OSCAR) project. The report describes the design and development of a portable, extensible analytical system capable of combining hospital and telehealth data in order to provide automated reporting. The project lasted for nine months (January - September 2014) and was conducted for and within Philips Research, in Eindhoven.

This document also constitutes the final thesis of the author, Panagiotis Thomaidis as part of the Stan Ackermans Institute Software Technology PDEng program of the Eindhoven University of Technology and should serve as proof of his skills as a software designer. The Professional Doctorate in Engineering (PDEng) program, also known by its Dutch name Ontwerpers Opleiding Technische Informatica (OOTI), spans two years of training and projects for various industry partners. OSCAR is the author's final project that concludes the PDEng program.

The work of the author included technical aspects such as collecting the requirements, creating the architecture, design, and implementation and project management aspects such as planning and documenting.

Non-technical readers should refer to Chapters 1, 2 and 3 that set the context and goals of the project and Chapter 11 where the results are summarized. Technical readers will be interested in Chapters 5-9 that describe the system requirements, architecture, design, implementation, and deployment respectively.

September 2014

Acknowledgements

A number of people contributed through the duration of this project. First and foremost I would like to thank Helen Schonenberg and Charalampos Xanthopoulakis my supervisors from Philips Research. On our weekly update meeting they provided guidance and support in the development effort of this project. I would also like to thank Mykola Pechenizkiy my TU/e supervisor for his valuable contribution and meaningful feedback.

All the OOTI coaches should be thanked for improving my technical and professional skills in this two year program. I would also like to thank my colleagues at OOTI who provided a friendly and constructive working atmosphere. Special thanks should go to Ad Aerts the General Director of the Software Technology program and Maggy De Wert, the OOTI secretary.

I would also like to thank the anonymous community of the internet who provided answers to technical questions, problem solving, tutorials, and useful domain knowledge.

Last but not least I would like to thank my family and loved ones for their support during the past two years and especially during the writing of this report.

Panagiotis Thomaidis
September 2014

Executive Summary

The On-Site Customer Analytics and Reporting (OSCAR) project lasted nine months and was conducted at Philips Research in the Chronic Disease Management group as part of the H2H Analytics Project. Its goal was to develop an extensible data warehousing framework that collects and links hospital and telehealth data and provides automated reporting and Return on Investment analysis of telehealth solutions.

Philips has access to telehealth data from their Motiva tele-monitoring and other services. Previous projects within Philips Research provided a data warehouse for Motiva data and a proof-of-concept solution, DACTyL, which demonstrated the linking of hospital and Motiva data and subsequent reporting. Severe limitations with the DACTyL solution resulted in the initiation of OSCAR. A very important one was the unwillingness of hospitals to share personal patient data outside their premises due to stringent privacy policies, while at the same time patient personal data is required in order to link the hospital data with the Motiva data. Equally important is the fact that DACTyL considered the use of only Motiva as a telehealth source and only a single input interface for the hospitals.

OSCAR was initiated to propose a suitable architecture and develop a prototype solution, in contrast to the proof-of-concept DACTyL, with the twofold aim to overcome the limitations of DACTyL in order to be deployed in a real-life hospital environment and to expand the scope to an extensible solution that can be used in the future for multiple telehealth services and multiple hospital environments.

In the course of the project, a software solution was designed and consequently deployed in the form of a virtual machine. The solution implements a data warehouse that links and hosts the collected hospital and telehealth data. Hospital data are collected with the use of a modular service oriented data collection component by exposing web services described in WSDL that accept configurable XML data messages. ETL processes propagate the data, link, and load it on the OSCAR data warehouse. Automated reporting is achieved using dashboards that provide insight into the data stored in the data warehouse. Furthermore, the linked data is available for export to Philips Research in de-identified format.

Extensibility of the solution is achieved first of all with the use of a modular data collection component that allows new web services to be implemented and placed to work side-by-side the existing ones. While it was considered that a generic non-programming solution was not feasible for the data collection component we kept the software design simple while automating the extension and customization process as much as possible. For the data warehouse and reporting components, the use of off-the-self tools allows for easy maintenance and extensions. More specifically, new facts and dimensions, new ETL processes, new OLAP cubes, and new dashboards can be added to the system, to support new reporting requirements without affecting the existing ones.

All in all, OSCAR provides Philips with a framework that can serve as the basis portable analytics platform for supporting reporting for its healthcare customers in the future.

Table of Contents

List of Figures

List of Tables

1.Introduction

This chapter introduces the On-Site Customer Analytics and Reporting (OSCAR) project and its context. More specifically, the organizational context and previous relevant projects are presented. An outline of the rest of this report is also provided.

1.1 Context

Philips Research is part of the Philips Corporation. Its function is to provide technology options for innovations in the areas of Healthcare, Consumer Lifestyle, and Lighting. Philips Research works on everything from spotting trends and ideation to proof of concept and, where needed, first-of-a-kind product development [1]. Within Philips Research, OSCAR is part of the H2H Analytics Project, which is part of the Chronic Disease Management group, as shown in Figure 1.

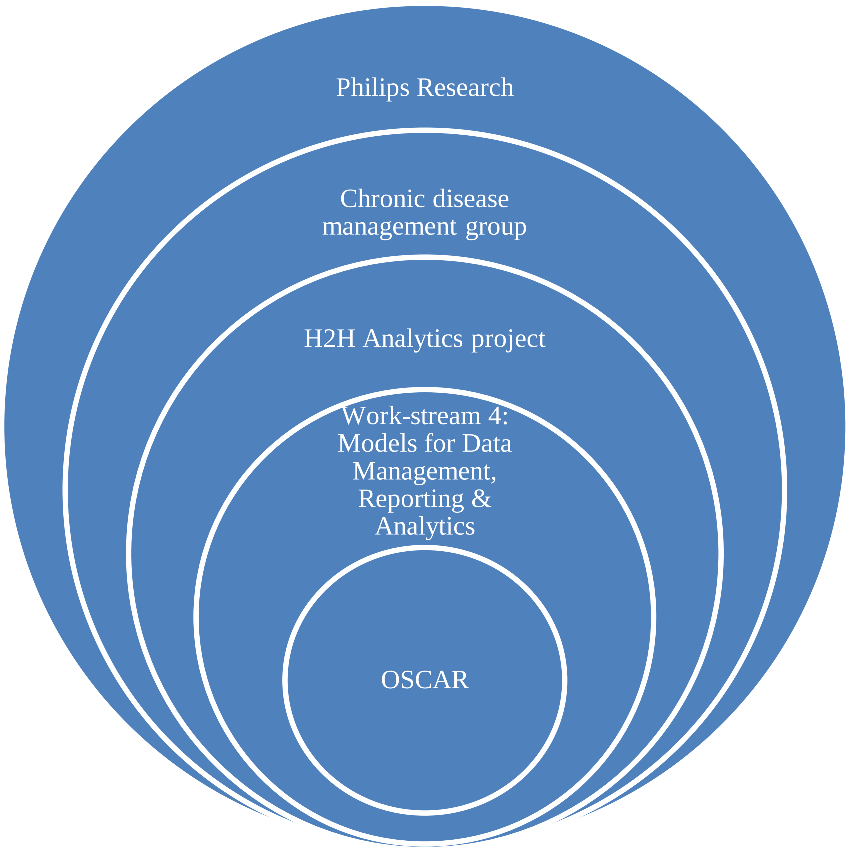

Philips Research
Chronic disease management group
H2H Analytics project
Work-stream 4: Models for Data Management, Reporting & Analytics
OSCAR

Figure 1 - OSCAR organizational context

1.2 Motivation

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 provide high-quality care for an increasing number of patients more efficiently using limited financial and human resources [2].

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, the caregiver is geographically separated from the care consumer and the treatment is 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 [2]. In the area of telehealth, Philips offers Motiva, a system that supports chronic disease management, including telehealth, education and coaching. The Motiva system is managed by the Hospital to Home (H2H) business sector of Philips Healthcare.

Philips is interested in supporting customer (e.g. hospitals) reporting needs and obtaining healthcare data for research purposes. More specific to the aim of OSCAR, there is a demand for linked hospital and telehealth data, so as to investigate the tangible benefits of Philips telehealth services, a particular use case being the reporting on the Return on Investment of the Motiva service.

1.3 Motiva

Motiva serves as a telehealth data source for OSCAR. For this reason, it is briefly described here, to provide the reader with an overview of the Motiva system in particular, but also telehealth systems in general.

Using measurement devices, such as weighing scales, blood pressure monitors and glucose meters enhanced with the Bluetooth technology, the vital signs of a patient are transmitted to a set-top box. That device acts as a medical gateway; see Figure 2, 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 any unnecessary 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 sector of Philips Healthcare.

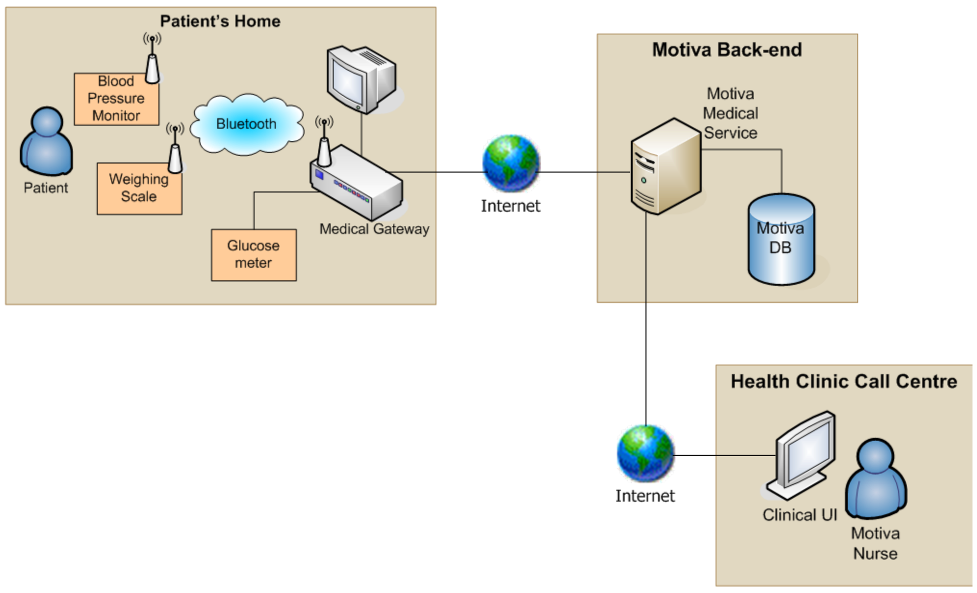


Figure 2 - Motiva Architecture

Figure 2 shows the three main components that comprise Motiva:

- The measuring devices and medical gateway in the patient's home
- The back-end which hosts a database that maintains all Motiva data
- A health clinic call center where the Motiva nurse is located

The operation of Motiva consists of the following steps:

1. The patient takes his vital-sign measurements using the corresponding electronic devices
2. The measurements are transferred to the medical gateway via Bluetooth
3. The medical gateway sends the data to the Motiva DB for storage and processing
4. The Motiva nurse views the patient data, vital signs, lab results, medications on a Clinical UI and intervenes if necessary
5. The patient views measurements, messages, videos, and surveys on his television

Motiva serves as the specific telehealth product whose data was used to demonstrate OSCAR. The interest for OSCAR focuses in the Motiva DB, see Figure 2, and the data that it contains. OSCAR is thus designed under the principle that it should be compatible with Philips telehealth solutions that share the same general architecture as Motiva.

1.4 Project history

The developments within Philips that led to OSCAR began with the Motiva system, more particularly, the data collected in the Motiva DB and the need for an automated reporting tool for it. Thus, the Motiva Data Warehouse (Motiva DWH) project was conducted in which a data warehouse for Motiva data was developed to provide automated, scalable reporting on Motiva Key Performance Indicators (KPIs).

The need for linked telehealth and hospital data still remained. Therefore, the DACTyL project was initiated. It delivered a proof-of-concept solution to answer the question of whether the Motiva DWH system can be extended to link hospital and telehealth data to provide combined reporting and how a system could be implemented to gather the necessary hospital data.

OSCAR was initiated to continue were DACTyL left off in order to move from a proof-of-concept solution to a functional prototype. Figure 3 depicts the sequence of the projects that led to OSCAR. The projects are described in more detail in the following sections.

Motiva Data Warehouse
DACTyL
OSCAR

Figure 3 – The projects that led to OSCAR.

1.4.1. Motiva Data Warehouse (Motiva DWH)

The Motiva DWH project was an attempt for the creation of a telehealth data warehouse. Its aim was to replicate the existing Motiva reporting in a data warehouse, moving from labor intense, un-scalable, manual reporting to a scalable, automated reporting system.

The outcome of this project is a data warehousing solution that consists of:

- An analytical database, with a data model in the form of a star schema, on top of which various dashboards and reports can be built to investigate the Key Performance Indicators (KPI's) of Motiva

- A set of ETL processes that load the data from the Motiva DB to the analytical database
- A set of reports that visualize the Motiva data, provided to demonstrate the applications of the underlying data warehouse.

To give an impression of the reporting capabilities the data model contains the following fact tables:

- Task
- Suspension
- Vital measurement

And the following dimension tables:

- Location
- Clinician
- Patient
- Task
- Closing reason
- Suspension reason
- Measurement type

Readers unfamiliar with data warehousing and star schema concepts may refer to Chapter 4.

The database was implemented using the Oracle Database (ODB) tool, the ETL was implemented using the Oracle Data Integrator (ODI) tool, and the reporting using the Oracle Business Intelligence (OBIEE) tool.

In conclusion the Motiva DWH project showed that a data warehouse is suitable for automated reporting and provided a specific implementation customized on Motiva data.

1.4.2. Data Analytics, Clinical, Telehealth, Link (DACTyL)

DACTyL was a proof-of-concept project launched to investigate:

- How to obtain hospital (clinical) data
- How to link hospital and telehealth data
- How to report on the linked dataset

The DACTyL solution was built on top of the Motiva DWH by expanding the data warehouse schema to include hospital related tables (to record hospitalization events with respect to the patient, admission, discharge, mortality dates, and diagnosis codes) and by adding support for methods to accept hospital data and load them into the data warehouse.

The DACTyL implementation had the following characteristics:

- Oracle tools for the data warehousing and reporting (the same as the Motiva DWH project)
- A Java server hosted on the Apache server to accept hospital data in the form of XML messages
- Reporting on hospitalization events, in addition to those of the Motiva DWH
- Implementation on a physical machine (laptop)

There were two possible deployment options for the DACTyL system, namely deployment within Philips or within the hospital. The two options are depicted in Figure 4 and Figure 5.

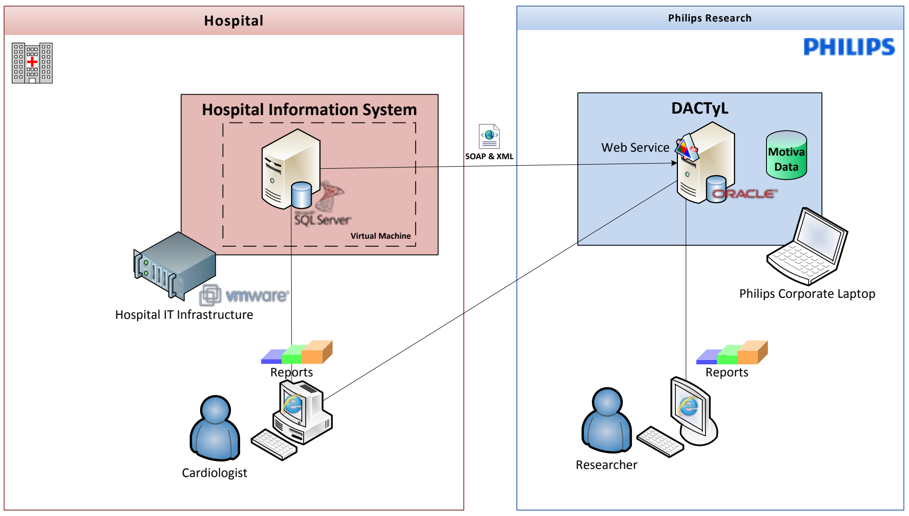


Figure 4 - DACTyL off-site deployment

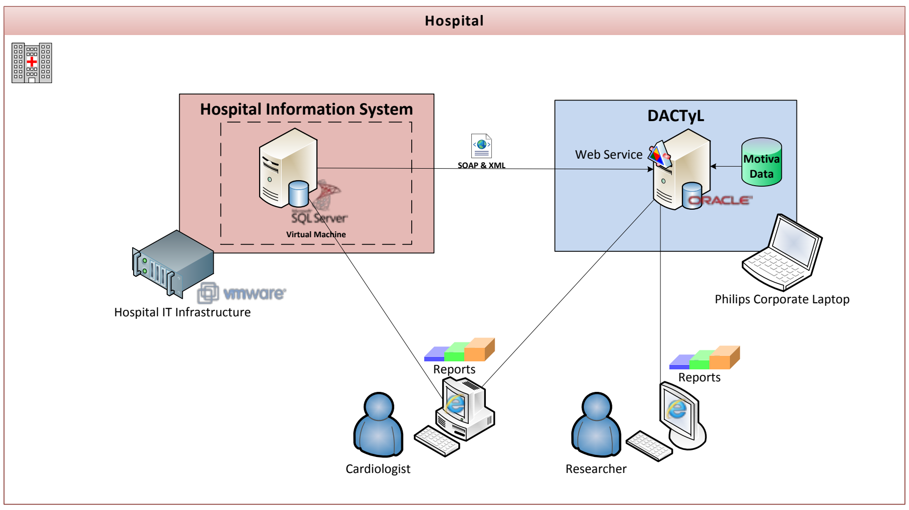


Figure 5 - DACTyL on-site deployment

In Figure 4, the St. Anna hospital is depicted on the left and Philips Research on the right. Within the hospital lies the hospital information system that contains, among others, the data stores and communication sub-systems that DACTyL interacts with. The DACTyL system is hosted within Philips Research. This deployment assumes a communication channel between the hospital information system and DACTyL via the internet, so that DACTyL can collect the necessary hospital data in a push-fashion model. Telehealth data would already be loaded on the DACTyL data warehouse. After the data is collected and linked, reports are provided to the cardiologists and researchers. Nevertheless, this approach has a very important limitation. That is, the unwillingness of hospitals to share personal patient data outside their premises due to stringent privacy policies, while at the same time personal patient data is required in order to link the hospital data with the Motiva data.

The limitations with the off-site deployment led to the on-site deployment, Figure 5, where the laptop that hosts DACTyL has been moved to the hospital. This deployment option met the unwillingness of the hospital IT department to accept the deployment of foreign hardware on their premises, due to security concerns.

Despite its limitations, the results of the DACTyL project were demonstrated in a hospital (St. Anna Ziekenhuis, Geldrop) on their request, where they acknowledged the added value of linked reporting offered by DACTyl. This, in combination with the limitations of DACTyL was the start of the current project. The limitations of DACTyL are described in Chapter 3 where the problem analysis of OSCAR is performed.

1.5 OSCAR Scope and Goals

In previous work within Philips, the Motiva DWH project demonstrated the application of data warehousing and automated reporting on Motiva data (telehealth). DACTyL went one step further and combined hospital and Motiva data. OSCAR is based on the proof-of-concept DACTyL and was initiated to build upon it, to make it scalable and suitable for deployment at hospitals. To build upon the DACTyL system, OSCAR aims to address those requirements that are needed to deploy it in a real-life context, such as interoperability and extensibility. In addition the DACTyL system should be re-designed and re-implemented using more appropriate technologies.

OSCAR aims to create a hospital and telehealth analytics framework. Specific implementation details of this framework include the interfaces used for the data collection, the schemas of the databases in the system, and the design of the exposed reports-dashboards; these aspects define the business processes or use cases that OSCAR supports. While DACTyL supported one business process, namely patient hospitalizations, OSCAR aims to develop a system that is extensible to multiple business processes. For example OSCAR may initially support reporting on patient hospitalizations intended to provide an overview to the hospital clinicians. Later, when a new reporting need is identified, for example vital sign measurement reports, OSCAR will need to be extended to accept the new data, store it, and provide the corresponding reports.

OSCAR enables Philips to begin cooperation with the hospital stakeholders in order to define specific input data interfaces to support and business processes to report on. This process can only take place in an iterative, agile, fashion in collaboration with the relevant stakeholders, the hospital clinicians, IT department and data owners. This includes the issue of gathering requirements for specific end-user reports. We consider this the topic of future work and thus outside of our scope.

All in all, OSCAR is expected to provide value to Philips by:

- Addressing customer reporting needs, for example to provide new insights based on the complete patient's story as he moves between home monitoring and hospitalization
- Enabling and providing experience on the cooperation with hospitals and other healthcare business partners, for example from a legal perspective
- Providing a source for linked hospital and telehealth data for research purposes, which can be used for the development of new models and algorithms

The OSCAR project covers the following:

- Design and implementation of the functionality as mentioned in the system functional and non-functional requirements
- Investigation of relevant technical and healthcare standards
- Investigation of alternative technologies
- Creation of examples of reports (interactive dashboards) to demonstrate its functionality

1.6 Report outline

Chapter 2 introduces the OSCAR stakeholders and explains their interest and influence in the project.

Chapter 3 describes the problem of collecting and linking healthcare data and providing reporting in the context of this project

Chapter 4 provides the reader with the necessary domain knowledge, including clinical and data warehousing concepts, to understand the later chapters.

Chapter 5 documents the functional and non-functional system requirements. It also explains the user roles that are associated with OSCAR and lists important constraints.

Chapters 6, 7, and 8 describe the proposed solution in terms of its architecture, design, and implementation. The design decisions are documented and there are links back to the system requirements.

Chapter 9 describes issues related to the deployment of the OSCAR solution.

Chapter 10 documents how the system is tested and offers a mapping between the implemented features and the system requirements.

Chapter 11 documents the results of OSCAR and outlines relevant future work.

Chapter 12 documents the project management activities.

Chapter 13 is a reflection of the author over the whole project and its process.
■

2.Stakeholder Analysis

This chapter provides a short overview of the parties that are stakeholders for the OSCAR project.

2.1 *Philips Research*

Philips Research and more specifically the H2H Analytics project is interested in the OSCAR software solution, the experience gained from the cooperation with the hospitals, and the linked data that is collected. A more indirect stakeholder is the Philips H2H business unit of Philips Healthcare which is planning future healthcare projects (e.g. the Digital Healthcare Platform, a platform to create, deploy and manage health applications) with which OSCAR must try to align.

Specific stakeholders from Philips are the following:

- Helen Schonenberg - R&D Scientist (Company Supervisor)
- Charalampos Xanthopoulakis PDEng - Senior Software Designer (Company Supervisor)
- Steffen Pauws - Senior Scientist (Project Manager of the H2H Analytics project)

Helen Schonenberg provided guidance concerning the form of the solution and provided most of the requirements. She was also the connection with the previous projects (Motiva DWH and DACTyL). The role of Charalampos Xanthopoulakis was that of a technical advisor. He provided assistance with decisions concerning the architecture, design and implementation and evaluated the proposed solutions. Steffen Pauws communicated the longer term strategy from a business perspective.

Another Philips stakeholder is the Philips IT department which imposed specific requirements on the used technologies.

2.2 *Eindhoven University of Technology (TU/e)*

The Eindhoven University of Technology is responsible for the educational aspect of this project.

The representative stakeholder from the TU/e is Mykola Pechenizkiy - Assistant Professor, who has the role of university supervisor. The university supervisor should make sure that the design and documentation meet the standards of a PDEng project. Meetings with the university supervisor took place every month in order to share and discuss the state of the project. He also had the role of helping with the academic aspects of the project.

Another TU/e stakeholder is Ad Aerts, the Program Director of the Software Technology PDEng program. He is a stakeholder in the sense that he must ensure that the project meets the quality requirements of the PDEng program.

2.3 *St. Anna hospital in Geldrop*

The St. Anna hospital in Geldrop will be the organization where the OSCAR prototype is deployed. More specific stakeholders from the hospital are

- The **clinicians** at the hospital who will be the users of the reporting capabilities of OSCAR
- The **data owner** who is responsible for maintaining the hospital data
- The **hospital IT department** who will be responsible for deploying and integrating with OSCAR

- The **legal officer** who is responsible for dealing with legal issues, such as obtaining patient consents for using their data within the project

Discussions with the St. Anna hospital took place before the beginning of this project, when the results of the DACTyL project were demonstrated to them. The discussion and demonstration served to get the hospital stakeholders interested in the project, to establish some reporting requirements, and to discuss deployment options. The outcome of those meetings heavily influenced the envisioned OSCAR solution mostly by making apparent some constraints on the deployed system (privacy issues in sharing patient personal data outside the hospital, security concerns when deploying foreign computers to the hospital). The constraints are discussed in the problem analysis, Chapter 3, and the system requirements analysis, Chapter 5.
■

3.Problem Analysis

This chapter describes the problem that OSCAR attempts to solve. The problem description begins from the existing DACTyL solution.

3.1 DACTyL constraints

DACTyL was overall a successful project but it was bounded by certain limitations which became apparent in follow-up meetings with the hospital stakeholders. The results are summarized below.

- Results of DACTyL
  - Working proof of concept
  - Displayed end-to-end functionality (from XML input to hospitalization reports)
  - Displayed data linking and reports based on the source data sets
- Constraints imposed by the hospital
  - Regarding the off-site deployment of DACTyL, hospitals may share only de-identified patient data outside their context due to strict privacy policies. Thus rendering data linking outside the hospital impossible since it would require the hospital to share the personal details of their patients
  - Regarding the on-site deployment, the hospital IT is skeptical about hosting external computers (DACTyL laptop) within their premises
- Constraints provided by Philips IT
  - The Philips IT department strongly favored the use of Microsoft tooling and discouraged the use of the technologies employed in DACTyL

As can be seen from the list of constraints, DACTyL could not be immediately deployed to the hospital due to the aforementioned constraints. Also, the scope of DACTyL was too narrow, as it only considered Motiva as its telehealth data source and only hospitalizations events as the business process for which it collected data and reported on. Thus, the interfaces with the hospital were custom made and not extensible enough, as is the case with WSDL-based web services.

3.2 OSCAR

The aim of OSCAR was to expand on the solution proposed by the DACTyL project, re-designing and re-considering the design decisions taken, in order to overcome any weaknesses and make a realistic prototype. In addition, OSCAR was required to widen the scope in order to take into account the scalability of the system to multiple telehealth systems, multiple hospital business processes, and multiple sites (hospitals).

OSCAR Problem statement: *How can we provide on-site reporting on identified, linked clinical and telehealth data for the clinician, combined with de-identified data extracts for external researchers in a scalable, configurable way?*

During our initial meetings the stakeholders from Philips Research described the problem by using an envisioned solution, a proposed system with the desired characteristics in order to communicate their view on the system requirements. This served as an initial input to us, in order to later define and refine the system requirements, which are described in Chapter 5.

In the envisioned solution the DACTyL laptop has been replaced by a virtual machine as the container that hosts the software system. The virtual machine should be deployed within the hospital virtual machine infrastructure. The reason for choosing a virtualized solution is twofold.

- It overcomes the constraint that the hospitals are not willing to accept foreign physical machines in their premises
- More and more hospitals turn to virtualized solutions to set-up their infrastructure; therefore, it will be easy to deploy such a solution.

The envisioned architecture for OSCAR is shown on Figure 6.

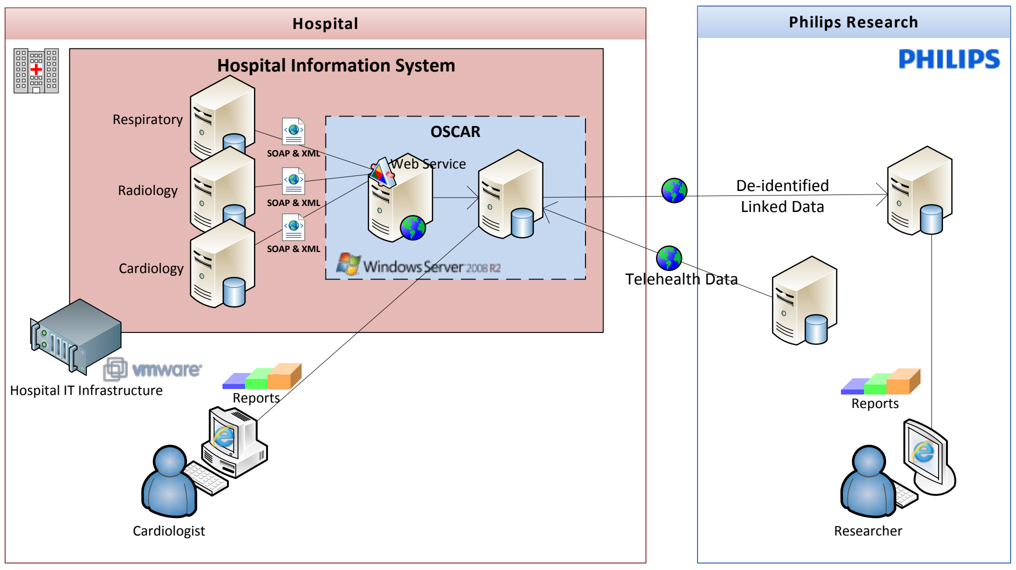


Figure 6 - OSCAR Envisioned Architecture

Figure 6 shows the separate hospital and Philips IT infrastructures. OSCAR is deployed within the hospital. It gathers the hospital data from the various hospital data stores and the telehealth data are uploaded from Philips. It presents the linked data to the Cardiologist in the form of reports and also shares de-identified data with Philips to be used by the researchers. The reader should note that the envisioned solution is generic in the sense that it does not consider specific reporting use cases. This genericity was expected to be an important aspect of the final OSCAR solution.

The following steps explain the functionality of OSCAR:

1. Philips and a healthcare institution agree on some information that clinicians wish to view on reports, for example that the clinicians want to receive an overview of their patients' hospitalizations along with their Motiva vital sign measurements.
2. If OSCAR supports the desired interfacing and reporting functionality it can be used as is. Otherwise the OSCAR maintainer configures an OSCAR instance based on the agreed reporting requirements. OSCAR should be configured to accept the necessary input data and generate the required reports-dashboards.
3. OSCAR is given to the hospital IT administrator, for example in a storage medium such as a USB stick or an external hard disk.
4. The hospital IT administrator installs OSCAR on the hospital virtual infrastructure. In addition the hospital IT administrator must configure the network access to the virtual machine.
5. The hospital IT administrator connects the hospital data sources to the OSCAR web interfaces. This allows the hospital data to be provided to OSCAR.
6. Messages from the hospital, containing for example hospitalization data, are sent to OSCAR via the interfaces.

7. Motiva data is imported to OSCAR by the OSCAR maintainer after the hospital IT administrator has given him/her access rights to the system. The data are digested by OSCAR, to extract for example Motiva vital sign measurements and other information.
8. OSCAR generates and exposes reports based on the linked hospital and Motiva data. These reports contain personalized patient data and are intended for the hospital clinicians.
9. The clinicians log-in on the OSCAR dashboards and view the reports on their own PC.
10. Philips researchers get data exports of the de-identified linked data. The extraction of the data exports must be performed in collaboration with the IT department of the healthcare institution, for example a date and time can be agreed for Philips to connect to the deployed OSCAR instance via a secure remote connection to download the extracted data files.

The steps that were outlined already allow us to shape a view of the functionality of OSCAR. We believe that the main challenge of this project lies in supporting the process described in step 2, namely the configuration of OSCAR to support new use cases.

3.3 Design opportunities

We identified the following design criteria, defined in [3], as relevant for this project. These criteria can be used to assess the quality of the proposed solution.

Ease of use: This concerns the ease of use for the stakeholders. The relevant stakeholders are the hospital IT department, the clinicians, and the Philips parties who will be in charge of maintaining, configuring, and extending OSCAR.

Genericity (Extensibility, Reusability): The extent to which OSCAR can be used in multiple hospitals for multiple reporting needs. Every hospital and every reporting requirement introduces new data collection, data storage, and data presentation needs. In addition, the different input data models introduce interoperability issues that need to be dealt with before systems can share their data.

Technical feasibility: This concerns the certainty that it is technically possible to produce OSCAR and deploy it in a real-life context. In other words the project should not just be a theoretical study; it should comprise the first steps towards the development of a production application.

■

4. Domain Analysis

This chapter contains the analysis of the domains that are relevant to OSCAR.

4.1 Introduction

The OSCAR project lies on the intersection of three major domains, namely the web services, healthcare, and data warehousing domains. Moreover, it involves aspects of integration (between different technologies) and interoperability (of data between different sources). In this chapter we attempt to provide the reader with the necessary knowledge on these domains in order to better understand the architecture, design, and implementation chapters, Chapters 6, 7, and 8.

4.2 The healthcare data domain (healthcare analytics)

Healthcare is one of the domains that are being transformed by the introduction of data analytics. Every aspect of the healthcare process creates a vast amount of data that can be captured and analyzed in order to improve the level of the services provided. OSCAR is concerned with data from hospitals and telehealth, such as Motiva, described in Chapter 1.

4.2.1. Hospital Information Systems (HIS)

A Hospital Information System (HIS) is a comprehensive, integrated information system designed to manage all the aspects of a hospital operation, such as medical, administrative, financial, and legal and the corresponding service processing. Hospitals are extremely complex institutions with large departments and units that coordinate care for patients. Hospitals are becoming more reliant on the ability of the HIS to assist in the diagnosis, management, and education for better and improved services and practices. In health organization such as hospitals, implementation of HISs is inevitable due to many mediating and dominating factors such as organization, people and technology. As soon as OSCAR is deployed on the hospital it will essentially be a part of the HIS. It should be able to interface with the rest of the HIS systems in order to collect the clinical data and provide reports.

An important part of the HIS is the Electronic Medical Record (EMR) where patient details are stored. The data in the EMR is the legal record of what happened to the patient at the hospital. EMRs are often part of a local stand-alone health information system that allows storage, retrieval and modification of records. 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. The EMR system plays an important role for this project, because it is a source location of the clinical data OSCAR aims to collect.

Another important domain term is the Electronic Health Record (EHR). The term describes a record in digital format that is capable of being shared across different care delivery organizations. The EHR is a shareable version of the EMR. 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 may occur by means of network-connected enterprise-wide information systems and other information networks or exchanges. EHRs can form the basis of the messages that are sent from the HIS to OSCAR.

In order to connect the various subsystems of the HIS together, a utility system is used, usually called communication broker or interface engine. The interface engine 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 party for all messaging between hospital systems, as illustrated in Figure 7. The interface engine is an important system for this project because it is able to create a communication path between the hospital systems and OSCAR.

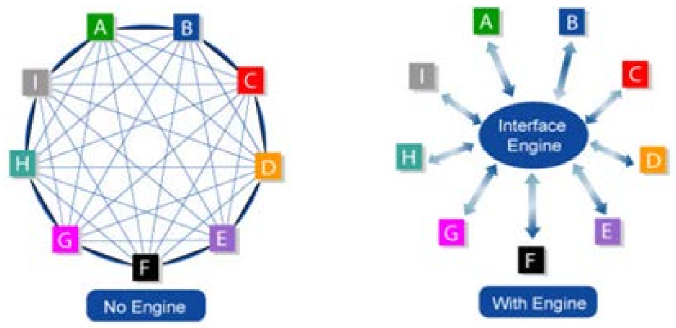


Figure 7 - Hospital Interface Engine

4.2.2. Healthcare Standards

Healthcare standards are standards that are designed to allow healthcare-related organizations and systems to meaningfully exchange information by relying on a common set of concepts and a common vocabulary. Healthcare standards are a way to simplify the interfacing process, since all systems will be able to speak a “common language” as defined by the standard. In this respect, they are relevant for the current and future versions of OSCAR, and, in general, any healthcare system that aims to exchanging meaningful messages.

Health Level 7 (HL7)

The task of standardization in healthcare is done by various organizations. The leading position belongs to Health Level 7 (HL7). HL7 is a U.S.-based, ANSI-accredited health information standards development organization. Its specifications are mostly for application-level messaging among hospital information systems (HISs). 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.

The third version of HL7 aimed primarily to defining application messages, but now using a well-defined information model, the Reference Information Model (RIM). Its main classes are shown in Figure 8. The core classes of RIM are shown in blue and the first level of subclasses is shown in white. The RIM aims to form the foundation for all information modeling within HL7. Since the RIM aims to be a reference model that encompasses the entire healthcare domain, its knowledge can be useful in understanding any healthcare application.

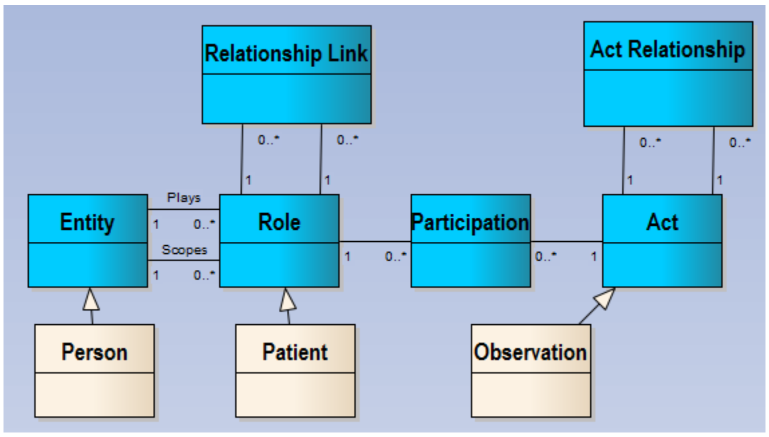


Figure 8 - Main classes of the HL7 Reference Information Model

The classes in the RIM model are briefly described below [4]:

Act: Every happening is an Act
Examples: Procedures, observations, medications, supply, registration

ActRelationship: Relate Acts with one another
Examples: composition, preconditions, revisions, support

„
Participation: Participation defines the context for an Act
Examples: author, performer, subject, location

„
Roles: The participants in an Act are Roles
Examples: patient, provider, practitioner, specimen, employee

„
Entities: Roles are played by Entities
Examples: persons, organizations, material, places, devices

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. Its semantics are derived from the HL7 RIM and it 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. The CDA specification is intentionally abstract so that the implementors can define their own flavor of the CDA document 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.

The CDA is important for this project as it is the state of the art standard designed to provide semantic interoperability, applicable in the data collection aspect of OSCAR. 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 and in also the form of XML Schema specifications. The latter is the form that we are mostly interested in since the XML Schema document can help in the implementation of the OSCAR data collection interfacing components. As already mentioned, OSCAR should try to make use of, or at least be compatible with, possible CDA documents the hospital is using for exchanging patient information with third parties.

Clinical Vocabularies

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

A very useful vocabulary is the 10th revision of International Classification of Diseases (ICD-10). ICD was originally published by the World Health Organization for classifying and coding of mortality cases. 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 1. ICD-10 is relevant for this project as it is used to relate a hospitalization event with particular diseases by including the relevant ICD-10 codes in the respective hospital messages.

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

Table 1 - Example of ICD-10 Hierarchy

4.3 Business Intelligence (BI) and Data Warehousing

OSCAR can be characterized as a data warehousing solution. The purpose of a data warehouse is to provide the technical infrastructure to support BI applications. In simpler terms data warehouses are tools that can help people understand their business better, based on data.

BI is the process of transforming data to useful information for decision makers. BI applications are usually associated with a specific data warehouse that maintains the necessary data in the appropriate form. Figure 9 shows a generic architecture for a data warehouse that supports BI.

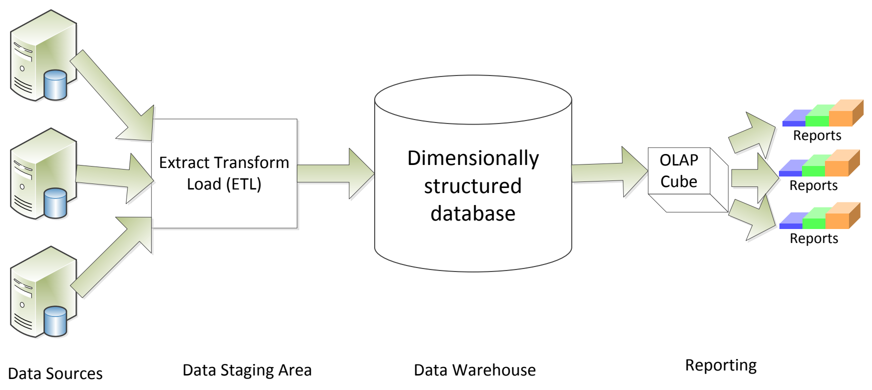


Figure 9 - Data warehouse generic architecture

Data sources (Operational Systems)

They are the traditional database systems that store transactional data of the organization's business. Operational systems in hospitals are part of the HIS and store the daily transactions with patients; for example one such data store is the EMR that stores patient details, or a database that stores patient hospitalizations. Another data source for OSCAR, apart from the hospital ones, is the Motiva database, described in Section 1.3. These databases are generally used one record at a time in contrast to analytical databases which are used for reporting and where summaries and aggregates of multiple data records are usually preferred. The value of the data warehouse lies in collecting data from multiple data sources and presenting a unified view on the data they contain.

Data Staging Area

The data staging area is a storage area as well as a set of ETL processes that extract data from the source systems. It is everything between the source systems and the data warehouse. Data staging is not meant for reporting, but as a temporary location where data from the source systems is stored, cleansed, and transformed in order to be loaded into the data warehouse.

The storage of the staging area is not necessarily a database. It could also be flat files for example. It can be structured like the normalized source systems or be an unstructured data storage space. It fully depends on the design requirements and the needs of the development process.

Extract, Transform, Load (ETL)

ETL, as the name implies, is the process of extracting data from one location, transforming it, and loading it in another location, the locations usually being databases or files. During transformation, the data is converted to the desired schema of the output database or file, values are perhaps filtered and cleansed, and in general it is ensured that the data is in a suitable format.

ETL processes can be implemented either by using a suitable programming language, like Java, or by using dedicated tools like the Microsoft SQL Server Integration Services or the Oracle Data Integrator.

Data transformation is a necessary step in the data warehousing process, because the schemas of the input data stores and the schema of the data warehouse are often quite different. First of all, we would like to load the data warehouse with data that come from multiple data stores, thus a need for a unified data schema. Secondly, the data sources are operational systems (schema more suitable for day-to-day business operations) whereas the data warehouse is dimensional (schema more suitable for reporting).

Presentation Area

The presentation area is what is generally called a data warehouse and takes the form of one or more databases. It is the place where cleaned, transformed data is stored in a dimensionally structured database schema and made available for analysis purposes. In the presentation area, the disparate data that has been introduced to the system from the various sources are linked and presented as a unified model.

Linking refers to the process of combining data from multiple input data stores into a unified data warehouse schema. It is done based on an entity that is known to both the input system and the data warehouse, for example patient records. Linking then refers to the process that matches patient records in the input data with patient records in the data warehouse, updating the patient personal information when required. Thus, the patient records form a meeting point between the data from the multiple input sources. Common attributes for linking a specific patient record between different data stores are certain unique identifiers, such as the patient Social Security Number, of the combination of the first name, last name and date of birth.

Dimensionally structured database (star schema)
The data presentation area usually takes the form of a database with a star schema. The star schema consists of one or more fact tables referencing any number of dimension tables. Fact tables record business events and dimension tables set the context of those events, for example where and how the events took place and who was involved. A visual example of a star schema is shown in Figure 10.

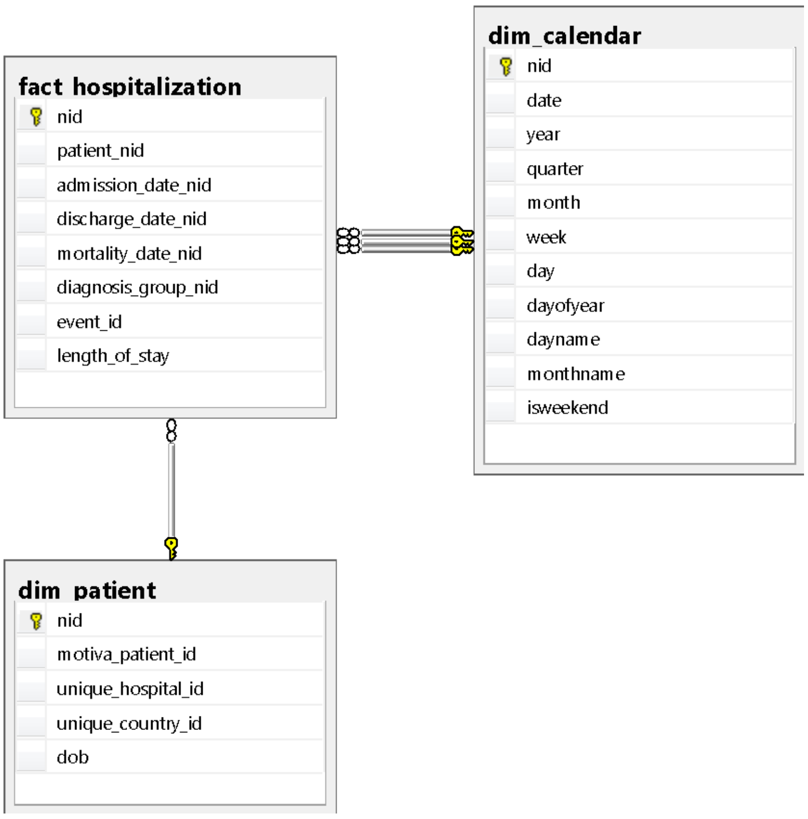


Figure 10 - Example of a simple star schema

In the example of Figure 10 the business event that is recorded in the fact table is patient hospitalizations. The two dimension tables set the context of the hospitalization in terms of the admission, discharge, and mortality dates (if applicable), and patient personal details.

Fact tables record measurements or metrics for a specific event. Fact tables generally consist of numeric values, and foreign keys to dimensional data where descriptive information is kept. Dimension tables usually have a relatively small number of records compared to fact tables, but each record may have a very large number of attributes to describe the fact data. Dimensions can define a wide variety of characteristics concerning when and where the event happened as well as who and what was involved.

Online Analytical Processing (OLAP) cube
The OLAP cube is a data structure that indexes the available data from the fact tables of the data warehouse. The dimensions of the cube correspond to the dimension tables in the data warehouse and form a coordinate system that can be used to select specific fact records based on constraints in the dimensions. Figure 11 shows an illustration 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 (the example is taken from [2]).

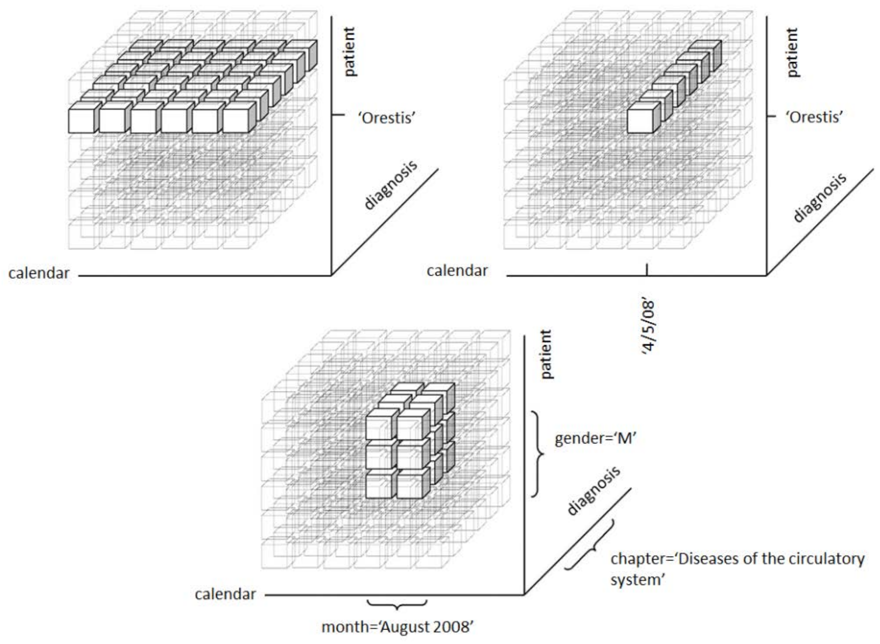


Figure 11- OLAP cube with 3 dimensions [2]

Every intersection of three dimension points is a fact, represented by a record in a fact table. Figure 11 could contain facts about patient hospitalizations. If we request data from the cube for patient "Orestis" with diagnosis code "I12.0" at calendar date "04-05-2008" the cube will return at most 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.

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 has "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=August 2008." The "month=August 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 setting "month=April 2008", specified the dimension point "year=2008," he would receive a more general result as an answer and this is termed "drilling up." Instead, if he specified "calendar week=15", he would get a more detailed result and this is termed "drilling down."

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 2 the measurement "hospitalizations" is summarized. For other use cases the average, minimum, maximum, of a measurement can be used.

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

Table 2 - Simplified report on hospitalizations [2]

Software vendors provide tools that simplify the creation of OLAP cubes out of a data warehouse, for example the Microsoft SQL Server Analysis Services [5] tool that we used for this project. The cube acts as an interface between the reports used by the user and the actual data warehouse. The OLAP cube requires upfront configuration and loading of the data warehouse data and then it provides fast access to that data for reporting. Based on the OLAP cube, software vendors sell tools that can be used to create automated reports and interactive dashboards. One such tool is the SharePoint dashboard designer.

Agile business intelligence
In order to design reports meaningful to the users an evolutionary approach is required, termed agile business intelligence. The approach consists of multiple iterations of designing example dashboards, getting user feedback, and refining them until the users are satisfied. During the process the underlying facts and dimensions of the data warehouse may also have to be refined.

4.4 Web Services

Web services can be used to fulfil the data collection needs of OSCAR, especially its interfacing with the hospital information systems. This section discusses some aspects of web services that are related to OSCAR.

Network communication protocols operate in layers, where each layer uses the functionality supported by the layers below, see Figure 12.

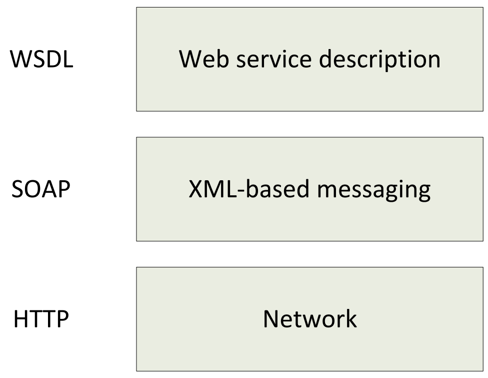


Figure 12 - The web services stack

HTTP
HTTP functions as a request-response protocol in the client-server computing model. Communication is initiated when the client submits an HTTP request message to the server. The server, then provides resources such as HTML files and other content, or performs other functions on behalf of the client, and returns a response message. HTTP resources are identified and located on the network Uniform Resource Locators (URLs).

SOAP
SOAP is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It relies on the XML Information Set for its message format, and relies on HTTP for the message transmission. SOAP forms the foundation layer of a web services protocol stack, providing a basic

messaging framework upon which web services can be built. It defines an envelope that can hold the structured XML messages that are communicated in a web service.

WSDL
The Web Services Description Language is an XML-based interface definition language that is used for describing the functionality offered by a web service. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. It thus serves a purpose that corresponds roughly to that of a method signature in a programming language. The WSDL file also contains the XML Schema definitions that describe the input and output messages that are sent and received. The WSDL file describes the interfaces between the HIS and OSCAR.

XML and XML Schema
The Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. Using XML messages is a way of exchanging structured data over the network.

The XML Schema (XSD) is a language that allows one to create a description of a type of an XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.

The interoperability aspect of web services
Interoperability is a key term for OSCAR. It describes the communication between OSCAR and the hospital information systems and is the aspect that defines whether meaningful data can be exchanged.

Syntactic interoperability
If two or more systems are capable of communicating and exchanging data, they are exhibiting syntactic interoperability. Specified data formats, communication protocols and the like are fundamental. XML or SQL standards are among the tools of syntactic interoperability. This is also true for lower-level data formats, such as ensuring alphabetical characters are stored in a same variation of ASCII or a Unicode format (for English or international text) in all the communicating systems.

Syntactical interoperability is a necessary condition for establishing semanting interoperability.

Semantic interoperability
Beyond the ability of two or more computer systems to exchange information, semantic interoperability is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of both systems. To achieve semantic interoperability, both sides must refer to a common information exchange reference model. The content of the information exchange requests is unambiguously defined: what is sent is the same as what is understood.

4.5 Virtualization

The term virtualization broadly describes the separation of a resource or request for a service from the underlying physical delivery of that service. With virtual memory, for example, computer software gains access to more memory than is physically installed, via the background swapping of data to disk storage. Similarly, virtualization techniques can be applied to other IT infrastructure layers - including networks, storage, laptop or server hardware, operating systems and applications. This blend of virtualization technologies - or virtual infrastructure - provides a layer of abstraction between computing, storage and networking hardware, and the applications running on it. The deployment of virtual infrastructure is non-disruptive, since the user experiences are largely unchanged. However, virtual infrastructure gives administrators the advantage of managing pooled resources across the enterprise, allowing IT managers to be more responsive to dynamic organizational needs and to better leverage infrastructure investments.

Virtualization is an important concept in OSCAR, since it is a technology increasingly being adopted by hospitals. By building OSCAR in the form of a virtual machine, it should be relatively easier to deploy it within the hospital VM infrastructure.
■

5.System Requirements

In this chapter the OSCAR system requirements are laid out. The requirements present a more analytical approach towards the definition of the problem and desired solution. In this sense they are based on the stakeholder analysis, Chapter 2, and problem analysis, Chapter 3.

5.1 Requirements overview

The goals of OSCAR can be summarized as follows:

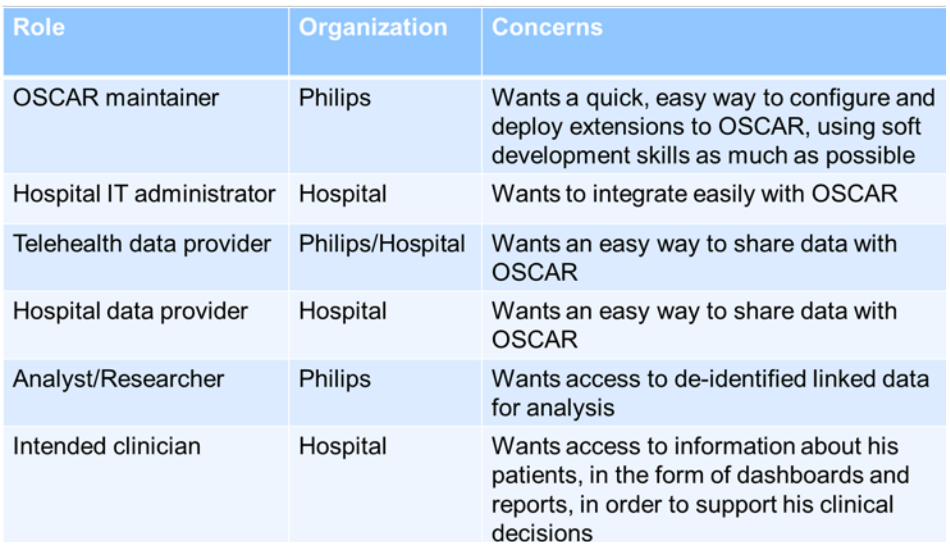
- To extend the DACTyL system in order to address those requirements that are needed to deploy it in a real-life context, such as
  - Interoperability
  - Extensibility
  - Portability
  - Any other aspect that is deemed important during the course of the project
- To re-design and re-implement the DACTyL system using more appropriate technologies

Derived from the above, OSCAR should address the following issues:

- **Interoperability:** To design a system able to retrieve data from hospital information systems by interconnecting using the relevant standards to allow as broad interfacing capabilities as possible
- **Extensibility:** To create tools and methodologies that will allow a user (maintainer) to add support for future interfacing or business intelligence requirements
- **Integrating with the hospital:** To design a system that
  - Supports BI dashboards and reports for clinicians/cardiologists
  - Makes it as easy as possible to integrate into the hospital IT infrastructure
- **User acceptance:** To design a system that is accepted and used by the stakeholders

5.2 User roles

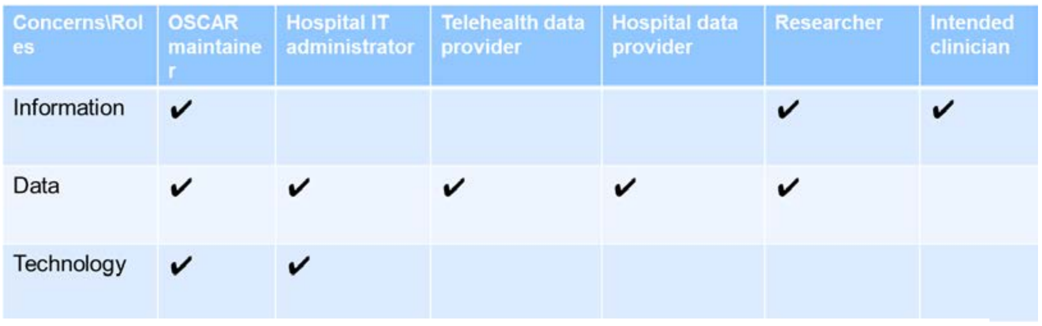
User roles are the behaviors that external actors are expected to play in relation to OSCAR. Based on the stakeholders described in Chapter 2 and the problem analysis in Chapter 3, we identify the following user roles, see Table 3.



Role	Organization	Concerns
OSCAR maintainer	Philips	Wants a quick, easy way to configure and deploy extensions to OSCAR, using soft development skills as much as possible
Hospital IT administrator	Hospital	Wants to integrate easily with OSCAR
Telehealth data provider	Philips/Hospital	Wants an easy way to share data with OSCAR
Hospital data provider	Hospital	Wants an easy way to share data with OSCAR
Analyst/Researcher	Philips	Wants access to de-identified linked data for analysis
Intended clinician	Hospital	Wants access to information about his patients, in the form of dashboards and reports, in order to support his clinical decisions

Table 3 - OSCAR user roles

To better understand each role, we divided them to three groups according to the aspect they are concerned with, see Table 4.



Concerns\Roles	OSCAR maintainer	Hospital IT administrator	Telehealth data provider	Hospital data provider	Researcher	Intended clinician
Information	✔				✔	✔
Data	✔	✔	✔	✔	✔	
Technology	✔	✔				

Table 4 - OSCAR aspects and role concerns

Technology: The roles associated with the technology aspect are interested in obtaining software licenses, setting-up, deploying, and integrating OSCAR in the hospital IT infrastructure.
Data: The roles associated with the data aspect are interested in integrating, linking, de-identifying, and gaining access to data.
Information: The roles associated with the information aspect are interested in the business intelligence and insights that the reports are providing.

5.3 *Use case scenarios*

We identify the following use case scenarios for OSCAR:

- Configure OSCAR
- Deploy OSCAR to the hospital
- Load telehealth data from Philips
- Export data for Philips
- View business intelligence reports

The scenarios and corresponding actors are shown in Figure 16.

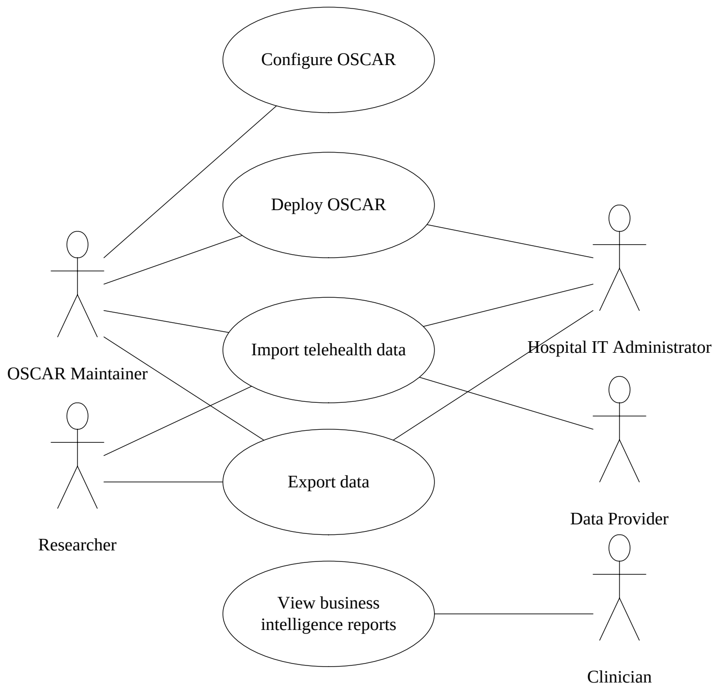


Figure 13 - OSCAR use case scenarios

The use case scenarios are decomposed in the specific steps below.

Configure OSCAR

1. Configure the data warehouse with the appropriate facts and dimensions
2. Configure the data collection component, including the landing area and input interfaces (this step requires the input of the hospital IT department since the interfaces must be mutually agreed upon)
3. Configure the ETL processes within OSCAR
4. Configure the reporting component including the dashboards (this step requires the input of the hospital clinicians since the reports-dashboards are intended for them)

Deploy OSCAR to the hospital

1. The OSCAR maintainer places OSCAR in a storage device
2. The OSCAR maintainer gives the storage device to the hospital IT administrator
3. The Hospital IT administrator installs and initializes OSCAR in the hospital IT infrastructure
4. The Hospital IT administrator performs all necessary configuration tasks to connect the hospital interface engine to the deployed OSCAR instance

Import telehealth data from Philips

Scenario 1:

1. The OSCAR maintainer visits the hospital premises with the telehealth data extract in a storage device
2. The hospital IT administrator provides access to OSCAR
3. The OSCAR maintainer loads the data files to OSCAR
4. The OSCAR maintained runs the appropriate ETL process to process the telehealth data files

Scenario 2:

1. The hospital IT administrator provides secure remote access to the deployed OSCAR instance
2. The OSCAR maintainer connects via the secure connection and loads the telehealth data extract files to OSCAR

Export data for Philips
Scenario 1:
1. The OSCAR maintainer visits the hospital premises
2. The hospital IT administrator provides access to OSCAR
3. The OSCAR maintainer copies the de-identified linked data files generated by OSCAR to a storage device and brings them back to Philips

Scenario 2:
1. The hospital IT administrator provides secure remote access to the deployed OSCAR instance
2. The OSCAR maintainer connects via the secure connection and copies the de-identified linked data files to Philips

View business intelligence reports
1. The clinician uses a web browser and navigates to the OSCAR business intelligence URL
2. The clinician provides valid user credentials to a login page
3. The clinician selects a report from a list and views the content

5.4 Functional requirements

The system requirements were identified based on meetings with the stakeholders from Philips Research. During these meeting the stakeholders described the existing and desired systems and introduced their envisioned architecture and use case scenarios. The functional requirements that we extracted are described in Table 5. The reader can note that the requirements roughly correspond to the various parts of a data warehouse.

Id	Name	Description
F1	Deployment	OSCAR should facilitate easy deployment in different hospital IT infrastructures
F2	Interoperability	OSCAR should be able to communicate with different hospital IT infrastructures, using different data sharing standards, as well as different kinds of schemas defined by the specific informational requirements. For example common such technologies are Http, SOAP, XML and HL7 CDA. It is part of the project to identify the most relevant ones and implement support for them. A common way to achieve interoperability is by describing the interfaces (Web Services) between systems based on WSDL and XSD, therefore the OSCAR should support these technologies
F3	Data import	The on-site deployment of OSCAR should be able to digest data from internal data sources (hospital data) and external data sources (Telehealth, Motiva).
F4	Linking	OSCAR should link the incoming data from the different sources, based on patient identification information, in a unified model (data warehouse model). This linking should occur within the hospital boundaries, because the identification information that is required for the linking cannot be taken outside the hospital premises.
F5	Storage	OSCAR should store all the incoming data, in other words no incoming data should be “thrown away”.
F6	Data export	The on-site deployment of OSCAR should provide methods to export

		linked, de-identified data to an external destination, in order to facilitate off-site analysis of the de-identified data by Philips research.
F7	On-site reporting	The OSCAR should offer detailed reporting on-site based on the linked, identified data. This reporting is to be used by the Intended Clinician for decision support.
F8	Supporting future use cases	OSCAR should be extensible in order to add support for new or additional informational needs of its users with minimal effort. OSCAR should be extensible with respect to different hospital IT infrastructures and different input sources within a hospital with minimal effort.

Table 5 - OSCAR functional requirements

5.5 Non-functional requirements

The nonfunctional requirements describe the qualities of the system. The non-functional requirements that we consider relevant for SOCAR are listed in Table 6.

Id	Name	Description
NF1	Extensibility/Adaptability	OSCAR should be open for extensions with new provided services to the HIS, for instance to support new interfacing technologies or protocols. Furthermore, it should be as easy as possible to implement those extensions
NF2	Privacy & Security	Only authorized users should be able to access and perform operations on OSCAR
NF3	Ease of use (user-friendliness)	It should be possible for the Analyst to get access to his data, for the clinician to view his reports, for the maintainer to extend, for the data provider to configure his system and start sending messages in as few user steps as possible
NF4	Compatibility	OSCAR should use technologies that are compatible with the software policies of Philips and the Hospital IT departments
NF5	Reliability (in terms of technology)	OSCAR should pass the necessary validation tests before being handed to the hospital IT for installation
NF6	Reliability (of the overall data flow)	The OSCAR should ensure the integrity of the data at each stage of processing
NF7	Efficiency	The OSCAR should keep system usage as low as possible (memory, processing, and disk space)
NF8	Safety	OSCAR should not interfere with and pose a risk towards the rest of the hospital operations, especially critical ones

Table 6 - OSCAR non-functional requirements

■

6.System Architecture

This chapter begins by approaching OSCAR as a black box, outlining the relation between OSCAR and its context, the hospital information system. It continues with the view of OSCAR in high-level components and a justification for the architectural design decisions. The architecture aims to fulfill, in the best way, the requirements as described in Chapter 5.

6.1 OSCAR as a black box

OSCAR is designed with the view to be deployed within the hospital IT infrastructure. This is the only viable solution given the constraints that are imposed due to privacy and security issues (for justification and earlier alternative deployment options refer to Chapters 1 and 3). The technical context of OSCAR is summarized in Figure 14.

The input of OSCAR is patient-related data from healthcare services (hospital and telehealth). Its outputs are

- Reports aimed at clinicians within the hospital and
- De-identified linked data extracts aimed at researchers

In both output cases the data from the various data sources must be linked. In contrast, clinician reports expose identified data, whereas data extracts for Research can only contain de-identified data.

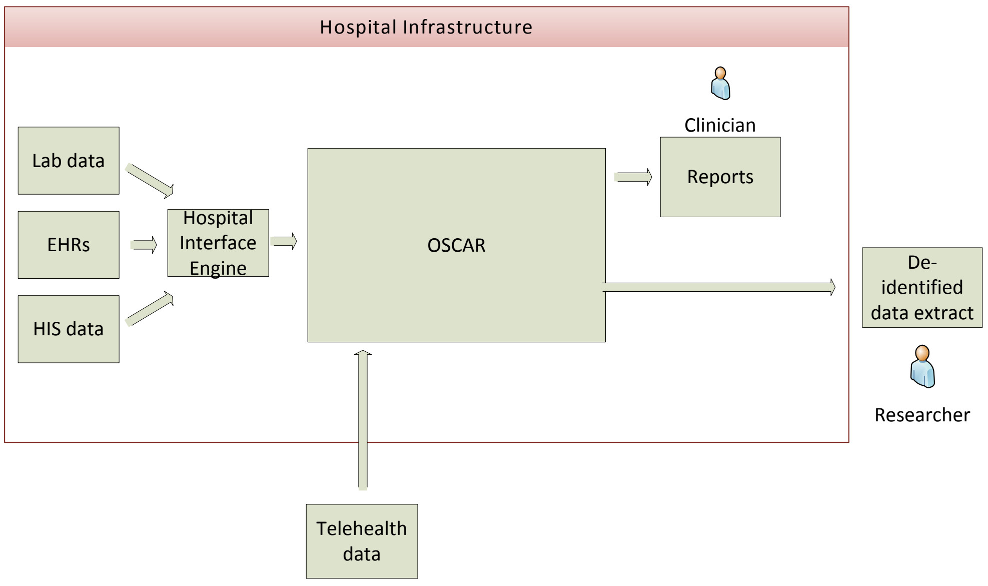


Figure 14 - OSCAR technical context

Figure 14 demonstrates, first of all, that OSCAR is deployed within the hospital infrastructure. Hospital input messages come from the Hospital Interface Engine that collects them from the various hospital data stores. Telehealth data are directly fed to OSCAR by the OSCAR maintainer. The two outputs, reports for clinicians within the hospital, and data extracts for researchers outside the hospital, are also shown.

Hospital data input

In general, the hospital data that OSCAR aim to collect exists in data stores scattered throughout the hospital; in Figure 14, the lab data, EHRs, and HIS data are examples of hospital data stores. In order to access that data, OSCAR either has to communicate directly with the data stores or wait passively and let a hospital application send the data to it, this application is the Hospital Interface Engine, described in Chapter 4. The former approach is pull-based and the latter push-based. OSCAR uses the push-based communication approach, where it is passively awaiting for messages and the hospital interface engine initiates all communication. The push-based approach was selected for two reasons:

- It avoids the problem of having to integrate with the multitude of hospital data stores by using a single point of contact (the interface engine)
- It limits the processing burden imposed on the hospital information system by allowing the hospital system to decide when the communication happens (preferably when it will not interfere with critical hospital applications) which is one of the non-functional requirements of OSCAR (NF8)
- Clinical data are generated in the course of time and so the HIS may decide on the most suitable moment in time to share the data with OSCAR

When OSCAR is deployed in a hospital the hospital IT administrator must configure the hospital interface engine in order to connect with OSCAR and send the appropriate messages with the data from the data stores

For more details on the distinction between pull and push based strategies and the integration options, the reader may refer to the DACTyL project report [2].

Telehealth data input
Telehealth data, in the current setup, comes in the form of periodic data extracts from a central database, the Motiva database that was described in Section 3.1. Since this database exists outside the hospital, and external connections to OSCAR are not allowed, due to hospital security and privacy policies, the data extract is loaded inside OSCAR manually by the OSCAR maintainer and is then parsed to extract the desired data. This includes, among others, patient personal details and information such as patient vital sign measurements.

On a higher level of abstraction, there is no conceptual difference between telehealth and hospital data since they all correspond to input data sources. Having a distinction only serves the purpose of describing two different input philosophies. For a description of the characteristics of each philosophy see Table 7.

	Hospital data	**Telehealth data**
Channel	Web Services	Periodic data extract + ETL process
Source	Internal to the hospital	External to the hospital
Periodicity	Continuously updated	Updated when a new data extract becomes available

Table 7 - The input philosophy of hospital vs. telehealth data

Detailed reports (clinicians)
The purpose of these reports is to give the clinicians an insight into the linked, identified data that OSCAR has gathered. Clinician reports take the form of dashboards with interactive tables and diagrams. The reports should be exposed and viewable by clinicians within the hospital network. Access rights to the reports are determined by the hospital IT administrator.

De-identified data extracts (research)
Research data extracts are a copy of the linked clinical and telehealth data that exist within OSCAR. Due to privacy policies, any patient identification data is removed before exporting these outside the hospital context.

6.2 OSCAR system overview

Based on the problem analysis in Chapter 3, the data warehousing domain analysis in Chapter 4, the system requirements in Chapter 5, and the previous analysis OSCAR is divided in three components.

1. Data Collection
2. Data Warehouse
3. Reporting

As shown in Figure 15, data enters OSCAR via the Data Collection component. Then, it is processed and stored in the Data Warehouse. Finally, it is exported via reports with the Reporting component. Data extracts for research are exported directly from the Data Warehouse.

The division in these components was chosen to underline the separation of concerns within OSCAR, to gather, store, and export/report on data. It is also a way of isolating the data gathering from the reporting functionality.

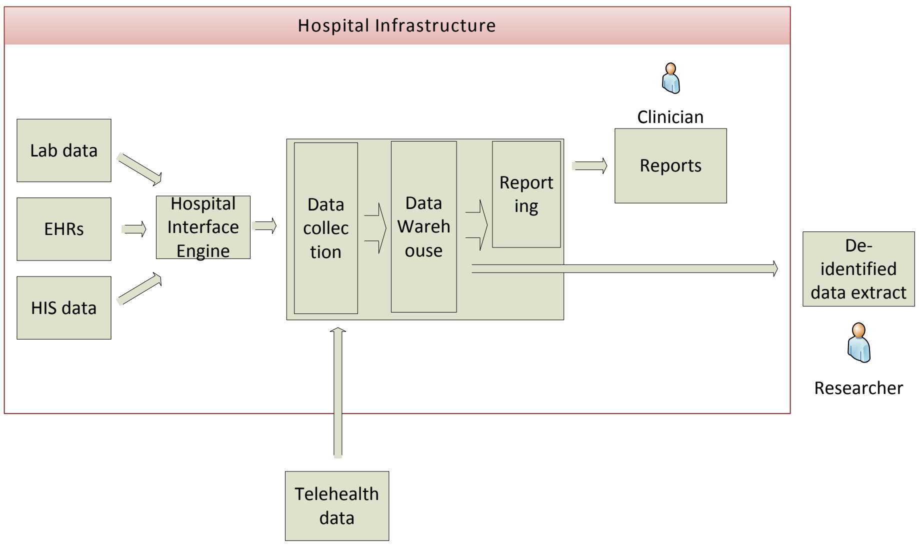


Figure 15 - OSCAR high-level architecture

6.3 Data Collection

The Data Collection component exists in order to decouple the rest of the OSCAR system from the external world, in this respect it can be viewed as a proxy of the Data Warehouse. It is an implementation of a data warehouse staging area. The Data Collection component carries the burden of managing the multiplicity of inputs and providing a simple output towards the Data Warehouse.

Some of the challenges that the designer of this component faces are the following

- Different/multiple inputs (different systems, different hospitals)
- One harmonized/normalized output
- Data transformations defined in an extensible way
- Easy to develop/deploy extensions

The following section shows how these challenges are handled at the architecture level by discussing the data flow within OSCAR.

6.3.1. Data flow

In general, data flows in from external systems via the OSCAR interfaces (web services), it is collected and stored in a central, dimensional database warehouse and it is exposed for reporting, see Figure 18.

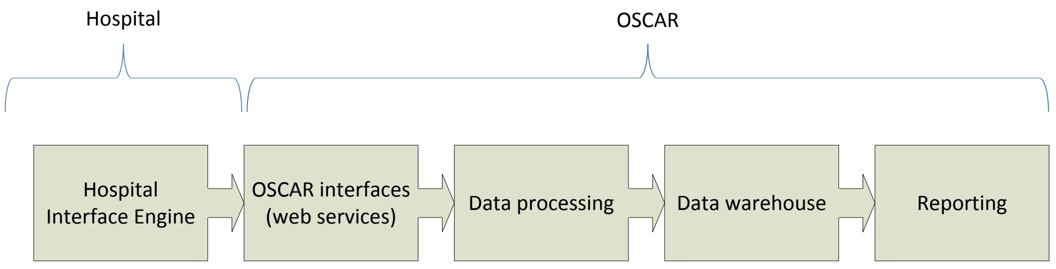


Figure 16 - OSCAR data flow

OSCAR is required to support multiple input interfaces and be extensible to new ones. In general, the more interfaces there are, the more options will be available in the process of integrating with a new system. Also, there is a possibility that a hospital wants to communicate with OSCAR using a specific interface, it should be possible then for the OSCAR maintainer to add support for that new interface.

As discussed, web services can be used as the interfaces to connect OSCAR to data input sources. Web services operate by exchanging messages in a request-response fashion. The messages that are exchanged conform to specific data models, also called schemas. OSCAR must support multiple input data schemas and be extensible to new ones.

It is also a requirement to support reporting on the linked data. Therefore, the data from the multiple sources needs to be linked, and stored in a single place in order to support the subsequent reporting. Essentially, this justifies the use of the data warehouse architecture in this project.

Data transformations
This section discusses the processing of the data after it has entered OSCAR via the interfaces (web services). As discussed in Section 4.3, data transformations are a necessary ingredient in any data warehouse. In OSCAR we considered two possible approaches to designing these transformations. The approaches are shown in Figure 17 and Figure 18.

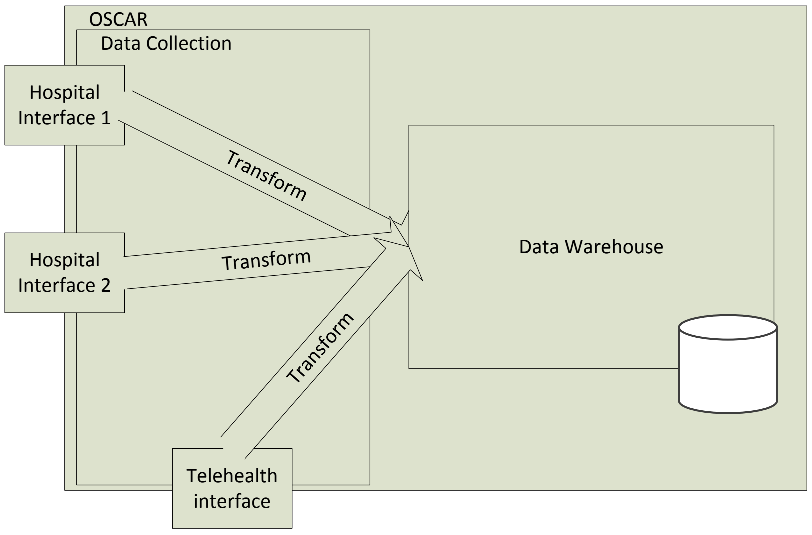


Figure 17 - Input transformation without a landing area

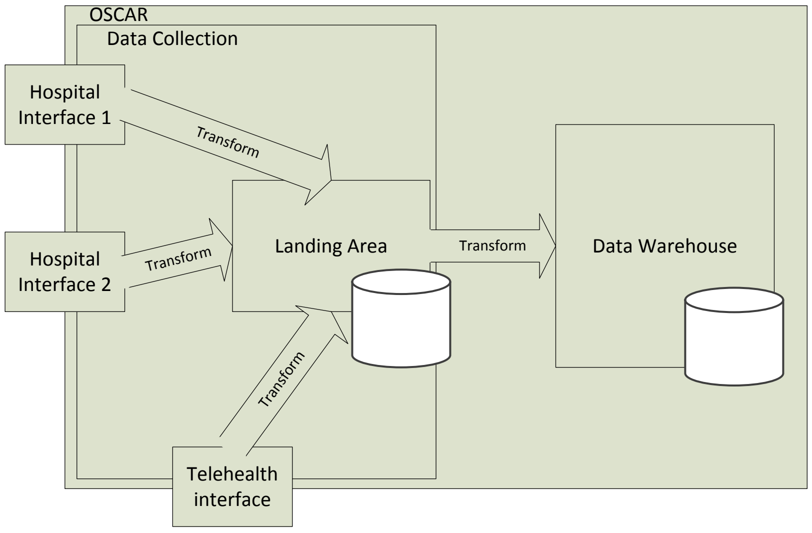


Figure 18 - Input transformation with a landing area

In Figure 17 input data is transformed on the fly by the Data Collection component and loaded directly to the data warehouse. In Figure 18 input data is first transformed and then loaded in the landing area database. Then, in a separate process, it is transformed and loaded into the data warehouse. The second approach has several advantages over the first one, which make it more suitable for OSCAR.

- It adds one more degree of freedom in the design of the transformation process.
- It allows for asynchronous communication between the Data Collection and Data Warehouse components. The benefit is that input data are stored as soon as they enter OSCAR leading to the lower usage of HIS resources at the time of the communication. The ETL processes that load the landing area data to the Data Warehouse can be performed at a different time when it is more suitable to the HIS.
- It allows transformations that are common to the various inputs to be implemented only once (between the landing area and data warehouse), thus avoiding duplication of functionality.

- There is a clear separation of concerns in the transformations; individual transformations are concerned only with getting the data inside OSCAR, whereas the aggregate transformations are concerned with cleaning the data, linking and maintaining a consistent state in the data warehouse.

The design of the schema of the landing area is described in Chapter 7.

Semantic interoperability
Semantic interoperability was explained in Section 4.4. In order for two systems to achieve semantic interoperability, the exchanged messages must conform to a model that is known to both systems, see Figure 19. Such a model is, for example, the HL7 CDA. When the exchanged messages are described in XML, the model is described using an XML Schema definition. Before integrating Philips and the hospital agree on the XML Schema definitions that will be used for the exchanged messages between the OSCAR and the HIS; then, the OSCAR maintainer customizes OSCAR and the hospital IT administrator customizes the hospital interface engine accordingly. Using this model-driven approach, it is much easier to integrate, since the model description is the only thing necessary to fully describe the messages passed between the hospital interface engine and OSCAR. Also, there exist software tools [6] that can parse the XML Schema definitions and automatically generate the processing infrastructure required.

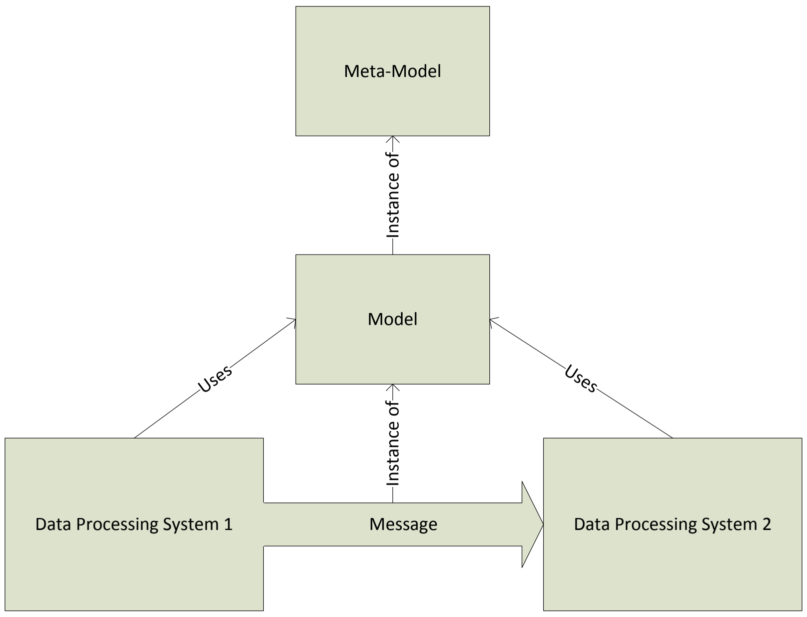


Figure 19 - Interfacing via a common data model

Syntactic interoperability
When OSCAR is deployed in a hospital it exposes web services that are ready to receive incoming data from the hospital 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 is XML documents packaged in SOAP messages. Http is deemed sufficient since the communication takes place strictly within the hospital intranet.

6.4 Data Warehouse and Reporting

The Data Warehouse component is responsible for storing the data that OSCAR receives. It should be able to accept its input data from the Data Collection component. Based on data warehousing concepts this storage was chosen to be a relational database in a star schema (dimensional design). The data warehouse should be able to hold the variety of the data that we want to report on, i.e. all data that enters OSCAR and be extensible to new facts and dimensions.

The Reporting component is responsible for generating and exposing OLAP cubes based on the data in the Data Warehouse. It is also responsible for exposing, via network, the dashboards for the clinicians.

Linking
The star schema of the data warehouse consists of facts and dimensions that describe (healthcare) business processes, for example hospitalizations. The various facts should be linked. This means that some dimensions need to be shared between facts, the data warehouse bus.

Linking was discussed in Section 4.3. There it was established that there is a need for a common entity on which to base the linking on. In the case of OSCAR, we assume that such an attribute is the social security number of the patient in combination with the date of birth. For the Netherlands the identification number is the BSN. Figure 20 schematically demonstrates the process.

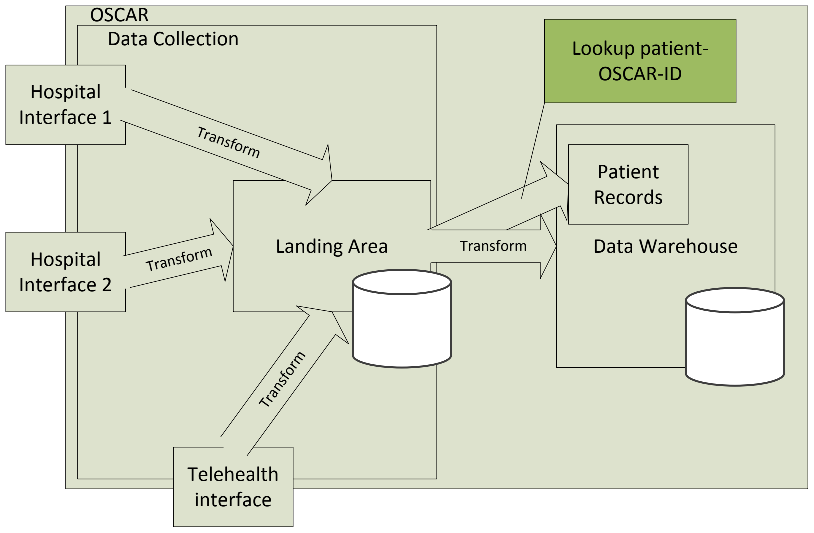


Figure 20 - OSCAR data linking

In Figure 20, patient data from the landing area is matched against the patient records in the data warehouse. Each patient known to OSCAR receives a specific unique ID. The patient matching operation is called a lookup. In case patients do not exist in the system yet, new patient records are created to describe their data.

De-identified data extract
In order to facilitate de-identified data exporting for researchers the Data Warehouse component contains an ETL process that generates a de-identified data extract. This data extract can be provided to the researchers at Philips in a way that conforms to the hospital privacy policies; for example a date and time can be agreed for the researchers to connect to the deployed OSCAR instance via a secure remote connection to download the extracted data files. If the previous approach is not accepted by the hospitals due to the external internet connection that is required, another approach is for a Philips researcher to visit the hospital premises and be given access via the hospital intranet.
■

7.System Design

This chapter continues where the Architecture description left off and elaborates on the internal design of each of the three main OSCAR components that were identified in Chapter 6, namely Data Collection, Data Warehouse, and Reporting.

7.1 Data Collection

The Data Collection component exposes interfaces that allow external systems to connect with and push healthcare data to OSCAR. Various standards and technologies can be used for this purpose.

In addition to being used for interfacing, web services may also be used for the internal structure of the component, in a service oriented design. Using a service oriented design provides a high level of flexibility. Services may be created, deployed and modified independently of each other; also, via the use of WSDL, a service oriented design encourages the use of well-defined interfaces between the various sub-components/services.

In OSCAR the web services are organized in a layered pattern where each subsequent layer exposes a new interface to its environment, see Figure 21.

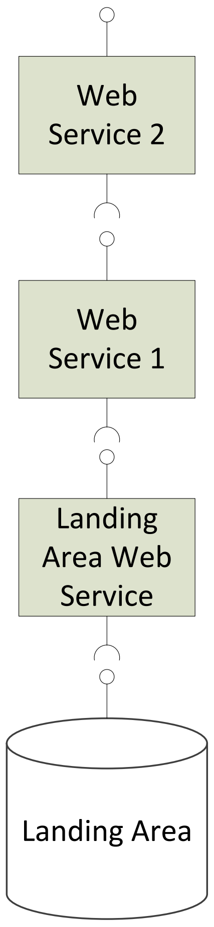


Figure 21 - Data Collection component structure

7.1.1. Landing Area (Persistence layer)

The base of the Data Collection component is a persistence layer, the landing area database shown in Figure 21. The importance of the landing area was discussed in Section 6.3. The landing area resembles a bucket where data is stored until it is processed and moved to the Data Warehouse.

The landing area database and the relational database management system (RDBMS) that exposes it constitute a minimal working Data Collection component. The RDBMS allows the remote insertion of data into the system by issuing SQL commands to the landing area database. This approach is used when the data from Philips Motiva telehealth services are loaded into the system. To insert those data into OSCAR there is an ETL process that extracts data from their source transforms it, and loads it to the landing area by connecting directly to the RDBMS.

7.1.2. Web Service layers

OSCAR exposes web services to facilitate the communication between itself and external systems. The “back-end” of the web services can be connected to the landing area and provides the means for the external systems to load data into it. The external system in the case of the hospitals is the hospital interface engine described in Chapter 4.

To improve the integration of OSCAR with other systems, the web services are described in Web Service Definition Language (WSDL) files. By providing these WSDL files to the hospital, it is straightforward for the hospital IT administrator to use existing tools to generate the communication infrastructure (web service client) required for exchanging messages with OSCAR. Some of the most commonly used programming languages (Java, C++, .NET), along with popular ETL tools (Oracle, Microsoft) do support fully automated generation of the web service client for a well-defined WSDL, thus accelerating the integration process.

For future interfacing requirements, such as a custom input schema requested by a hospital, it is possible to implement web services to support any format of input as long as the web service can transform and store it to the landing area or pass it to other web services for subsequent processing.

Specific input schemas that can be supported by the exposed web services include the following:

- **Landing Area schema:** The most straightforward option is to expose a web service that accepts messages in the landing area schema, as no schema transformations would be needed to load the input messages into the landing area database.
- **DACTyL hospitalization schema:** A schema that comes from the DACTyL project, described in Chapter 1. It is a custom input schema for receiving hospitalization data.
- **HL7 CDA schema:** The HL7 CDA schema, described in Chapter 3, is adopted by the NHS, UK and other healthcare institutions. It is already in use by hospitals and therefore it would require less effort from the hospital side to integrate with it.

Whereas Figure 21 depicts the generic design of the Data Collection component, Figure 22 shows the design of the specific Data Collection component that was implemented. It contains an example of a three-layered Data Collection component. The external systems (for example the hospital interface engine) can be connected to the hospital data web service or the landing area web service depending on which is easier for the hospital IT department to support. In addition, the Motiva data are loaded directly to the landing area via an ETL process that connects directly to the RDBMS.

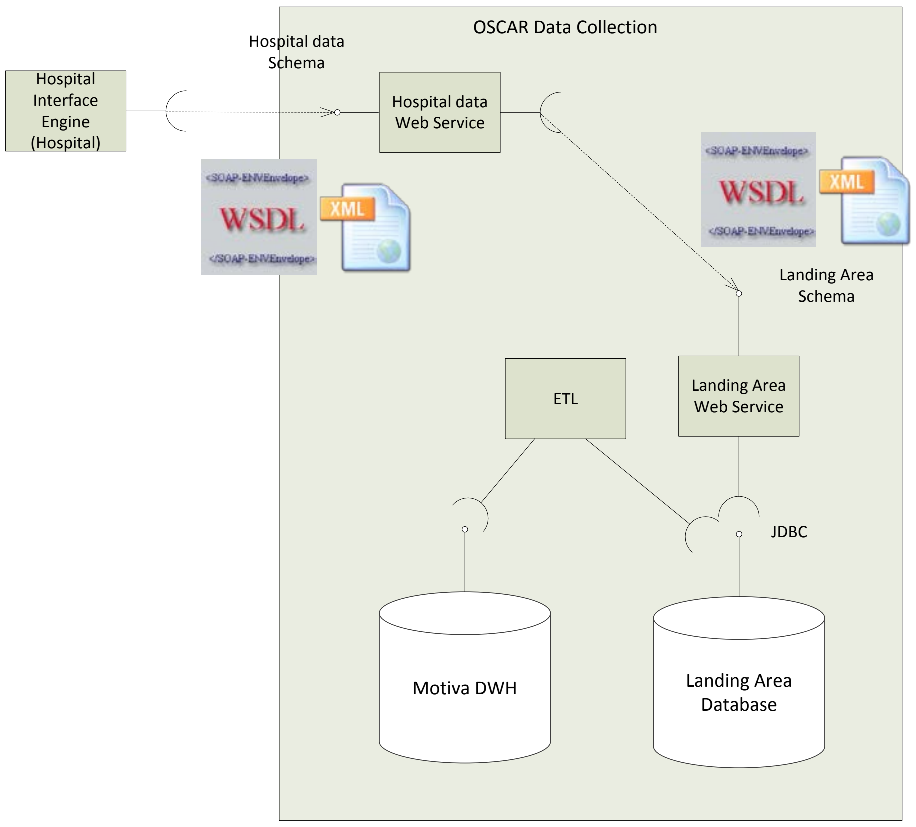


Figure 22 - Data Collection component design

7.1.3. Web services internal design

Landing Area Web Service

The landing area web service accepts messages in the same schema as the landing area. Once a message has been received its contents are stored in the landing area. Figure 23 outlines the data flow that takes place within the landing area web service.

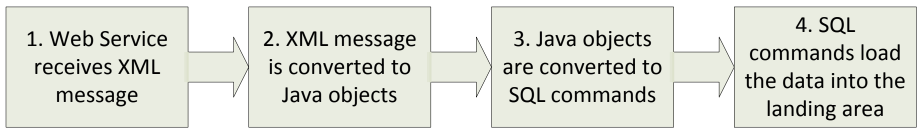

1. Web Service receives XML message
2. XML message is converted to Java objects
3. Java objects are converted to SQL commands
4. SQL commands load the data into the landing area

Figure 23 - Data flow within the landing area web service

The landing area web service consists of the following components:

- A web service described in WSDL that accepts XML messages that conform to the landing area schema (implements Steps 1 and 2 in Figure 23)
- A database connection that is used to store the input messages into the landing area (implements Steps 3 and 4 in Figure 23)
- A description of the landing area schema (as an XML Schema (XSD) and using Java classes) (this description must exist in order to make Steps 2 and 3 in Figure 23 possible)

Figure 24 shows the class diagram of the landing area web service.

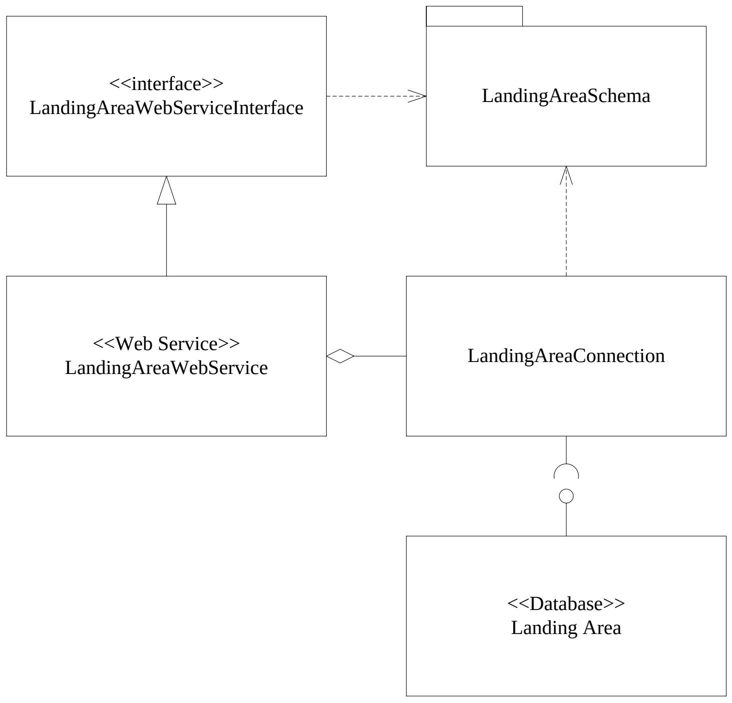


Figure 24 - Landing Area web service

In Figure 24 it is important to note that both the web service interface and the landing area connection use the same schema (the same data model).

Hospital data web service

The hospital data web service accepts messages in an input schema, for example the HL7 CDA schema. Once a message has been received its contents are transformed and sent to the landing area web service. The hospital data web service consists of the following components:

- A web service described in WSDL that accepts XML messages that conform to the desired input schema
- A client that sends output messages to the landing area web service; the client must conform to the WSDL description of the landing area web service
- A description of the input schema (XSD, Java)
- A description of the landing area schema (XSD, Java)
- A transformer that transforms instances of the input schema to instances of the landing area schema

Figure 27 shows the class diagram of the hospital data web service.

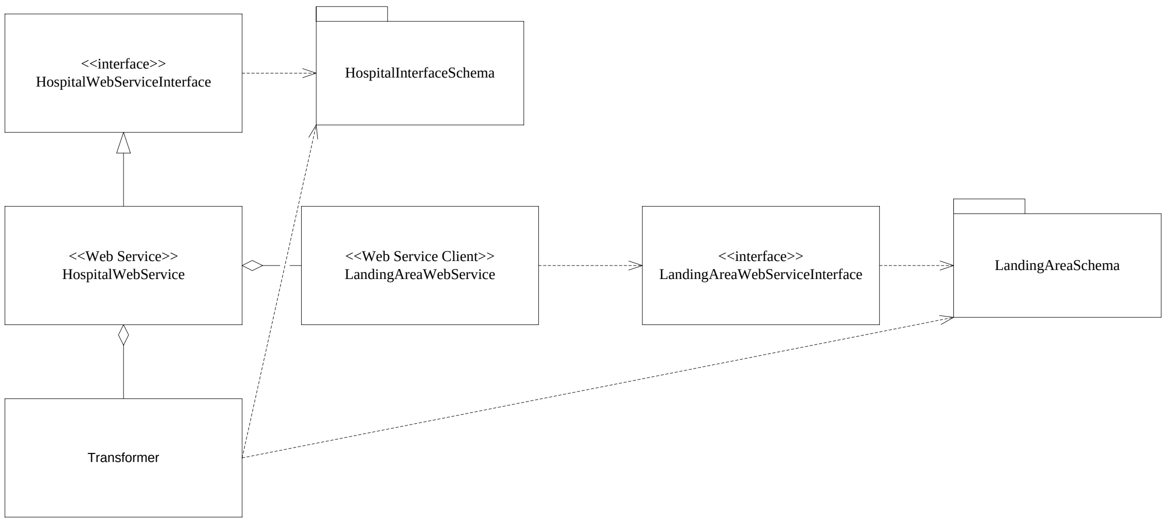


Figure 25 - Hospital data web service

It is interesting to note the differences between Figure 25, the hospital data web service and Figure 24, the landing area web service. In Figure 25 the connection to the database has been replaced by the client to the landing area web service. Also, a new component has been introduced, the transformer, which transforms the input data from the input schema to the landing area schema.

7.1.4. Landing Area schema

The data model of the landing area needs to be designed in such a way that it improves the extensibility characteristics of OSCAR. In terms of the data flow within OSCAR, the landing area is an intermediate step between the input interfaces and the data warehouse, see Figure 26. To "flow" between the input interfaces, the landing area, and the data warehouse, the data needs to be transformed from one schema to another. Initially, data from each input interface is transformed individually, and then all the data in the landing area is transformed aggregatively. Conceptually, the individual transformations are handled by the OSCAR Data Collection component and the common transformations are handled by the OSCAR Data Warehouse component.

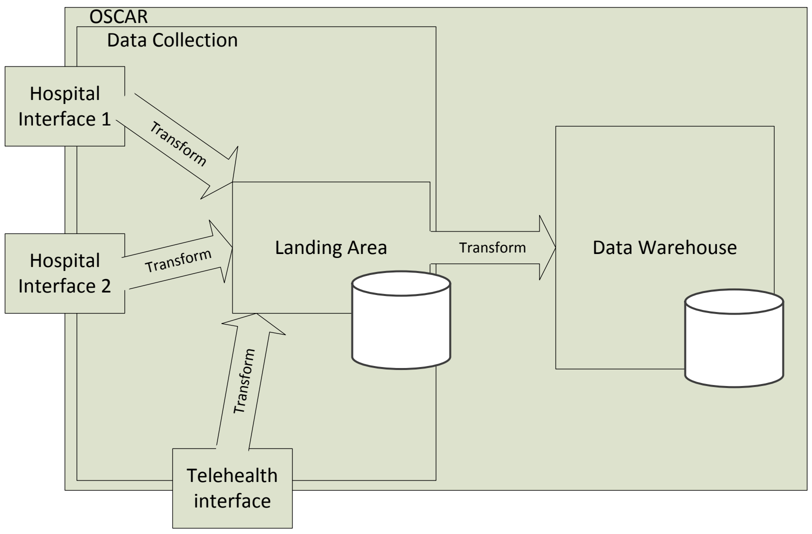


Figure 26 - Input transformations using a landing area

The Landing Area is what determines the division between the individual and the aggregate transformations. The ideal design would be one where individual transformations only account for the unique characteristics of each input model and the rest is handled by the aggregate transformations.

The design drivers for the Landing Area schema are the following:

- The Landing Area schema should depend on the Data Warehouse schema and not on the input schemas, for extensibility reasons
- It should be made easy to transform the input data to the Landing Area schema

We can bridge the two, seemingly conflicting, requirements by designing the Landing Area schema as a de-normalized version of the Data Warehouse schema. This means that the Landing Area schema mirrors the data warehouse schema with the difference that the foreign keys in each table are substituted with the attributes of the table they point to. Figure 27 demonstrates this with an example.

In the example, the data warehouse schema consists of a Fact table that records hospital admissions and the relevant Dimension tables, Patient and Calendar. The landing area schema contains the same tables but each table has been "de-normalized" to include the attributes of the tables it depends on; in this case, the Fact_Admission table has had its foreign keys replaced with the attributes of the corresponding dimension table.

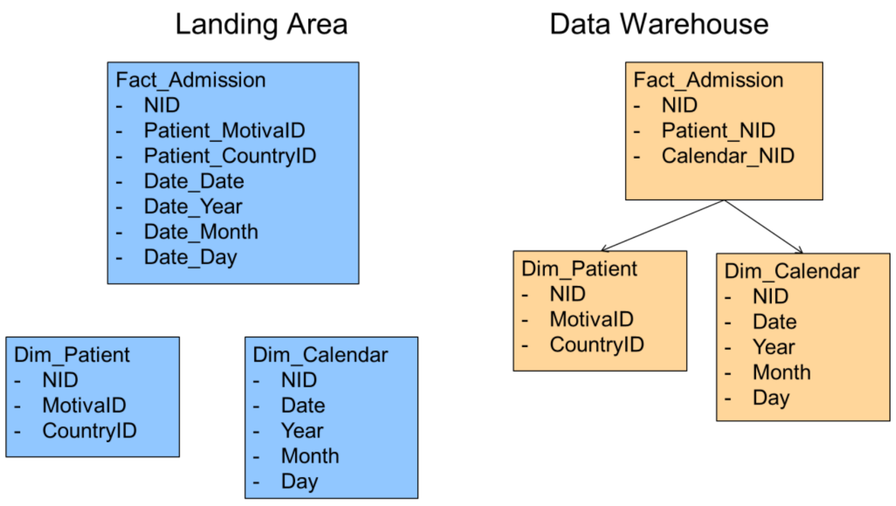


Figure 27 - Example of a landing area schema based on the data warehouse schema

The use of this design for the landing area simplifies the transformations. It is easier to transform the input data to the landing area schema than to the data warehouse schema, which is the main goal of using the landing area. Also, transforming data from the landing area to the data warehouse is straightforward since the tables in the two schemas have a one-to-one correspondence.

The advantages of the proposed design are listed below:

- Simpler transformations (individual and common)

- Lower implementation redundancy (the transformation between the landing area and the data warehouse is implemented only once and works for all input data)
- The landing area design depends directly on the Data Warehouse schema and not on the input schemas (stability)
- The solution is conceptually intuitive, the landing area represents an intermediate step in the transformation between input interface and data warehouse schemas
- The creation of the Landing Area can be automated based on the data warehouse schema

7.2 *Data Warehouse*

The Data Warehouse component is at the core of the OSCAR functionality. On the input side it is the place where all data is linked and stored. On the output side it forms the basis for the provided reports and data extracts. It is illustrated in Figure 28 and consists of the following parts:

- The analytical database (also called data warehouse)
- A copy of the analytical database but which does not contain patient personal identification data, the de-identified database
- The ETL process that loads data from the landing area to the data warehouse
- The ETL process that de-identifies the data for exporting

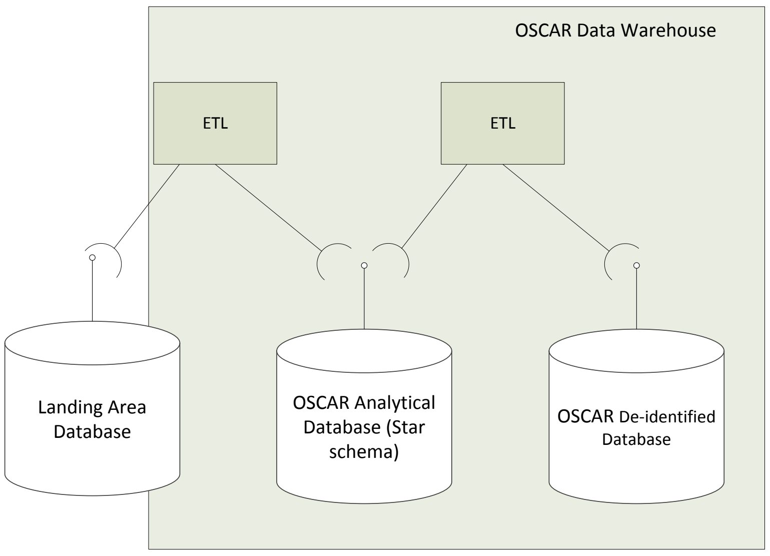

OSCAR Data Warehouse
ETL
ETL
Landing Area Database
OSCAR Analytical Database (Star schema)
OSCAR De-identified Database

Figure 28 - OSCAR Data Warehouse component

7.2.1. Data Warehouse schema

The Data Warehouse schema is in the form of a star schema, as dictated by the data warehouse design principles mentioned in Chapter 4. The facts and dimensions are derived based on:

- Previous Philips projects, namely the Motiva DW and DACTyL projects
- Specific data requirements proposed by the clinician
- Concepts from the HL7 RIM
- Concepts based on Clinical Data Warehouse implementations from the literature

On one hand the specific schema of the Data Warehouse should be stable and support a wide range of healthcare business processes. On the other hand the goal is not to limit OSCAR to some specific facts and dimensions, but rather to allow flexibility to add new ones and remove existing ones in the future.

7.2.2. Loading the landing area data into the data warehouse

Section 7.1 described the landing area schema and its relation to the data warehouse schema. Since we derived the landing area schema directly from the data warehouse schema, we should now be able to go "backwards" and design a generic way to transform the landing area data to load it into the data warehouse.

Process 1 - Loading tables without foreign keys (dimension tables)
The process to load dimension tables is simple since the corresponding table in the landing area is exactly the same, so there is one-to-one mapping of the attributes. Of course the ETL process has to take care of checking for consistency and checking for updating existing records, if they already exist.

Process 2 - Loading tables with foreign keys (fact tables)
The landing area table that corresponds to a fact table in the data warehouse contains the attributes of the fact table and the attributes of all related dimensions. The process to load a fact table has to follow the steps outlined below:

1. Split the attributes of the fact table and the related dimension tables
2. Load each related dimension by using Process 1
3. Lookup all foreign keys for the relevant dimension tables
4. Using the looked-up foreign keys and the attributes of the fact table fill in the fact table using Process 1

The process can skip any dimensions for which data is pre-calculated and pre-loaded to OSCAR. Such dimensions are Calendar, Time, and ICD-10 codes.

The previous description can be better understood with an example. Suppose that we have the landing area and data warehouse as described in Figure 29.

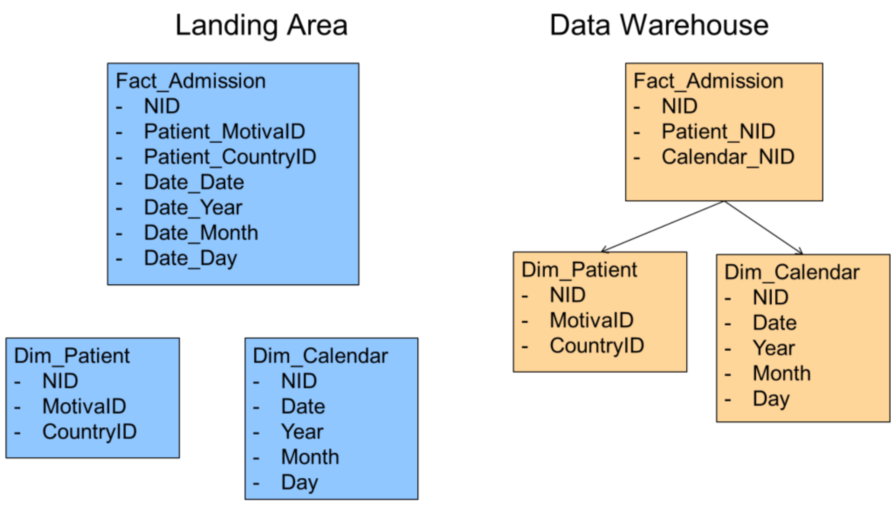


Figure 29 - Landing area example

The ETL process then performs the following steps:

Load dimensions

1. Load the patient dimension using Process 1 (take care to handle existing records correctly, avoid duplicates)

Load facts

1. Load the patient contained within the Fact_Admission table in the landing area using the attributes Patient_MotivaID, Patient_CountryID (take care to handle existing records correctly, avoid duplicates)

2. Lookup the surrogate key of the patient that corresponds to the attributes Patient_MotivaID, Patient_CountryID in the Dim_Patient table
3. Lookup the surrogate key of the date in the Dim_Calendar table
4. Load Fact_Admission in the data warehouse using the keys from steps 2 and 3 (take care to handle existing records correctly, avoid duplicates)

7.2.3. De-identification

De-identification is performed by altering the rows that contain patient personal information. The schema of the de-identified data extract is the same as the data warehouse schema only without any information that can link patient data to the actual person. This can be achieved by omitting all the patient personal attributes in the patient dimension table in the data warehouse.

7.3 Reporting

The Reporting component offers access to the data stored in the data warehouse. It exposes this data in the form of reports and dashboards intended to provide an insight in underlying trends in the data.

The Reporting component is illustrated in Figure 30 and consists of:

- OLAP cubes that expose the data contained in the data warehouse
- Dashboards that are exposed via a web application that connects to the OLAP cubes and visualize their data

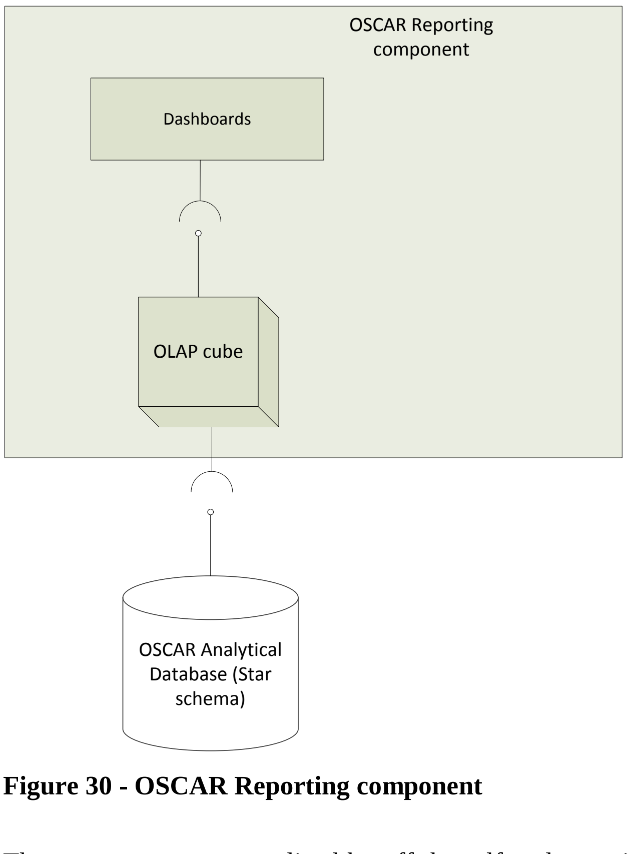

OSCAR Reporting component
Dashboards
OLAP cube
OSCAR Analytical Database (Star schema)

Figure 30 - OSCAR Reporting component

These components are realized by off-the-self tools to minimize the implementation effort required. These tools require configuration that specifies the location of the data warehouse, the data model and which tables/attributes are available for usage in the end user reports.

■

8.Implementation

This chapter describes the implementation of OSCAR. The discussion starts with a list of the tools and technologies that were used and continues with a description of each component. The implementation is based on the system requirements, architecture, and design described in Chapters 5, 6, and 7.

8.1 Process

The OSCAR implementation process consisted of the creation of a series of prototypes, each one enriched with improved functionality over its predecessors. The initial implementations were based on the same technologies as the DACTyL project, whereas later ones diverted from them, to better support the system requirements and constraints.

Table 8 presents an overview of the technologies used for the implementation of each OSCAR component.

Component	**Implementation tools**
Data Collection	Java, Eclipse, Apache Tomcat server
Data Warehouse	Microsoft SQL Server: Management Studio, Integration Services
Reporting	Microsoft SQL Server Analysis Services, SharePoint, SharePoint Dashboard Designer

Table 8 - Technologies used per OSCAR component

8.2 Technical background

Every software project is an attempt to build the desired functionality out of the set of languages, frameworks, and tools that are available. This section briefly describes the main technologies that were used to implement OSCAR. Some alternatives are also mentioned, which were used in early prototypes but were later rejected.

8.2.1. Microsoft SQL Server [7] and related tools

The Microsoft SQL Server is one of the most popular database management systems available. Furthermore, it comes bundled with tools for the creation of data warehousing solutions and business intelligence applications.

The Microsoft SQL Server was used due to its popularity, its alignment with the task of building a data warehouse, and the fact that the Philips IT department strongly encouraged the use of Microsoft tooling.

SQL Server Management Studio (SSMS) [8]
It is a software application that is used for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools. All the databases in OSCAR are built using this tool.

Microsoft SQL Server Integration Services (SSIS) [9]
SSIS is a tool that allows the user to build ETL processes and define workflow tasks. The OSCAR ETL processes that are implemented with SSIS are the following:

1. Loading the landing area data to the data warehouse
2. Loading the Motiva data to the landing area
3. De-identifying the data warehouse to generate the data extract

The ETL processes, with respect to the OSCAR architecture are shown in Figure 31

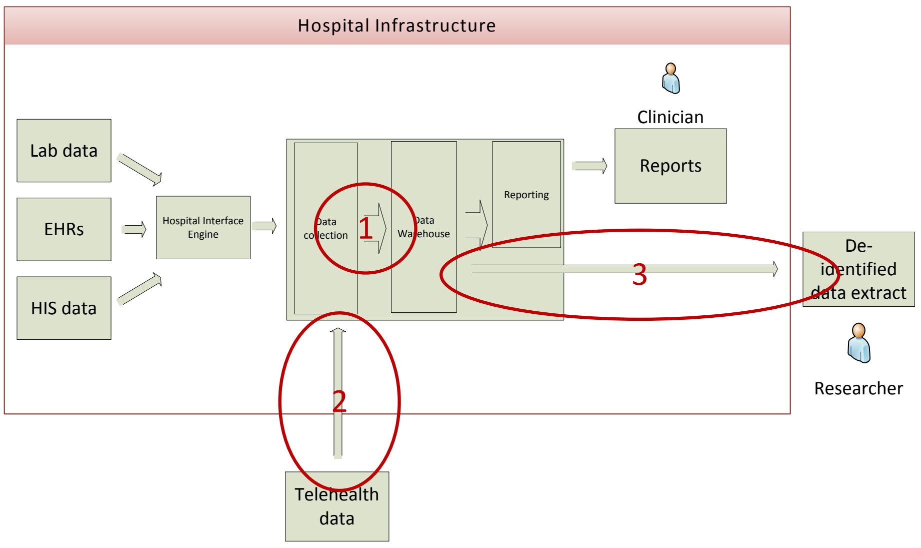


Figure 31 - ETL processes built with SSIS

The workflow tasks that are implemented using SSIS include the periodic refreshing of the data in an OLAP cube and the periodic execution of the aforementioned ETL processes.

Microsoft SQL Server Analysis Services (SSAS) [5]
SSAS is a tool that provides a visual way for designing and deploying OLAP cubes. It is used to build the OLAP cube that exposes the OSCAR data to a reporting application.

8.2.2. Microsoft SharePoint [10]

SharePoint is a web application platform, with tools intended for non-technical users. SharePoint can provide intranet portals, document and file management, collaboration, social networks, extranets, websites, enterprise search, and business intelligence. For OSCAR we are interested in the last capability, business intelligence.

SharePoint provides a tool to build interactive dashboards by connecting to an OLAP cube and identifying the relevant Facts, Dimension and data. It is then able to host these dashboards so that they are viewable via a web browser.

SharePoint was chosen because of its closeness to the other Microsoft tools we were using and because of its aim towards non-technical users.

8.2.3. Eclipse [11]

Eclipse is one of the most popular IDEs. It contains a base workspace and an extensible plug-in system for customizing the environment to support the development of any kind of application.

Eclipse was used as the development platform for the Connection layer (web services) in the DACTyL project, so it was chosen to be used for this project as well. In fact, initial OSCAR versions also used the same plug-ins as DACTyL, but were later changed to others that were deemed more suitable.

Eclipse Modelling Framework (EMF) [12] **and Teneo** [13]
EMF is a model driven development framework where the developer can design models and the framework can produce java source code that implements them. For more information see the DACTyL report [2].

One of the most useful tools of EMF is Teneo. Teneo provides the ability to map database tables to EMF models, thus creating a model-oriented way of storing models in a database. Internally, it uses Hibernate and performs the mappings automatically to ease the developer.

Teneo was used in the early versions of OSCAR, but it was later replaced by Hibernate, more details are provided in Section 8.3.

Eclipse Enterprise Edition
We decided to use the Enterprise Edition of Eclipse due to its support for creating applications that include, and combine, web services, XML, and data storage.

8.2.4. Java annotations [14]

Annotations are a form of syntactic metadata that can be added to the Java source code. Annotations are often used by frameworks as a way of conveniently applying behaviors to user-defined classes and methods that must otherwise be declared in an external source (such as an XML configuration file). Their biggest value is that by using them the source code becomes much more clean and concise. The following annotation APIs are relevant for OSCAR:

- Java API for XML Web Services (JAX-WS), used to define the web services in OSCAR, more details in Section 8.3
- Java Persistence API (JPA), used via the hibernate implementation to define the mapping between Java objects and database tables.

8.2.5. Apache Axis [6]

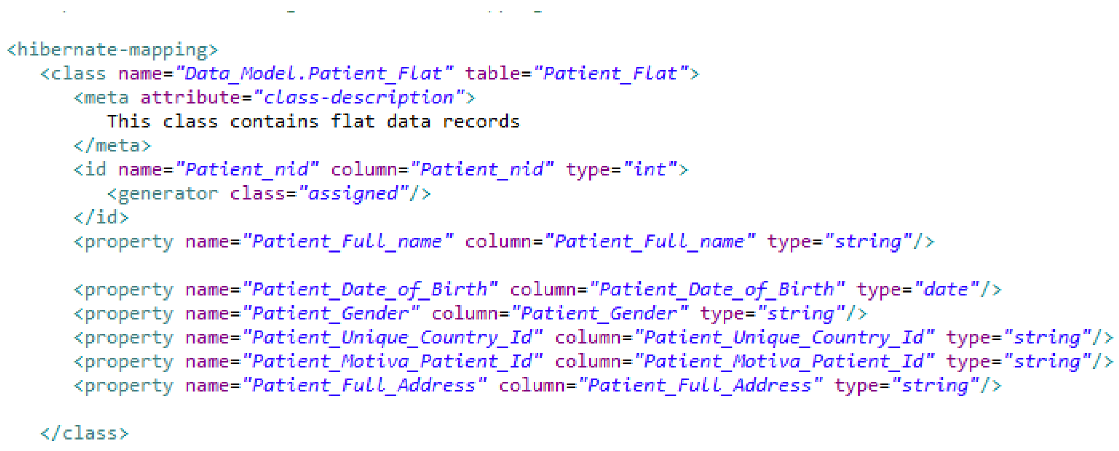
Apache Axis (Apache eXtensible Interaction System) is an open source, XML based Web service framework. It consists of an implementation of the SOAP server and various utilities and APIs for generating and deploying Web service applications. It comes as a plugin in the Eclipse IDE. It is used to generate the client components of the OSCAR web services based on their WSDL definitions.

In earlier versions in was also used to generate the web services, but was later replaced by the JAX-WS annotations.

8.2.6. Hibernate [15]

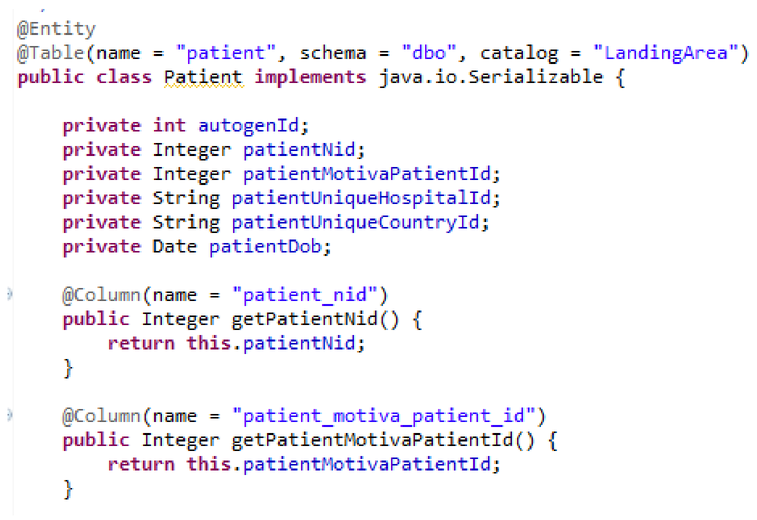
Hibernate is a Java library that converts working with SQL commands to working with Java objects. It does this by mapping each table in a relational database to a Java data object. There are two ways to specify the mapping:

Hibernate mapping file (XML): A hibernate mapping file is an XML document that describes the mapping between the attributes of Java classes and database tables. An example is shown on Figure 31.
JPA annotations (JPA): JPA annotations are Java annotations that essentially perform the same function as the XML mapping file but without the need for a separate file. An example is shown on Figure 32.



```
<hibernate-mapping>
  <class name="Data_Model.Patient_Flat" table="Patient_Flat">
    <meta attribute="class-description">
      This class contains flat data records
    </meta>
    <id name="Patient_nid" column="Patient_nid" type="int">
      <generator class="assigned"/>
    </id>
    <property name="Patient_Full_name" column="Patient_Full_name" type="string"/>

    <property name="Patient_Date_of_Birth" column="Patient_Date_of_Birth" type="date"/>
    <property name="Patient_Gender" column="Patient_Gender" type="string"/>
    <property name="Patient_Unique_Country_Id" column="Patient_Unique_Country_Id" type="string"/>
    <property name="Patient_Motiva_Patient_Id" column="Patient_Motiva_Patient_Id" type="string"/>
    <property name="Patient_Full_Address" column="Patient_Full_Address" type="string"/>

  </class>
```

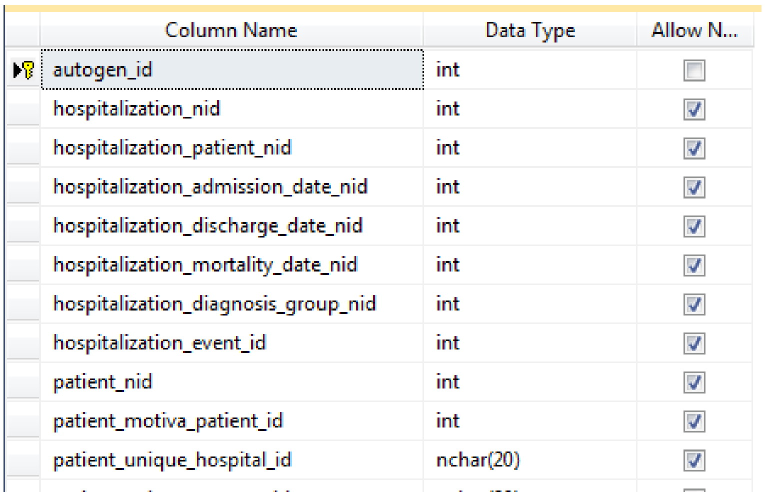
Figure 32 - Hibernate mapping file



```
@Entity
@Table(name = "patient", schema = "dbo", catalog = "LandingArea")
public class Patient implements java.io.Serializable {

    private int autogenId;
    private Integer patientNid;
    private Integer patientMotivaPatientId;
    private String patientUniqueHospitalId;
    private String patientUniqueCountryId;
    private Date patientDob;

    @Column(name = "patient_nid")
    public Integer getPatientNid() {
        return this.patientNid;
    }

    @Column(name = "patient_motiva_patient_id")
    public Integer getPatientMotivaPatientId() {
        return this.patientMotivaPatientId;
    }
```

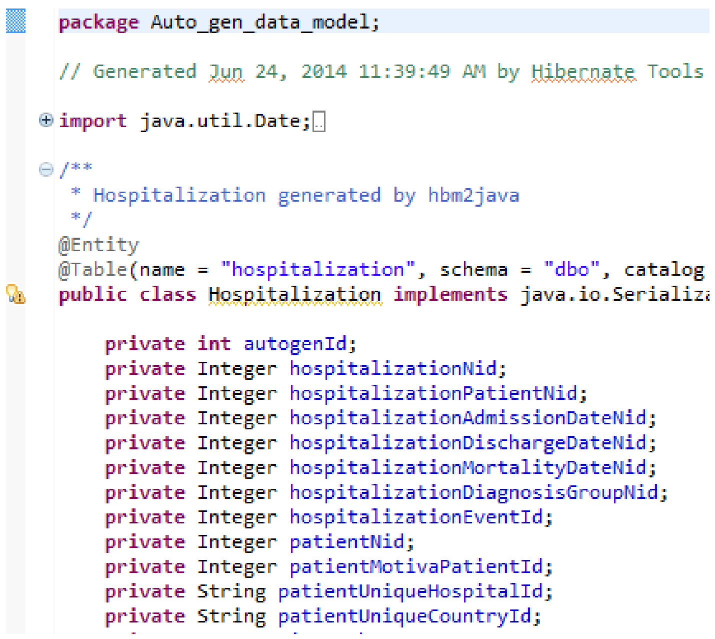
Figure 33 - Java class with JPA annotations

Hibernate Tools [16]
Hibernate Tools are used to generate Java classes that correspond to tables in a database. They are useful in that they automate an otherwise tedious process. They come as a plugin in the Eclipse IDE. Along with the desired Java classes they generate the mapping configuration using either a Hibernate mapping file or annotations. In OSCAR they are used to generate the classes for the tables in the landing area. Figure 34 and Figure 35 show the hospitalization table in the OSCAR landing area and the corresponding auto-generated Java class respectively.



	Column Name	Data Type	Allow N...
▶🔑	autogen_id	int	☐
	hospitalization_nid	int	✓
	hospitalization_patient_nid	int	✓
	hospitalization_admission_date_nid	int	✓
	hospitalization_discharge_date_nid	int	✓
	hospitalization_mortality_date_nid	int	✓
	hospitalization_diagnosis_group_nid	int	✓
	hospitalization_event_id	int	✓
	patient_nid	int	✓
	patient_motiva_patient_id	int	✓
	patient_unique_hospital_id	nchar(20)	✓

Figure 34 - Database Hospitalization table



```
package Auto_gen_data_model;

// Generated Jun 24, 2014 11:39:49 AM by Hibernate Tools

import java.util.Date;

/**
 * Hospitalization generated by hbm2java
 */
@Entity
@Table(name = "hospitalization", schema = "dbo", catalog
public class Hospitalization implements java.io.Serializa

    private int autogenId;
    private Integer hospitalizationNid;
    private Integer hospitalizationPatientNid;
    private Integer hospitalizationAdmissionDateNid;
    private Integer hospitalizationDischargeDateNid;
    private Integer hospitalizationMortalityDateNid;
    private Integer hospitalizationDiagnosisGroupNid;
    private Integer hospitalizationEventId;
    private Integer patientNid;
    private Integer patientMotivaPatientId;
    private String patientUniqueHospitalId;
    private String patientUniqueCountryId;
```

Figure 35 - Java Hospitalization class

8.2.7. Apache Tomcat [17]

Apache Tomcat is an open source web server container. It is used to host the web services in the OSCAR Data Collection component. Initially, the classic version of Tomcat was used, but it was later replaced by Tomcat Enterprise Edition due to its support for Java annotations.

Tomcat Enterprise Edition (TomEE) [18]
Apache TomEE is the Java Enterprise Edition of Apache Tomcat. It is used in OS-CAR due to its support for Java annotations.

8.2.8. VMware tools [19]

VMware is a company that provides cloud and virtualization software and services. The free, personal tool that they provide is called VMware Player. VMware player runs on desktop computers and has the ability to create and run virtual machines. OSCAR was initially developed using this tool.

VMware also offers the vSphere platform which

“Leverages the power of virtualization to transform datacenters into dramatically simplified cloud computing infrastructures and enables IT organizations to deliver the next generation of flexible and reliable IT services, using internal and external resources, securely and with low risk.” [20]

VSphere or similar platforms are used more and more to replace hardware servers due to the inherent advantages they provide. OSCAR takes advantage of their increased popularity and, by being in a form of a virtual machine, can be easily deployed in any organization that hosts a VMware infrastructure.

8.3 Data Collection

As discussed in the Architecture and Design chapters, Chapters 6 and 7, the Data Collection layer follows a layered design pattern.

8.3.1. Landing Area

The landing area that forms the base of the data collection component is implemented using the Microsoft SQL Server tools. The Microsoft SQL Server RDBMS forms the most basic way to interface with OSCAR. The schema of the landing area is derived from the schema of the data warehouse; this was discussed in detail in Section 7.1.

Landing Area Generator

Due to the direct relationship between the landing area schema and the data warehouse schema it becomes possible to build a tool to auto-generate the landing area database by using the data warehouse schema as input.

The Landing Area Generator consists of the following main components:

- **Data Warehouse Schema Getter:** A component to get the schema of the data warehouse, in the case of a data warehouse implemented with Microsoft SQL Server this requires queries to the system tables of the SQL Server
- **User Configuration Getter:** A component to get user configuration, via an XML configuration file
- **Database Builder:** A component to generate the tables of the landing area
- **Landing Area Generator:** A component to orchestrate the whole process

Figure 36 shows the internal structure of the Landing Area Generator tool and Figure 37 illustrates the workflow process of generating the landing area.

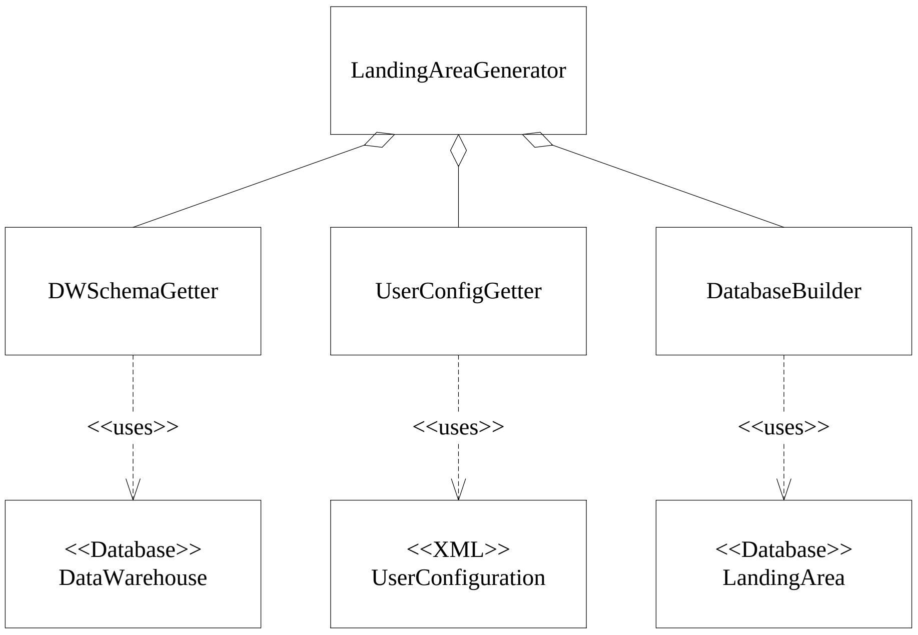


Figure 36 - Landing Area Generator components

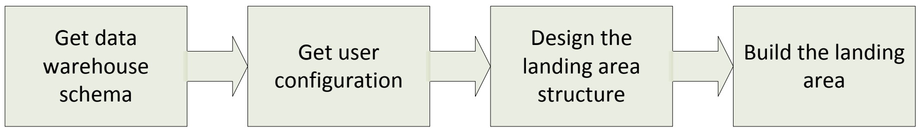


Figure 37 - Landing Area Generator workflow

The components of the Landing Area Generator were implemented in Java using the Eclipse IDE. The connections to the SQL Server database used the JDBC interface. The XML user configuration component was implemented using a library called XStream [21].

The Landing Area Generator is useful in making the implementation of the data collection faster by removing some of the manual work of the landing area database configuration. We implemented it to allow for the faster generation of OSCAR prototypes.

Loading the Motiva data to the landing area
The Motiva data extract is an SQL server database dump. The Motiva data need to be extracted from this database, transformed, and loaded into the landing area. We used SSIS in order to implement the ETL process that performs this task.

Landing Area Web Service
The Landing Area Web Service was described in Chapter 7. The data flow within the landing area web service, Figure 38, is repeated here for easier reference.

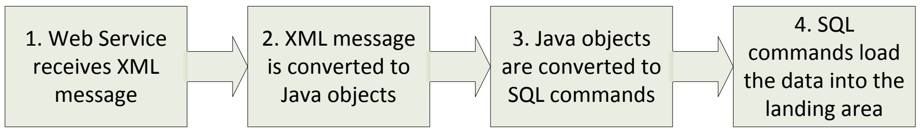

1. Web Service receives XML message
2. XML message is converted to Java objects
3. Java objects are converted to SQL commands
4. SQL commands load the data into the landing area

Figure 38 - Data flow within the landing area web service

The landing area web service is implemented using Java. The first step to implementation is to make our Java program aware of the landing area schema. For this we use the Hibernate Tools to generate Java classes that correspond to the database tables (the Java objects in Figure 38 are instances of these classes). The auto-generated classes already contain JPA annotations that can be used to map the XML message to Java objects as well as to map the Java objects to SQL commands. The former is achieved using the JAX-WS API implemented by the Apache TomEE server and the latter is achieved using the Hibernate tool.

Conceptually a web service takes the form of a function (in the software sense). It has a name, input types, and an output type. Using the JAX-WS API we write this function as a usual Java function and use annotations to define it as a web service. We create a separate web service for each Java class that corresponds to a table in the landing area. The WSDL description and the mapping between XML message and Java objects are implicitly implemented by the TomEE server using the Java annotations.

It is possible to automate the creation of the landing area web services, since they follow the same template, the only variation being the function name and input type. We created a Java program that uses the Apache Velocity [22] (a Java-based template engine) library to generate the web services based on the Java classes in the landing area schema.

By implementing the components as described above we have achieved a seamless way to store the data from the input XML to the landing area. Furthermore, we are able to automatically generate the new landing area and new web services after a change in the data warehouse. The workflow to implement the landing area and the corresponding web services is described in Figure 39.

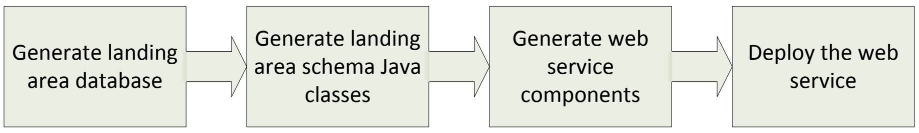

Generate landing area database
Generate landing area schema Java classes
Generate web service components
Deploy the web service

Figure 39 – Data collection implementation workflow

Hospital data web service
This web service is implemented in Java in a similar way to the landing area web service. The difference is that instead of a connection to the database it has a web service client that implements the interface described by the landing area web service WSDL. The client is automatically generated, using the Apache Axis [6] tool, from the landing area web service WSDL file.

The hospital web service is described in its own WSDL file, which can be given to the hospital for configuration of the hospital interface engine. The transformation of the data from the input schema to the output schema is defined in Java.

8.3.2. Design alternatives

This section discusses some technology alternatives that were considered for the implementation of the Data Collection component.

Modeling
By modeling we refer to the way that the various data schemas are represented within the system. We had three main alternatives

- Java classes
- EMF models
- XML schema (XSD)

The decision on which one to use was based on the level of maturity of its related tools and the support of those tools for the OSCAR non-functional requirements.

The main goal was to achieve a seamless flow of data from web service input to database. This was achieved by auto-generating the Java classes from the landing area database (Hibernate tools) and using those same classes with annotations in order to allow Tomcat EE to generate the XSD schema.

Persistence
Persistence here refers to the way that the Java program communicates with the database. The alternatives are listed in Table 7

JDBC	The most primitive way is to use the JDBC interface library provided by Microsoft. This allows for SQL queries to be sent to the database.
<u>Hibernate</u>	Hibernate uses JDBC but allows the programmer to work with Java classes instead of SQL queries directly.
Teneo (EMF)	Teneo goes one step further and allows the programmer to work with EMF models instead of classes or SQL queries.

Table 9 - Persistence technology options

JDBC was rejected from the beginning because it is a much more crude and error-prone way of communicating with the database. Teneo was the easiest one to use but it could not be combined well with the web services. In other words, we could not find a direct way to use the EMF models to define the web services. Also, Hibernate is a much more mature tool in general.

Transformations
The matter of how to define transformations is closely related to the modeling approach that is taken. Some approaches are listed in Table 8.

<u>Java</u>	The most straightforward approach when implementing a Java program
EMF transformations	Various tools that allows the programmer to define EMF model transformations, model to model or model to text
XSLT	A standard for the definition of XML transformations
<u>SSIS</u>	A tool for building ETL processes

Table 10 - Data transformations technology options

SSIS is a good option, especially due to the aim of the tool towards non-technical users, but is only suitable to define transformations between databases and cannot be combined with web services.

XSLT transformations are a good candidate and there exist Java libraries that can execute them. They were not implemented due to limited project time and unfamiliarity on the part of the team members. Nevertheless, they can be the basis for the implementation of future web services.

EMF transformations were considered but rejected due to the immaturity of the tools. Perhaps in the future this will become a more suitable approach.

Java was chosen to implement the transformations mostly on grounds of its versatility.

8.4 Data Warehouse

The Data Warehouse component is responsible for maintaining the analytical database, loading it with data from the Landing Area, and generating the de-identified data extract. It consists of a database and two ETL processes. The database was implemented using the Microsoft SQL Server. The ETL processes were implemented using SSIS.

Implementation of the ETL to load the landing area data
In Chapter 7 we described a generic way to design the ETL processes that load the landing area data to the data warehouse. Based on that design, and shaped by the specifics of the configuration of the SSIS tool our ETL implementation takes its final shape.

SSIS divides the ETL process in Data Flow Tasks. Each task gets the input from a data store, performs various operations on it and loads it in another data store. For the purpose of the ETL processes that load the landing area data the operations on the data correspond to the steps described in Section 7.2.2. The best way to divide out ETL processes for scalability and elegance is to implement a Data Flow Task for each table in the landing area. In this way, new Data Flow tasks can be added, when new tables are added to the landing area without affecting the existing implementation.

8.4.1. Design Alternatives

Building the ETL processes
There were two main alternatives:

- SSIS
- Java

We decided to use SSIS. The decision was based on the following facts

- Using SSIS requires less implementation effort since the tool provides a complete framework for implementing and deploying ETL whereas Java would require much of this functionality to be built from scratch
- The project team has mostly soft programming skills and prefers, where possible, to use off-the-self tools that require configuration instead of programming
- The SSIS solution provides scalability, it is also tested and verified, while with Java such provisions would greatly increase the implementation effort required

A full-blown Java implementation approach would use Service Oriented Architecture to define simple transformations as web services. Other web services would orchestrate the use of the simpler ones to perform the required tasks. We decided that such an approach would essentially re-create the functionality that is already present in SSIS.

8.5 Reporting

The output of the Reporting component is predefined dashboards such as the Return on Investment reports and patient reports that take their input from the data warehouse. The intermediate step between the data warehouse and the dashboards is an OLAP cube.

To implement the OLAP cube we used the Microsoft SSAS tool. The tool requires the connection details to link to the data warehouse. It parses the structure of the database and generates the OLAP cube with limited further configuration required.

In the OLAP cube configuration we can specify which table attributes are visible to the reporting tool and specify the dimension hierarchies for every dimension. Unlike other tools, SSAS does not allow for changes to the underlying data schema such as calculating new attributes.
OSCAR dashboards were built using the SharePoint Dashboard Designer and are hosted on a SharePoint server. The dashboards are compositions of different elements such as interactive tables, user prompts and graphs which are designed based on the structure of the OLAP cube.

The decision to use Microsoft SSAS and SharePoint to implement the Reporting component was driven by the following facts:

- Preference for easy-to-use tool (soft-programming-skilled users)
- Philips IT department encourages use of Microsoft tooling
- Off-the-self components are already tested and verified

It is off course possible in future versions of OSCAR to use different tools for reporting, as long as the tools can connect to the SQL Server data warehouse that hosts the collected, linked data.
■

9.Deployment

This chapter describes the deployment of OSCAR. It starts with an introduction on some general deployment concepts and continues with the specific case of the deployment within the Philips IT infrastructure.

9.1 General

In order to overcome the various privacy related constraints, it was decided that OSCAR should take the form of a virtual machine. Thus, all the components described in the Chapters 6, 7, and 8 should exist within this virtual machine. Virtualization also offers the added ability to provide OSCAR with extensible hardware resources; for example, to add more hard disk space to OSCAR to support increasing data space requirements.

The OSCAR prototype is deployed in the form of a VMWare Virtual Machine. VMWare was selected as it is the leading virtualization platform according to their market share. Also, both Philips and hospitals, where OSCAR will be deployed, host a VMWare virtualization infrastructure. This allows the easy deployment of OSCAR within the hospital VMWare infrastructure and within the Philips VMWare infrastructure for testing.

The Virtual Machine itself contains the Windows Server 2008 operating system. Windows server contains an installation of Microsoft SQL Server 2012, Microsoft SharePoint 2010, and the Apache TomEE server.

Figure 40 shows the planned OSCAR deployment within the hospital IT infrastructure.

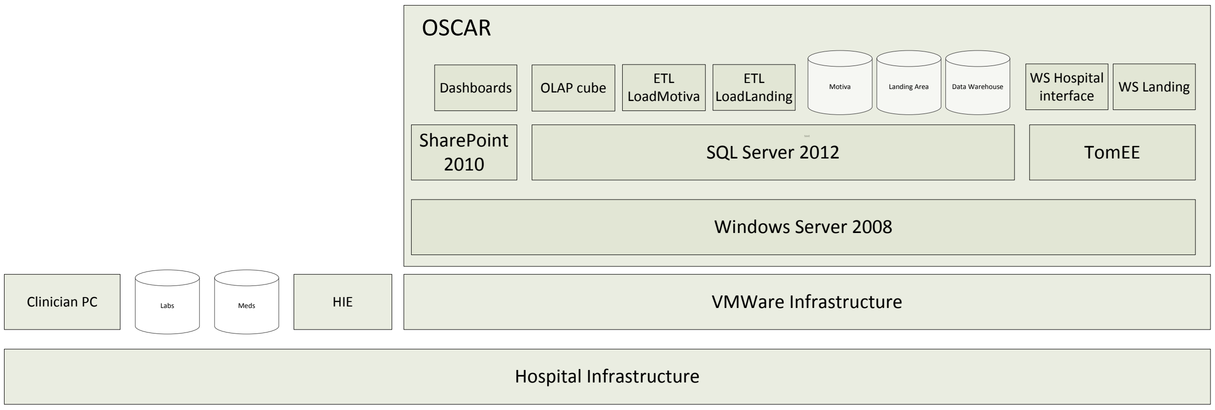


Figure 40 - OSCAR hospital deployment diagram

The bottom layer in Figure 40 is the Hospital IT infrastructure since OSCAR is deployed in the hospital. The hospital IT contains the clinician PC, the various data stores, the Hospital Interface Engine, and the VMWare infrastructure. OSCAR is hosted in the VMWare infrastructure and contains all the described components.

The TomEE server hosts the web services as .war files. The SQL Server hosts the databases and SSIS ETL processes (which are run periodically) as well as the OLAP cube. The SharePoint server hosts the implemented dashboards.

For the deployment to be complete the HIE must be configured to fetch the data from the hospital data stores and push it to the OSCAR web services. Also, the URL of the

OSCAR dashboards must be provided to the clinicians so that they are able to view them via their PCs. In addition, the hospital IT administrator should also determine the access rights to the intended clinicians and open the appropriate network connections.

9.2 Deployment at Philips

OSCAR was deployed within the Philips VMWare infrastructure for testing. The OSCAR virtual machine was initially created and deployed in a laptop using the VMware Player tool. In order to deploy it within the Philips VMWare infrastructure the virtual machine files were copied in a USB stick and given to the Philips IT administrator. He deployed OSCAR and provided its URL location within the Philips intranet so that we could connect via remote desktop connection and manage it.

Later, when new versions of OSCAR were developed, the existing OSCAR deployment was upgraded using the remote desktop connection which provides the look and feel of a physical machine.

■

10. Verification & Validation

This chapter provides an account of the suitability of the OSCAR system to meet the system requirements as described in Chapter 5. The suitability is described in terms of verification (checking if OSCAR works as expected) and validation (checking if OSCAR fulfills the system requirements).

10.1 Verification

This section describes how each OSCAR component was verified. Due to the different technologies involved, we adjusted our approach per component.

Data collection
To ensure the correct functioning of the data collection component, we have to ensure that data from the various inputs reach the landing area correctly. To test this, we implemented a small application that acts as the hospital interface engine and feeds OSCAR with dummy data via its interfaces. This hospital interface engine simulator generates messages with random data and sends it to the hospital interface web service and the landing area web service. Once the messages were sent, the landing area was checked to make sure that it contained the same data as the XML messages.

The WSDL files of the OSCAR web services were tested by using them to generate web service clients in Java as well as in C#. In both cases, the web service clients were able to send messages to OSCAR successfully.

Data warehouse component
In the data warehouse component, we had to ensure that the implemented ETL processes were working correctly. We did this by running the ETL processes with various inputs, while checking the contents of the analytical database.

Concerning the ETL that loads the data from the landing area to the data warehouse, it was tested by running the process multiple times, with various inputs and observing the data that was loaded in the data warehouse. The following cases were covered:

- Run the ETL process with empty input records
- Run the ETL process with duplicates in the input data
- Run the ETL process with input data that already existed in the data warehouse to observe that the processes did not crash and no duplicates were inserted

Reporting component
By using off-the-self tools for the reporting component, the testing was kept to a minimum. We implemented the OLAP cube and dashboards and checked whether they were viewable from a remote PC using a browser and navigating to the proper URL. The dashboards also served as a testing tool since they expose the underlying OLAP cube structure and data revealing any errors.

10.2 Validation

This section summarizes how the requirements for this project were met by mapping the implemented features with the system requirements as described in Chapter 5.

Require-quire-ment Id	**Name**	**Description**
F1	Deployment	OSCAR should facilitate easy deployment in different hospitals.
Met by	The use of a VMware virtual machine facilitates the easy deployment in the VMware	

	infrastructure found in hospitals

F2	Interoperability	OSCAR should be able to communicate with different hospital IT infrastructures, using different data sharing standards, as well as different kinds of schemas defined by the specific informational requirements. For example common such technologies are Http, SOAP, XML and HL7 CDA. It is part of the project to identify the most relevant ones and implement support for them. A common way to achieve interoperability is by describing the interfaces (Web Services) between systems based on WSDL and XSD, therefore the OSCAR should support at least these technologies
Met by	As described in Chapter 8, OSCAR supports both WSDL and XSD. Furthermore, by using a service oriented design for the data collection component we allow for future extensions to support any desirable input technology and schema	

F3	Data import	The on-site deployment of OSCAR should be able to digest data from internal data sources (hospital data) and external data sources (Telehealth, Motiva).
Met by	OSCAR facilitates the importing of telehealth data extracts by copying them to a specific location within the virtual machine. Then, there is an ETL process to load the telehealth data to the landing area. Hospital data are digested in the OSCAR web services.	

F4	Linking	OSCAR should link the incoming data from the different sources, based on patient identification information, in a unified model (data warehouse model). This linking should occur within the hospital boundaries (because the identification information that is required for the linking cannot be taken outside the hospital).
Met by	The data warehouse component of OSCAR maintains records of patients and their personal details. The ETL process that loads the data from the landing area to the data warehouse uses these records to perform the linking.	

F5	Storage	OSCAR should store all the incoming data, in other words no incoming data should be “thrown away”.
Met by	There are no processes implemented for emptying the defined Data Warehouse, only the hospital IT administrator can do that. Also, the data warehouse and landing area schemas are designed so as to be able to store all the data contained in the input messages.	

F6	Data export	The on-site deployment of OSCAR should provide methods to export linked, de-identified data to an external destination, in order to facilitate off-site analysis of the de-identified data by Philips research.
Met by	There is an ETL process in the data warehousing component that de-identifies the data and creates the data file for exporting.	

F7	On-site reporting	The OSCAR should offer detailed reporting on-site based on the linked, identified data. This reporting is to be used by the clinician for decision support.
Met by	The requirement is met by the dashboards that are exposed by the SharePoint server based on the SSAS OLAP cube.	

F8	Supporting	OSCAR should be extensible in order to add support for new or addition-

	future use cases	al informational needs of its users with minimal effort. OSCAR should be extensible with respect to different hospital IT infra-structures and different input sources within a hospital with minimal ef-fort.
Met by	This requirement is supported by the OSCAR architecture and design which allow for new components to be added. More specifically: • New dashboards can be added using SharePoint • New input interfaces can be added by creating new input web services • New facts and dimensions can be added to the data warehouse and landing area	

■

11. Conclusions

This chapter summarizes the results of OSCAR and discusses possible future extensions for the OSCAR system.

11.1 Results

OSCAR was initiated to build upon the results of the previous proof-of-concept project, DACTyL, with the view to overcoming its limitations and providing an extensible solution to efficient collection and reporting on linked hospital and telehealth data.

The DACTyL solution allowed the collection of patient hospitalization data, linkage with the Motiva data at Philips and automated reporting on the Return on Investment of the Motiva service. A previous Philips project, the Motiva DWH, had already provided a data warehouse for automated reporting on Motiva data, yet it was limited with respect to the supported reporting functionality.

In light of the constraints identified, as well as the stakeholder functional and non-functional requirements, a high level system architecture was proposed and a prototype implementation was developed that accepts, links, and provides reports on hospital and Motiva data, while at the same time being extensible to new types of data input and reporting requirements. The prototype OSCAR implementation was deployed within the Philips virtualization infrastructure in order to demonstrate the proposed solution. Thus, a transition from the proof-of-concept solution of DACTyL to a prototype solution OSCAR took place successfully.

A major challenge in this project was the heterogeneity of the different HISs with which OSCAR is required to interface. We refrained from setting a strict input interface for OSCAR and expecting the HISs to conform to it; instead we designed our Data Collection upon the principles of extensibility, so that it can be adapted to support the most suitable interfaces negotiated between Philips and the hospital IT; this constitutes a considerable change in viewpoint from DACTyL. To the extent possible, we tried to make the customization process faster by proposing the use of a service oriented design for Data Collection and a model-driven approach throughout the system. We also used off-the-self data warehousing tools to further enhance this character.

Support for specific use cases, such as specific input data and output reports, is a matter of working alongside the end users (clinicians) and refining gradually their requirements. Also, the reports are constrained by the data available for input. Thus, as new data becomes available, new reports are possible, and the process is repeated.

The support of specific inputs is a matter of negotiation between the data owners (managers of the hospital data stores), the hospital IT admin (manager of the interfacing layer) and the OSCAR maintainer (manager of the OSCAR interfaces). The process has both technical aspects, but also organizational and legal aspects to it. Technical aspects include the syntactic and semantic interoperability of OSCAR and the HIS systems, while organizational aspects include the negotiations that need to take place, privacy issues, and the acquisition of patient consents for using their data.

It is evident that the above processes cannot be solved with a “simple” data warehousing software solution, due to their cutting across organizations and departments and requiring negotiations between multiple stakeholders. Nevertheless, the resulting OSCAR system supports Philips by providing the technical basis to ease and automate the technical steps involved.

11.2 Future work

Future work for OSCAR should focus on going forward with a real-life deployment in a hospital. Apart from hands-on experience with practical aspects of deployment and usage, this will also result in new use cases from the clinicians. Such a process would allow OSCAR to be refined and improved. Other aspects that need to be refined are the processes for the exchange of telehealth data and de-identified linked data exports between the hospital and Philips.

During the OSCAR project we proposed a generic architecture and design that may be potentially implemented using a variety of different tools and platforms; for example using Oracle tools instead of Microsoft or Linux as the operating system of the virtual machine instead of Windows. This genericity allows for different versions of OSCAR to be implemented and deployed in the future according to the preferences and policies of the host organization.

On a different vein, there is ongoing work within Philips on healthcare IT frameworks projects, most notably the Data Healthcare Platform, and the alignment of OSCAR with them will be important in the future as these projects offer an opportunity for OSCAR to find new use cases. In general, the landscape is changing and any future work on OSCAR should help find a place for it in this landscape.
■

12. Project Management

Whereas the previous chapters described the results of OSCAR, this one is concerned with the process of the project, those aspects that shaped the way that the project was developed.

12.1 General

In general, during the project we followed an agile process, each step being guided by the results thus far and discussions that we had with the project team. A reason for this is that developing a data warehousing and business intelligence application is an agile process as it requires collaboration with the end users in a process of gradual refinement of their requirements. In addition, due to our unfamiliarity with the data warehousing domain and the heterogeneity of the technologies involved it was hard to make accurate time estimates of the tasks during this project. Therefore, each phase of the project was planned as soon as the previous one was completed. At the end of each phase the work done was presented in the project team meetings. In general, tight deadlines were not set to allow modifications along the way whenever a better way of doing things was found.

Our tasks during the project can be roughly divided in four groups.

1. **Domain:** Study the domain, the user requirements, and previous relevant Philips projects
2. **Design:** Create the architecture, design
3. **Implementation:** Development and deployment of OSCAR prototypes
4. **Documentation:** Document and communicate the work

Tasks associated with the **domain** involved getting acquainted with the relevant domains and concepts. It was very important to investigate and build some toy applications with the various tools that could potentially be part of the final solution, such as web services with Java, Eclipse Modelling Framework applications, and Microsoft SSIS and SSAS application examples. Also, published research papers on the clinical data warehousing domain and relevant internet websites were a source of domain knowledge for us. In addition, we studied previous work done inside Philips, namely the DACTyL and Motiva DWH projects. Lastly, our discussions with the project team allowed us to obtain the required knowledge on the stakeholder requirements.

Tasks associated with the **design** included defining a suitable architecture, and designing each OSCAR component in a way that best support the system requirements.

Tasks associated with the OSCAR **implementation** involved coming up with the best tools and technologies to match the system design while supporting the system functional and non-functional requirements. The challenge with implementation was our initial unfamiliarity with the domain as well as the vast array of tools that are available.

Lastly, tasks associated with **documenting** the project included writing the final report, preparing PowerPoint presentations and presenting the project results in meetings with the stakeholders.

12.2 Process

A general timeline of the OSCAR project is shown in Figure 41.

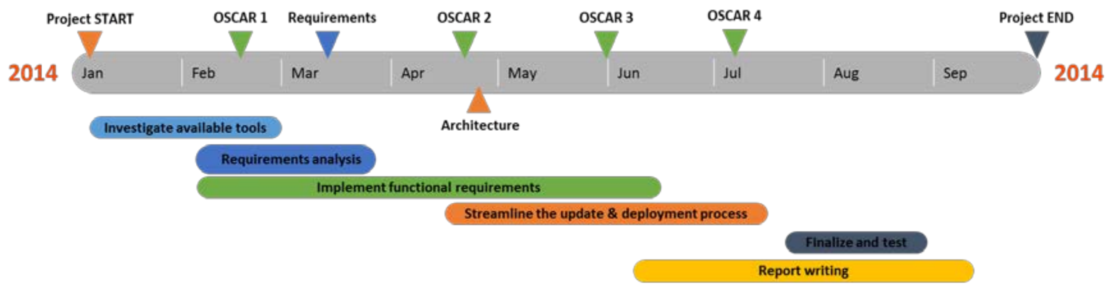


Figure 41 - OSCAR project timeline

As mentioned before, the process was agile with the results of each phase determining the following one. The main goals of each phase are outlined below, in chronological order.

In the first phase we investigated the domain and experimented with available tools and technologies. We also studied the DACTyL project and developed an initial OSCAR prototype based on the technologies of the DACTyL solution for interfacing. 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 various tools used (Microsoft BI tools). The knowledge and experience gathered from these activities contributed in the specification and decomposition of the requirements.

In the second phase we defined and refined the system requirements, based on our discussions with the project team. The requirements were based on the team's previous experience with the DACTyL project and their previous meetings with Philips and hospital stakeholders. We created a corresponding PowerPoint presentation for better communicating the requirements with the project team.

The implementation phase involved the development of OSCAR prototypes using various technology alternatives as well as the definition of the high level system architecture and inter-component designs. An important step was to re-implement the parts of the Motiva DWH project related to OSCAR, namely the Motiva data warehouse and corresponding ETL processes, from an Oracle to a Microsoft implementation. The new implementation allowed us to demonstrate the data warehousing and reporting capabilities of OSCAR ETL processes, OLAP cube, dashboards).

The next phase was involved taking a look into the system non-functional requirements, mainly the extensibility requirements and trying to support it in OSCAR in the best possible way. For instance, we implemented an improved Data Collection component, with improved auto-generation of software parts.

The last phase included writing of the final report, implementation of a final demonstration and presentations for stakeholders within Philips.

12.3 Communication

For the first two months of the project once per week we had a progress update meeting with the two company supervisors. Afterwards this was replaced with a bi-weekly meeting. In addition, there was a bi-weekly meeting with the work package team. Finally, there were informal meetings whenever it was considered necessary. During the progress update meetings the following tasks were performed:

- Report on the progress of the current deliverables
- Presentation of intermediate results-findings

The requirements were communicated by the Philips supervisors using an envisioned solution as a communication medium to explain what was required. Meeting with the hospital stakeholders were conducted before the beginning of the OSCAR project and their requirement were communicated via the Philips supervisors as well.

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:

- Presentation of the current position of the project
- Presentation of future work
- Discussion over the state of the project

■

13. Project Retrospective

This chapter presents a reflection account of the author on the course of the OSCAR project.

13.1 General

This project was both very challenging and interesting. Since we started with limited knowledge on the domain and technologies involved, at every step, we learned something new. In the meantime, we discovered that each of the domains that constitute this project is huge in its own right with many sub-domains. Achieving the right balance between research depth (research into the domain specific) and width (research into a wide range of domains) was a particular challenge for us.

13.2 Design opportunities revisited

In Chapter 3 the following design criteria were considered important for OSCAR:

- Ease of use
- Reusability (Extensibility)
- Technical realizability

We believe that the OSCAR architecture, design, and implementation support these criteria. The Microsoft tooling that we used helps the ease-of-use aspect of OSCAR because they are well-known tools specifically targeted to data warehousing solutions. Moreover, by using a modular, service-oriented approach for the data collection and providing clear WSDL interfaces the data collection component can be extended without having to depend upon the specific implementation of the current services, this supports the reusability aspect of OSCAR. By adding new web services next to the existing ones allows the whole system to be re-used even when the interfacing requirements change. The technical realizability of the system is demonstrated by the deployment of the OSCAR virtual machine within the Philips VMware infrastructure.

13.3 Challenges

A challenge came from the fact that most of the used technologies were unknown to us in the beginning of the project. Furthermore, during the course of the project we kept discovering more technologies that required research and prototyping to be evaluated.

Another challenge was the lack of specific use cases, specific in terms of the data that was required to be collected and information to be reported on, since OSCAR was expected to be a generic reporting framework. This led to vagueness in the requirements and difficulty in defining the specific goals of the project.

A last challenge was the fact that the previous projects DACTyL and Motiva DWH used technologies incompatible with OSCAR, therefore, we did not have the luxury of re-using parts of those solutions but a considerable effort was required to re-implement any required functionality. This includes the database schemas and the ETL processes that load the Motiva data to the data warehouse.

13.4 Strong Points

We identify the following strong points for our approach and results:

- Balancing the exploration of a huge domain with a concrete implementation. We believe that we singled out concepts of the domain that are relevant for this project and for the future versions while implementation a concrete solution.
- 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.

13.5 Improvement Points

We identify the following points in which we could improve:

- Defining concrete usage scenarios (specific data inputs, specific reports) in the beginning of the project and basing the rest of the work on that scenario. It would be very useful to have a concrete use case running throughout the project because it would provide a more concrete understanding to us and the stakeholders and a basis for demonstrations.
- Conducting the deployment of OSCAR in a real-life environment. It would be very interesting and useful to conduct a deployment during the course of the project in order to judge the quality of our work and refine the user interfacing and reporting requirements based on a tangible system.
- Estimating documentation effort. We underestimated the documentation effort. We could have avoided this pitfall if the documentation effort had started earlier.

■

Glossary

BI	Business Intelligence
CDA	Clinical Document Architecture
DACTyL	Data Analytics, Clinical, Telehealth, Link. The name of a project predecessor to OSCAR.
Data Model / Data Schema	A description of the data in terms of concepts, their properties, and relationships
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.
DW/DWH	Data Warehouse
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.
H2H	Hospital to Home
HIE	Hospital Interface Engine, a system used for the communication between the various systems within the hospital IT infrastructure
HIS	Hospital Information System
HL7	Health Level Seven International [24]
ICD-10	International classification of diseases version ten
Linking	The process of unifying data from multiple sources based on a common known entity
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.
OSCAR	On-Site Customer Analytics and Reporting. The name of the current project.
RDBMS	Relational Database Management System, a system that manages relational databases, offering interfaces for executing SQL queries on them.
RIM	Reference Information Model, see also **HL7**

Bibliography

[1] "Philips Research," [Online]. Available: http://www.research.philips.com/.

[2] A. Papakostopoulos, "DACTyL," Eindhoven University of Technology, 2013.

[3] K. v. H. a. K. v. Overveld, "Criteria for assessing a technological," 2010.

[5] [Online]. Available: http://technet.microsoft.com/en-us/library/ms175609%28v=sql.90%29.aspx.

[6] "Apache Axis," [Online]. Available: https://axis.apache.org/axis/.

[7] "SQL Server," [Online]. Available: http://msdn.microsoft.com/en-us/sqlserver/aa336270.aspx.

[8] [Online]. Available: http://msdn.microsoft.com/en-us/library/ms174173.aspx.

[9] [Online]. Available: http://technet.microsoft.com/en-us/library/ms141026.aspx.

[10] [Online]. Available: http://technet.microsoft.com/en-us/library/ee428287%28v=office.14%29.aspx.

[11] [Online]. Available: https://www.eclipse.org/.

[12] [Online]. Available: http://www.eclipse.org/modeling/emf/.

[14] [Online]. Available: http://en.wikipedia.org/wiki/Java_annotation.

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

[16] "Hibernate Tools," [Online]. Available: http://hibernate.org/tools/.

[17] "Apache Tomcat," [Online]. Available: http://tomcat.apache.org/.

[18] "Apache TomEE," [Online]. Available: http://tomee.apache.org/apache-tomee.html.

[19] "VMware," [Online]. Available: http://www.vmware.com/.

[20] "Vsphere feature," [Online]. Available: http://www.vmware.com/files/pdf/key_features_vsphere.pdf.

[21] "XStream," [Online]. Available: http://xstream.codehaus.org/.

[22] "Apache Velocity," [Online]. Available: http://velocity.apache.org/.

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

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

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

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

[42] Philips, [Online]. Available: http://www.philips.com/about/company/missionandvisionvaluesandstrategy/index.page.

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

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

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

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

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

[49] Oracle, "Oracle Data Integrator," [Online]. Available: http://www.oracle.com/technetwork/middleware/data-

integrator/overview/index.html.

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

[52] Apache Software Foundation, "Code Generator Wizard Guide for Eclipse Plug-in," [Online]. Available: http://axis.apache.org/axis2/java/core/tools/eclipse/wsdl2java-plugin.html.

[55] "Easily create Web services clients with Visual Studio .NET," [Online]. Available: http://www.techrepublic.com/article/easily-create-web-services-clients-with-visual-studio-net/.

[56] L. C. w. J. Stagnitto, Agile Data Warehouse Design, DecisionOne, 2012.

About the Authors

Panagiotis Thomaidis received his Diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece in 2011. During his studies he focused on Software Technology and Computational Intelligence. His diploma thesis, titled "Computational Intelligence (Transductive Support Vector Machines and applications on classification problems)", studies the robustness of a semi-supervised learning variant of the Support Vector Machine. Also, during his studies, he got involved with PANDORA, the University's robotics team, where he worked on improving the robot's SLAM subsystem, in C. Furthermore, he did an internship in ALTEC Software working with Java to create a Blog feed monitor application. He has broad interests ranging from painting and graphic design to software and artificial intelligence and is always researching new tools and languages looking for new concepts and ideas. In October 2012 he started working for the Eindhoven University of Technology as a PDEng candidate for the Stan Ackermans Institute Software Technology Program. From January 2014 until September 2014 he worked at Philips Research on the project described in this report.