

# A unified quality measure engine for the Philips HealthSuite digital platform

***Citation for published version (APA):***

Sykoudi Amanatidou, P., & Technische Universiteit Eindhoven (TUE). Stan Ackermans Instituut. Software Technology (ST) (2015). *A unified quality measure engine for the Philips HealthSuite digital platform*. [EngD Thesis]. Technische Universiteit Eindhoven.

***Document status and date:***

Published: 25/09/2015

***Document Version:***

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

***Please check the document version of this publication:***

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

[Link to publication](#)

***General rights***

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

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

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

[www.tue.nl/taverne](http://www.tue.nl/taverne)

***Take down policy***

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

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

**A Unified Quality Measure Execution Engine for the  
Philips Health Suite Digital Platform**

Pelagia Sykoudi Amanatidou  
September 2015



**A Unified Quality Measure Execution Engine for the  
Philips Health Suite Digital Platform**

Pelagia Sykoudi Amanatidou  
September 2015

# A Unified Quality Measurement Engine for the Philips Health Suite Digital Platform

Eindhoven University of Technology  
Stan Ackermans Institute / Software Technology

## Partners



Philips Research



Eindhoven University of Technology

## Steering Group

Pelagia Sykoudi  
Muhammad Asim  
Ad Aerts

## Date

September 2015

## Document status

Public

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

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 978-90-444-1414-1  
A catalogue record is available from the Eindhoven University of Technology Library  
(Eindverslagen Stan Ackermans Instituut; 2015/076)

Abstract This document conveys the results of the Unified Quality Measure Execution Engine (UQMEE) for Philips Health Suite Digital Platform (HSDP) project. The nine month project started on January 2015 and was conducted at Philips Research in the Health Informatics Solution Services department. The objective of this project is focusing on developing a tool for measuring the quality of healthcare services.

Healthcare delivery is shifting from volume (pay per visit, hospitalizations & tests) driven care to outcomes driven care. This has resulted in increased focus on quality which focuses on safe, effective, patient-centered, timely, efficient and equitable healthcare delivery. Quality measure is a quantitative tool to assess the performance of an individual or organization's performance in relation to a specified process or outcome via the measurement of an action, process, or outcome of clinical care.

This project is focusing on the implementation of a quality measure execution engine using HL7 standards i.e. Health Quality Measure Format (HQMF). HQMF is standard for concise and unambiguous representation of quality measures, hence enable interoperable execution of quality measures and benchmarking. HQMF specifies semantics, data concepts and logic for representation of data and population criteria. UQMEE is envisioned as a common asset for HSDP due to its applicability for multiple business units of Philips Health Tech.

Keywords quality measures, quality measure execution engine, HSDP, health quality measure format

Preferred reference P.Sykoudi.Amanatidou, [A Unified Quality Measure Execution Engine](#) for Philips Health Suite Digital Platform, SAI Technical Report, September 2015. (978-90-444-1414-1)

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

Disclaimer Endorsement	Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology or Philips Research. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or Philips Research, and shall not be used for advertising or product endorsement purposes.
Disclaimer Liability	While every effort will be made to ensure that the information contained within this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.
Trademarks	Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.
Copyright	Copyright © 2015. Eindhoven University of Technology. All rights reserved. No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the Eindhoven University of Technology and Philips Research.



# Foreword

Healthcare delivery is shifting from volume-driven (pay per visit, hospitalizations, and tests) care to outcome-driven care. This has resulted in an increased emphasis on quality, which focuses on safe, effective, patient-centered, timely, efficient and equitable healthcare delivery. The quality measures or indicators are quantitative tools to assess the performance of an individual or the organization's performance in relation to a specified process or outcome via the measurement of an action, process, or outcome of clinical care.

Philips Research is keen on delivering innovative solutions that leverage the quality of healthcare services in the long run. As such, introducing a way to quantify the quality of healthcare services from different healthcare providers is a matter of particular importance within the organization. In the Health Informatics Solution Services (HISS) business unit of Philips Research, we readily identify the need to promote the merit of using a standard-based solution of a unified quality measure execution engine. We appreciate the necessity of a standard-based solution, because multiple business units inside Philips are often requested to provide quality measure dashboards to their clients. As a result, we have to transfer knowledge of quality measure standards in our district. For that case, we wished for a young professional who would thoroughly investigate the relevant healthcare quality standards, so as to provide us with a standard-based engine.

Pelagia worked meticulously in order to derive a comprehensive software architecture that could fit into our specifications. Thinking out of the box, she proposed a design that welcomed changes. As a trainee for 9 months in Philips Research, Pelagia lived up to my original expectations. As a graduate of the Electrical and Computer Engineering Faculty of the Aristotle University of Thessaloniki (AUTH), I was confident about her technical background and I appreciate the inquiring mind she developed in the OOTI program.

Pelagia's work is paving the way for an extensible unified quality measure engine, which can subsequently be integrated into the Philips Health Suite Digital Platform. She cast light on the intricate complexity of the HL7 Quality Metrics standards and delivered a prototype that is able to digest any compatible HL7 HQMF documents and return the results in standard HL7 QRDA documents, thus enhancing the interoperability aspects required in her assignment. She made a valuable contribution towards better understanding the value of the healthcare quality standards and we can already see the potentials in further elaborating her engine in the future.

Muhammad Asim  
Senior Scientist  
Philips Research





# Preface

This document you are holding is the technical report of the Unified Quality Measure Execution Engine for the Philips Health Suite Digital Platform (UQMEE for HSDP). The report contains information about the requirements, the design, and the development performed in this project. The project lasted for nine months (January - September 2015) and was conducted for and within Philips Research, Eindhoven.

This document also constitutes the graduation thesis of the author, Pelagia Sykoudi Amanatidou, as part of the Stan Ackermans Institute Software Technology PDEng program of the Eindhoven University of Technology. The Professional Doctorate in Engineering (PDEng) program, also known by its Dutch name “Ontwerpers Opleiding Technische Informatica (OOTI),” is a two-year training program in which projects for various industry partners are conducted. UQMEE for HSDP is the author’s graduation project of the PDEng program.

During this nine-month project, the author worked towards gathering the stakeholders’ requirements, deriving the architecture and the design and executing the implementation. Based on the previous processes, other important aspects of the project are also applied, such as planning and documentation.

Non-technical readers should refer to Chapters 1, 2 and 3 that set the context and the goals of the project and Chapter 10, where the results are summarized. Technical readers will be interested in Chapters 5-8 that describe the system requirements, architecture, design, implementation, and deployment respectively.

September 2015



# Acknowledgements

I would like to express my deep gratitude to my company supervisor Asim Muhammad. I have learned many things since I worked with him. He spends a lot of time ensuring that I have learned correctly healthcare concepts and guiding me on managing the project. In addition, he puts effort on providing me useful feedback on my final report. I would also like to thank very much my second company supervisor, Charalampos Xanthopoulos. His help was invaluable. He introduces me possible technology choices and provides me every day useful feedbacks and motivation for the proper fulfillment of this project. Furthermore, I would like to express my gratitude to my university supervisor Ad Aerts, the General Director of the Software Technology program. He was really supportive, willing to answer to my questions during this 9-month project. He also spends a lot of time providing me a meaningful feedback on my final report.

Special thanks to all the OOTI coaches 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 take this opportunity to show my gratitude to the open source Cypress community, they provided me useful answers to unknown concepts and procedures using their tools. In addition, I would like to thank the anonymous community and open source communities of the internet provided answers to technical issues, tutorials, problems.

Last but not least important, I owe more than thanks to my family, my friends and loved ones for the encouragement throughout my life and work. Especially I would like to thank from the depth of my heart Arash Shafiei for his support and the inspiration he gave me for working and life these last two years being abroad.

September 2015



# Executive Summary

The Unified Quality Measure Execution Engine for the Philips Health Suite Digital Platform (UQMEE for HSDP) project was conducted for nine months in Philips Research, Eindhoven. It is conducted under the auspices of the Philips Health Informatics Solution Services business group. UQMEE is designed with a view to being ported on the HSDP, a unified storage platform of Philips Research health data. As a consequence, this project proposes and implements a prototype architecture which aims at giving insights to the HSDP business group.

This project attempts to develop a quality measure execution engine in an interoperable and unified way. Nowadays, due to the automation in healthcare, health data is increasing. In that way, monitoring patient data and conducting quality measures in healthcare delivery becomes more difficult. Medical practitioners need to estimate the healthcare performance in specific operations such as clinical or financial processes, outcomes or structural aspects. In order to handle the huge volume of health data a uniform and constructive way needs to be applied. For the use and exchange of health information healthcare standards are implemented by International HL7, a non-profit organization. Especially for quality measures it has developed the Health Quality Measure Format (HQMF). This standard establishes a unified form using semantics, computation logic and representation of health elements for the quality measure execution.

This project aspires the development and the execution of quality measures orchestrated by international healthcare standards. UQMEE provides interoperability to different Philips Research departments for conducting quality measures using HQMF standards. For the representation of the measure results is applied the Quality Report Document Architecture (QRDA Cat 3).

UQMEE main functionality is to transform the elements of the health standards documents into executable code. It is a web application software solution. This solution promotes the use of HL7 standards for the execution of quality measures over clinical and claims data. It proposes a modular architecture. Based on the technology choices and the design, it allows a further extensibility of HQMF functionalities and backwards compatibility of HQMF previous releases.

Philips Research aims to introduce the added value using HL7 standards in its departments performed quality measures through this project. With the advent of HSDP as central health data storage, Philips aspires the implementation of such an engine on HSDP. As a consequence, it may allow different healthcare providers to execute their quality measures and view benchmarking reports using one unified engine.



# Table of Contents

Foreword.....	vii
Preface.....	ix
Acknowledgements .....	xi
Executive Summary .....	xiii
Table of Contents .....	xv
List of Figures.....	xix
List of Tables .....	xx
<b>1. Introduction .....</b>	<b>1</b>
1.1 Context.....	1
1.2 Introductory Concepts.....	2
1.3 Project Scope.....	5
1.4 Outline.....	5
<b>2. Stakeholder Analysis .....</b>	<b>7</b>
2.1 Philips Research.....	7
2.2 Eindhoven University of Technology (TU/e).....	8
<b>3. Problem Analysis .....</b>	<b>10</b>
3.1 Philips business need.....	10
3.2 HL7 tools .....	11
3.3 Analysis of the problem.....	13
3.4 Design Opportunities .....	14
<b>4. Domain Analysis .....</b>	<b>15</b>
4.1 Introduction.....	15
4.2 Healthcare standards .....	15
4.2.1. Introduction to HL7 .....	15
4.2.2. Data Structure .....	20
4.2.3. Health Quality Measure Format .....	22
4.2.4. Quality Report Document Architecture Category III .....	29
4.3 HL7 quality measuring tools .....	33
4.3.1. Web based open source quality measure tools .....	33
4.3.2. Health-data-standard.....	33
4.3.3. HQMF2JS.....	36
4.3.4. Quality Measure Engine .....	38
4.3.5. Ruby on Rails Applications.....	38
4.4 Health data storage.....	39



4.4.1. Relational Databases.....	39
4.4.2. Document Schema-less databases .....	40
<b>5. Requirements .....</b>	<b>42</b>
5.1 Introduction.....	42
5.2 User requirements .....	42
5.3 Functional requirements .....	43
5.4 Non-Functional requirements .....	44
<b>6. System Architecture .....</b>	<b>45</b>
6.1 UQMEE for HSDP overview.....	45
6.2 Quality Measure Interface.....	47
6.3 Quality Measure Engine.....	50
6.4 Data model .....	52
<b>7. System Design .....</b>	<b>56</b>
7.1 QME design introduction.....	56
7.1.1. HQMF to Query transformation .....	57
7.1.2. HQMF_Parser – HQMF_Document.....	58
7.1.3. Mapper2Schema .....	61
7.1.4. Query Generator .....	61
7.1.5. Data_Criteria_2Query .....	61
7.1.6. Population_Query .....	62
<b>8. Implementation.....</b>	<b>64</b>
8.1 UQMEE Implementation.....	64
8.2 Technology choices for implementation of UQMEE components... 64	
8.2.1. QMI and QME on Ruby on Rails framework .....	64
8.2.2. Parsing HQMF and QRDA libraries.....	66
8.2.3. Relational data management system and other alternatives .....	66
8.2.4. Data.....	67
8.2.5. QME – Object Relational Mapping.....	68
8.3 Conclusions .....	68
<b>9. Verification &amp; Validation .....</b>	<b>69</b>
9.1 Introduction.....	69
9.2 Verification.....	69
9.2.1. Verification of the executed measures.....	69
9.3 Validation.....	71
<b>10. Conclusions.....</b>	<b>73</b>
10.1 Results.....	73
10.1.1. Results based on the requirements.....	73
10.1.2. Conclusions .....	74
10.2 Future work .....	76
<b>11. Project Management.....</b>	<b>77</b>

11.1	<i>Introduction</i> .....	77
11.2	<i>Project time line</i> .....	77
11.3	<i>Communication</i> .....	78
<b>12.</b>	<b>Project Retrospective</b> .....	<b>80</b>
12.1	<i>Introduction</i> .....	80
12.2	<i>Design opportunities revisited</i> .....	80
12.3	<i>Strong Points</i> .....	80
12.4	<i>Improvements Points</i> .....	81
	<b>Appendix A</b> .....	<b>82</b>
	<b>Appendix B</b> .....	<b>88</b>
	<b>Glossary</b> .....	<b>95</b>
	<b>Bibliography</b> .....	<b>96</b>
	<b>About the Author</b> .....	<b>97</b>



# List of Figures

Figure 1: Example usage of the HSDP storage space from different business groups .....	10
Figure 2: UQMEE on top of HSDP.....	10
Figure 3: Clinical Quality Measures roadmap.....	12
Figure 4:HL7 RIM primary subjects .....	17
Figure 5: Example of 5 vendor systems with 5 interfaces.....	18
Figure 6: HL7 RIM .....	19
Figure 7: XML format.....	21
Figure 8: JSON format .....	21
Figure 9: HQMF high level document structure .....	23
Figure 10: QRDA category III overview.....	30
Figure 11: QRDA Category III document report .....	32
Figure 12: Activity diagram of HQMF parser of Health Data Standards.....	33
Figure 13: HQMF R2.1 class model, based on RIM model .....	35
Figure 14: HQMF2JS components.....	37
Figure 15: Patient records in QRDA category I format.....	37
Figure 16: Ruby on Rails logo .....	38
Figure 17: Map-Reduce process.....	41
Figure 18: UQMEE for HSDP as a black box.....	45
Figure 19: UQMEE for HSDP .....	45
Figure 20: UQMEE for HSDP overview diagram.....	46
Figure 21: QMI and QME sub systems.....	46
Figure 22: Quality measurement process .....	47
Figure 23: MVC .....	47
Figure 24: MVC package diagram of QMI .....	48
Figure 25: Flow process of QMI .....	49
Figure 26: Class diagram of QMI.....	49
Figure 27: Sequence diagram from the moment the user selects an indicator until the moment the corresponding HQMF is sent to QME .....	50
Figure 28: Sequence diagram from the moment the QRDA is received till the moment the results are shown to QM_Interface.....	50
Figure 29: QME package diagram .....	51
Figure 30: QME flow process overview.....	51
Figure 31: QME class diagram.....	52
Figure 32: Example of HQMF data elements' relations .....	53
Figure 33: Data model of UQMEE for HSDP.....	54
Figure 34: Package model of Engine.....	56
Figure 35: Highlighting the Measure Execution Engine process .....	56
Figure 36: Relation between HQMF information to SQL-based queries .....	57
Figure 37: Class diagram of Engine .....	58
Figure 38: HQMF_Document class diagram.....	60
Figure 39: Example mapping between data criteria and query .....	61
Figure 40: In order tree-traversal.....	62
Figure 41: Sequence diagram of query execution .....	63
Figure 42: Relation between QMI and Ruby on Rails components .....	65
Figure 43: QME component using schema less database.....	67
Figure 44: Verification of executed measures.....	69
Figure 45: UI of QMI.....	73
Figure 46: Project time line .....	78
Figure 47: Flow process of queries execution .....	88

# List of Tables

Table 1: Example of Avendis Donabedian's model .....	2
Table 2: Interface number based on system number .....	18
Table 3: Population Types .....	27
Table 4: Measure Types .....	27
Table 5: Logic Operator types.....	27
Table 6: User Requirements.....	42
Table 7: Functional Requirements for UQMEE for HSDP.....	43
Table 8: Non-Functional Requirements for UQMEE for HSDP.....	44
Table 9: Indicators .....	54
Table 10: Data criteria elements examples of their query form .....	57
Table 11: Example of a continuous variable measure calculation .....	58
Table 12: Data criteria elements .....	59
Table 13: Example Data Criteria elements .....	59
Table 14: Population criteria elements.....	59
Table 15: Example Population Criteria elements.....	60
Table 16: Data criteria - SQL elements mappings .....	61
Table 17: Technologies used to realize each component .....	64
Table 18: Population types.....	70
Table 19: Condition checks of executed measures .....	70
Table 20: HQMF-QRDA possible states .....	71
Table 21: Indicator titles .....	71
Table 22: Indicators and their corresponding HQMF elements .....	72
Table 23: User Requirements.....	73
Table 24: Functional Requirements .....	74
Table 25: Non-functional requirements .....	74
Table 26: Temporal relations types.....	82
Table 27: Value element .....	83
Table 28: Attributes of Value element .....	83
Table 29: Arel-ORM functions .....	84
Table 30: HQMF elements.....	85
Table 31: Comparison operators' mappings .....	86
Table 32: Aggregation operators' mappings.....	86
Table 33: Logical operators' mappings.....	87
Table 34: Value elements mappings .....	87

# 1. Introduction

This chapter presents the main theme of this project. In this overview, some introductory concepts are provided, as well as the project scope.

## 1.1 Context

This project is defined in the domain of the quality of healthcare delivery. As good health is essential for people's life, the quality of healthcare delivery is equally important. Insight into this quality is necessary to know for which aspects of healthcare delivery the quality is good, and for which it needs to improve in order to achieve good health.

Usually, an insight is built on information. In order to get insights into the quality of healthcare delivery, it is necessary to monitor and evaluate actions that are taken in order to deliver healthcare. By monitoring, we mean that data about healthcare delivery and outcomes need to be recorded. In that way, it could be used to compare care delivery to patients or to examine the state of healthcare procedures. By evaluating, we mean that measures need to be properly defined. Defined measures could be used for a specific aspect of quality (e.g. medications prescribed to a heart-failure patient on hospital discharge). As a result, it is evaluated whether the delivery of healthcare meets the guidelines. It is crucial to measure quality, because without measuring we cannot improve it, as Lord Kelvin said.

The definition of measures for evaluating the healthcare delivery is promoted by the healthcare community. The healthcare community consists of non-profit, public or private organizations. These organizations are not only focusing on healthcare equipment development, but they are also providing procedures, such as developing guidelines ensuring that the proper treatment is applied to the right patients. These organizations have also attempted to establish standardized measures that allow the objective scoring of the performance of healthcare practitioners, procedures and resources.

Nowadays, due to the progress of technology in healthcare an increasing level of automation of support services exists. The healthcare community requires more health data and faster processes for the analysis of their processes and healthcare delivery. Since many measurements and examinations are taken for each patient in different health departments, the amount of health data is huge. As a consequence, the evaluation of the quality measures on healthcare data needs to be automated too.

Medical practitioners need the entire picture of patients' healthcare delivery to obtain a correct overview of their clinical state. This requires the combination of many different sources of healthcare data. Many of these sources are autonomous and not yet compliant with common standards. This leads to data integration problems. To get reliable and accurate values for the measures, the data has to be complete and consistent. This has led to another effort by the healthcare community to establish standards for the exchange of healthcare data between providers.

Philips Research is one of the members of the healthcare community. Since 1895, Philips has contributed in healthcare to its high-end medical procedures in hospitals by means of their advanced equipment, such as scanners, as well as to services for home care. This fact establishes Philips as one of the healthcare organizations who have obtained a very good overview on the healthcare delivery actions.

As Philips gained experience in healthcare, it is able to study related issues. For instance, Philips could study the issues that arise when health data of different sources

is combined and measures are evaluated on the basis of it. One of the issues is to assess whether the combined data used is sufficiently coherent and able to provide reliable outcomes, in case a measure is applied to it.

To get a deeper insight in this issue, Philips Research has initiated a project that focuses on the design of a quality measure execution engine. Medical practitioners need quality measures in order to get insights in the aspects of healthcare delivery i.e. process, structure and outcomes. This tool consists of an engine that can execute any quality measure defined according to the HL7 standard. Such standard definition of quality measures also allows for benchmarking different healthcare providers. The output of the tool is qualitative analysis reports upon the healthcare data of a provider. Such a tool thus supports the improvement of both measures and data and may contribute to obtaining more reliable insights.

## 1.2 *Introductory Concepts*

In this section, we elaborate on the types of measures for quality measurements and how they are applied, as well as why data consistency has a crucial role in the healthcare domain and how we can tackle interoperability issues. To clarify these aspects, some introductory information is provided which is grouped in four subsections trying to answer the following questions.

1. What are the proper measures to assess healthcare quality?
2. How can we apply the measures in healthcare delivery?
3. What is the added value of health data consistency?
4. How can health data be made interoperable?

### *1. What are the proper measures to assess healthcare quality?*

Common sense assumes a good quality of healthcare delivery when the resulting health of patients is good. However, this is a subjective statement since “good health” has different meaning to different people. There have been several attempts to objectify the way to measure quality. A proper quality measure needs to be objectively defined and backed by evidence.

In addition, it is important to know what aspect a quality measure is focusing on and how it relates to the outcome, which is the end goal to measure, but not always possible to measure. One of these attempts derives from Avendis Donabedian [1], who contributed to the study of quality in healthcare. Avendis established the Avendis Donabedian’s model which is divided into three categories:

- Structural measures, which represent how care is organized
- Process measures, which focus on what was done
- Outcome measures, which show what has actually happened to patients

Some examples of these categories are presented in the Table 1.

**Table 1: Example of Avendis Donabedian's model**

<b>Structural Measure</b>	<b>Process Measure</b>	<b>Outcome Measure</b>
Number of medical practitioners who have systems to track diabetes patients	Percentage of patients with diabetes who have had an annual eye exam during the last year	Percentage of diabetes patients who are blind or have compromised vision

The measures discussed here are often driven by the clinical practice guidelines [2]. Clinical practice guidelines set a list of “Evidence-based” recommendations for medical practitioners about the care of patients with specific clinical conditions. “Evidence-based” implies that the recommendation has been formulated as a conclusion of an un-biased process of clinical research with findings of the highest quality and it is trying to assist in patients’ care.

For instance, there is a clinical practice guideline which recommends that for obese patients it is good to follow a special healthy diet during their stay in the hospital. This guideline promotes the definition of a quality measure which assesses the process that patients with obesity indeed follow such a healthy diet.

## 2. *How can we apply the measures in healthcare delivery?*

Quality measures are used for measuring the quality of healthcare delivery. Quality measures can be also called indicators, because they indicate whether the healthcare delivery meets the desired goals.

An indicator is often calculated as a ratio or proportion. A ratio exhibits the relative size between two populations. It is required that the one is the subset of the other. The resulting ratio lies between 0 and 1 and it can be converted into a percentage by multiplying by 100.

For better understanding, an example of an indicator is given below [3] :

Clinical research has established that in case a patient has had a heart attack, taking aspirin every day reduces the chance of having a second one. Based on this, a clinical guideline has been established that recommends that physicians prescribe aspirin to all patients who have had heart attack after discharge.

Due to this guideline, a quality analyst can apply a question to a hospital, such as: how many heart attack patients were actually prescribed aspirin after their discharge. In that way, an analyst is trying to establish if physicians’ decisions are according to the guideline for the proper patients’ population.

However, not all the patients who have had heart attack receive aspirin, because some of them are allergic to aspirin and others left the hospital without receiving aspirin prescription.

In order to quantify the compliance with the guideline in this example, we define:

- Let  $P_{denom}$  be the number of patients who had heart attack
- Let  $P_{num}$  be the number of heart attack patients who were prescribed aspirin after discharge

Then, if we have a population of 100 heart attack patients ( $P_{denom}$ ), of whom 96 were prescribed aspirin after discharge ( $P_{num}$ ), we get:

$$P_{num} : P_{denom} = 0.96 \quad \text{Eq 1.1}$$

This number of heart attack patients represents the number of patients who actually receive the aspirin prescription. Not all the patients who have had a heart attack have received aspirin, because some of them may be allergic to aspirin or other patients left the hospital without receiving an aspirin prescription. The resulting percentage 96% indicates for the specific hospital that most physicians prescribe aspirin to the proper heart attack patients after discharge.

To determine whether the result reveals a good quality in this hospital, we should also obtain statistical information about the expected results for this clinical condition



in the specific hospital. For instance, if in this specific hospital it is estimated that 5% of the patients are often allergic to aspirin and they are not going to accept the medication, then the expected percentage of patients who have had heart attack and receive aspirin after discharge is 95%. In this case, the actual result reveals that the healthcare quality in this hospital meets the expectations.

### 3. *What is the added value of health data uniformity?*

By data uniformity we mean data that is structured in the same form, with the same semantics and concepts. When data is uniformed complying with the same standard, then all of it can take part in standardized quality measurements. As a result the more data participates in measurements, the more accurate results are provided. This fact highlights the importance of data uniformity and especially in healthcare where the quality is related to patients' lives.

Specifically, health data should have the same format, following the same semantics and structure. For example, let's define the measurement of obese patients in a set of health data by using the condition "Patients.Diagnosis = 'Obesity'." This measure can be computed on all health data that includes fields such as "Patients," "Diagnosis," and for this particular case has the state of diagnosis reported as "Obesity." In case the health data does not include this kind of fields, Patients and Diagnosis, it is excluded from this measurement. As a result, the used data in the measurement is not complete as not all data could participate in the computation. However, in case the health data contains this kind of fields, but the context is different, like "Cancer" instead of "Obesity", the quality measurement is executable giving 0 results.

Furthermore, including more health data in quality measurements brings more precise results, because the health information is richer. For instance, collecting health data of patients from different sources can reveal more information about previous medications and problems. This information can classify patients into correct groups and provide a better consideration about their clinical conditions.

The above facts reveal the high added value of data uniformity in quality measurements, so that the overview is complete and the value of measures' outcome has worth.

### 4. *How can health data be made interoperable?*

Health data interoperability can be strengthened by the establishment of international standards for all healthcare information. If each healthcare provider in the world complies with these standards, then health data can be transferred efficiently, as well as better studies and analysis on health data can be achieved.

A non-profit organization which contributes to health data interoperability is the Health Level-7 International (HL7). This organization provides a complete framework and related standards for the exchange, integration, sharing, and retrieval of health information in an electronic format. It comprises the HL7 standard development body, by which also standards are produced for interoperable exchange of clinical information in the form of XML documents.

HL7 has developed two important standards for the interoperable representation of quality measurement and the result of executing a quality measurement. These standards are the "Health Quality Measure Format (HQM)" and the "Quality Reporting Data Architecture (QRDA)."

HQM is used for defining the content and the structure of an indicator. In other words, HQM is used for representing indicators. This standard provides the necessary specifications for defining the structure, metadata, definitions, and logic of an

indicator. An indicator which is encoded according to HQMF is called an “eMeasure.”

QRDA is used to define the representation of quality measurement results which are used for analysis in a reporting data architecture document. The resulting format can be a percentage, ratio, or a pair of continuous variables depending on the measure type defined in the HQMF.

There are three categories of QRDA. QRDA Category 1 represents the results of an indicator that are relevant to one patient. QRDA Category 2 represents a summary of aggregate data across a defined population. The report may or may not identify patient specific data. QRDA Category 3 represents aggregated results computed for a population, so it is not patient specific. The first and the third specifications are the most used.

Taking everything into consideration, as Philips Research is able to study these aspects, it is working on the development of a quality measurement tool. This tool would be able to use HL7 standards such as HQMF and QRDA Cat 3 providing interoperability. It would be able to execute different indicators on the top of clinical and claims data. Finally, it will provide performance results to care providers, so they can take the proper actions in order to improve the quality of healthcare delivery.

### **1.3 Project Scope**

This project has nine-month duration and its main goal is the development of a quality measurement tool upon a set of health data providing quality analysis reports.

This project would provide insights into the architecture and implementation to a Philips business group that is responsible for the development of the Health Suite Digital Platform (HSDP). So far, the used health data is not applied to any other external system. In the future, in Philips Research, health data will be provided by the HSDP. However, in this project the HSDP was not the provider of such data.

The scope is to design a software solution, which is able to:

1. Provide a User Interface (UI) to quality analysts, in which they can select a specific indicator
2. Provide indicators using HL7 standards, like the HQMF
3. Provide a quality measure execution engine based on the HQMF standards
4. Provide the results of the HQMF according to the QRDA Category 3 format
5. Show the results for analysis using graphs on the UI

Out of scope on this project are the following:

1. This software solution is not required to be integrated into HSDP
2. This software solution is not required to measure quality in any arbitrary set of health data
3. The UI in this phase does not need to provide support for composing new indicators

### **1.4 Outline**

**Chapter 2** presents the stakeholders and explains their interest and contribution to this project.

**Chapter 3** describes the problem that this project is trying to solve.

**Chapter 4** introduces technologies that are used in this project

**Chapter 5** describes the issues and possible risks that this project may meet.

**Chapter 6, 7, 8** describe the functional and non-functional requirements of the project, as well as the system design and the architecture, and a detailed description of its implementation.

**Chapter 9** documents how the application is validated.

**Chapter 10** outlines possible future work and further considerations.

**Chapter 11** documents the project management of the project.

**Chapter 12** is a reflection of the author on the project and the methodology.

■

## 2. Stakeholder Analysis

This chapter provides a short overview of the stakeholder parties, their role and their expectations from this project.

### 2.1 *Philips Research*

There are three main stakeholders that have varying interests in this project, the Health Informatics Solutions Services (HISS) group, the Hospital to Home business group (H2H), and the Health Suite Digital Platform (HSDP). We introduce each of them, as well as their goals and their requirements.

#### **Healthcare Informatics Solutions Services (HISS) group**

The objective of HISS is to optimize patient care by presenting the right information at the right time to the right people. HISS supports users across the healthcare system by managing and analyzing clinical or claims data.

The end users of such a measuring tool could be for instance the Chief Financial Officers (CFOs), the Chief Executive Officers (CEOs), the Accountable Care Organizations (ACOs), and care management staff. One expected added value of this project is that it enables Philips customers, such as ACOs, to report the quality of their performance. In addition, it can be used as a means for consulting care providers achieving healthcare improvement. Furthermore, it can be used to provide insights from the aggregated claims or clinical data, so the cost and the utilization of the resources could be adapted to patients' needs. For instance, the quality reporting performance can show that the average visiting time in an emergency department (ED) is high. As a result, it is indicated to care providers to improve the ED healthcare delivery taking actions such as increasing the available resources.

The main technical feature that this project is expected to show is the usage of HL7 standards by introducing a coherent quality measuring tool. Various Philips business groups can apply this tool, such as TASY, EMR, HTS and H2H, providing performance reporting for a set of indicators.

#### **Hospital to Home (H2H) group**

The H2H group objective is to leverage the efficiency of the process of telehealth clinical programs. Telehealth is a domain where a big variety of clinical programs have been developed applying distant monitoring to patients by collecting their health data. Quality measurements can be applied to this data to provide insights in performance. The health data that has been collected in the monitoring process are consistent with HL7 standards.

The current project aims at introducing to this business group the value of using HL7 for measuring quality.

#### **The Health Suite Digital Platform (HSDP) group**

The HSDP is a platform aiming at consolidating the Philips telehealth services. It is expected to receive clinical and telehealth data from external providers, with the view to generating comprehensive reports and identifying patterns for improving the efficiency of various clinical programs.

The HSDP aspires to serve different business units with same needs for quality measurements. The establishment of a common quality measurement tool in this

platform is valuable, as it can be applied within different business units. Each unit could apply its individual indicators, and all units should follow the same structure compliant with the HL7 standards. The insights gathered by this project, which are related to the implementation, would be also helpful to the HSDP group.

### **Preventive Medicine**

Another associated project to Philips Research which is focusing on the improvement of health population is the Preventive Medicine, a project for a Brazilian healthcare provider called Unity [4]. This is a project that provides several consultation programs and activities to patients who are willing to prevent further symptoms of diseases, such as diabetes or obesity. Indicators of clinical and claims data can be used. These indicators are applied to patients' health data showing their performance during a period in which they follow a specific activity program.

This project can be of benefit to the Preventive Medicine project in two ways: 1) due to the need of handling a huge number of health data, it is helpful to use HL7 standards for indicators representation in order to have a scalable solution, and 2) they may use the project's software solution to visualize their results.

Specific stakeholders from Philips are the following:

- Muhammad Asim, Senior Scientist and HL7 Specialist and Designer (Company Supervisor)

His main requirement is the correct usage of the HL7 standards, as exemplified by the quality measure engine that has been tested on several clinical and claims data sets.

- Charalampos Xanthopoulos PDEng - Senior Software Designer (Company Supervisor)

His main requirement is that the software solution can demonstrate a modular, extensible architecture and sound technology choices, as manifested by the object oriented techniques.

- Steffen Pauws – Senior Scientist (Project Manager of the H2H Analytics project)

His main requirement is to obtain a software prototype which exposes the value of using the HL7 standards on performance reporting, as demonstrated by the Philips Research's needs.

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

The Eindhoven University of Technology is responsible for the educational aspect of this project. The TU/e's interest in this project is seeing to it that it tackles a sufficiently complex problem. The PDEng trainee carrying out the project should prove herself to be capable of solving the problem by providing a high quality design for its solution and validating this solution.

The representative stakeholder from the TU/e is Ad Aerts, the Program Director of the Software Technology PDEng program, who serves the role of university supervi-

sor. The university supervisor should ensure that the design and documentation meet the standards of a PDEng project.

A final stakeholder of this project is the author of this report. The author's ambition is to prove that she is able to derive a high quality design which will ensure her graduation from the PDEng program.

The author's role is to carry out and manage the project which includes deriving the design and validating it, as well as managing the project tasks, such as planning, risk management, and negotiating about which requirements to include in the project.

■

# 3. Problem Analysis

This chapter describes the problem that this project attempts to solve. It explains the problem in Philips Research that leads to the need for developing the UQMEE. Then it continues on presenting the existing tools applied for health standards and their limitations in order to meet Philips Research needs. It concludes by the definition of the actual needs of Philips Research based on its available resources.

## 3.1 Philips business need

It is expected that all Philips Research business groups will eventually use the HSDP. It is planned that different business groups can have a central storage space of their data. In case these groups need to apply quality measurements on their data, it makes sense to have a unified and standard solution. Philips Research wishes to introduce a common asset of the platform. It is also expected to provide a standard-based engine, so all different business groups can execute HQMF eMeasures. This concept of HSDP as a central storage is illustrated in the following Figure 1.

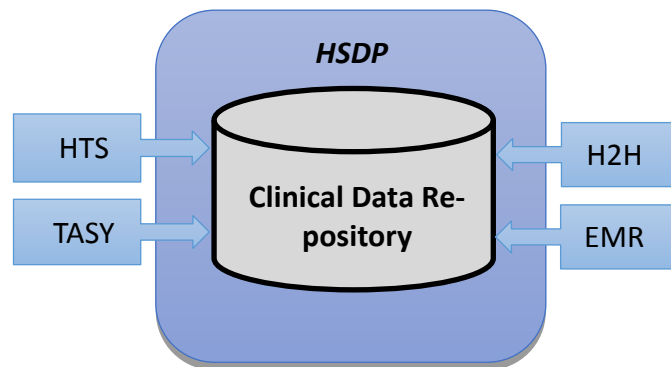


Figure 1: Example usage of the HSDP storage space from different business groups

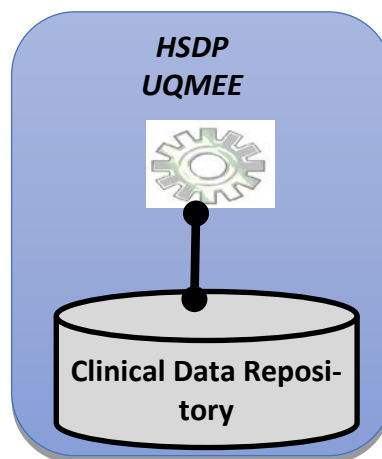


Figure 2: UQMEE on top of HSDP

Different business groups may benefit from the usage of a standard-based engine, as they can access the data uniformly and they are able to handle its growth. In addition, in case of a collaboration and data exchange between business groups, like the H2H and EMR groups, the data scalability is significantly improved.

So far, each business group uses its own technology for data analysis. This means that they use their own individual format of data structure. They also use individual data storage systems to analyze specific key performance on their own data. This has caused an issue when it is necessary to combine health data from different sources or when it is required to define a basis for performance analysis. As HL7 standards are also based on international clinical practical guidelines, they give a reliable means to measure quality.

For instance, in chapter 2 we mentioned the Preventive Medicine project. This project consists of specific key performance indicators and patient health data of an individual care provider called Unity [4]. This provider has specific needs and interests in health data analysis. Formulating the used performance indicators by means of HL7 standards improve their analysis of results. HL7 standards enhance the usage of indicators, because the applied key performance indicators derive from international quality models and clinical guidelines. In that way, quality measurements are more reliable. Care providers' results may be also presented internationally giving an overview on its performance to international healthcare quality.

In general, the usage of HL7 standards in quality measurement exhibits two positive side effects. The first is that the collaboration of different business groups is enhanced. The second effect is that the scalability of their quality measures is based on HL7 standards, which provides consistency on their measurements.

### **3.2**      ***HL7 tools***

The building of a quality measurement engine based on HL7 can be materialized by various tools.

HL7 joined forces with the National Quality Forum [5], a non-profit membership-based organization which works for healthcare improvement. In year 2009, they published the Health Quality Measure Format standard, an HL7 derived model for quality measure definition in electronic format.

Figure 3 shows the relation between the National Quality Forum and the International HL7 standards.

The National Quality Forum coordinates the authoring of quality indicators, using the Measuring Authoring Tool, (MAT) and publishes a number of valid quality measures in electronic format. The indicators or “eMeasures” use the HL7. HL7 also publishes the Quality Reporting Data Architecture which consists of QRDA Category 1, a data structure for patient data and QRDA Category 3, a data structure reporting the results of HQMF eMeasures.



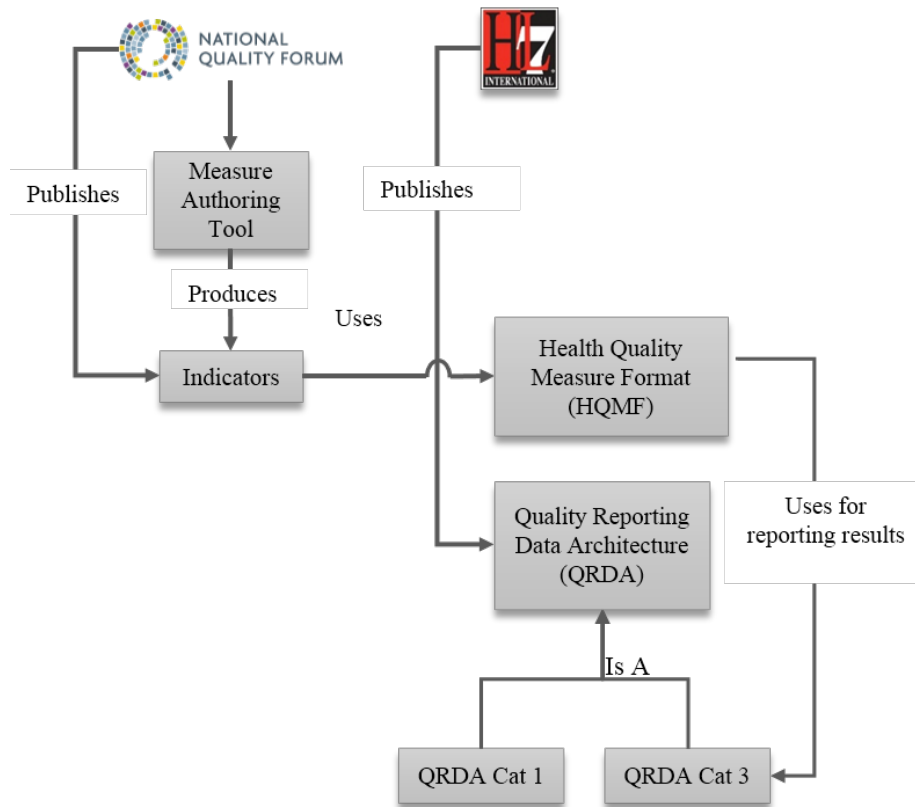


Figure 3: Clinical Quality Measures roadmap

In addition, MITRE [6] is a non-profit US corporation that conducts research and development centers sponsored by the federal government. Its attempt is to explore and develop better healthcare solutions contributing in quality healthcare measurements.

This organization provides clinical medical solutions and open source software tools for enhancing the usage of healthcare standards. Using HL7 standards, MITRE provides open source libraries, such as the Health Data Standards set, which is a set of libraries for supporting the generation and consumption of a variety of standards such as the HITSP C32, the QRDA and the HQMF.

These libraries are commonly used in a set of web-based open source measuring tools, such as Cypress and popHealth.

**Cypress:** Cypress is a testing tool of Electronic Health Records (EHRs) and EHR modules in calculating Meaningful Use (MU) Stage 2 Clinical Quality Measures (CQMs). It is open source and freely available for use by the health IT community.

**popHealth:** popHealth is an open source reference implementation software service that automates the reporting of quality measures. Its objective is to simplify the reporting of summary quality measures, and streamline the exchange of summary quality data.

These tools are a reference point to the current project. Their exploration is deemed appropriate, so as to leverage the knowledge gained, as well as to get inspired for building new tools.

### 3.3 *Analysis of the problem*

The direct usage of MITRE libraries yields technology limitations. These limitations stem from the choice of database technologies, the design of web-based applications and their design priorities such as performance.

On the other hand, Philips Research needs to introduce the use of HL7 standards with a further goal to obtain a quality measuring tool on top of the HSDP. However, the HSDP is not available yet, so there are no explicit limitations on the technology choice, as well as definite specification of the available interfaces.

The main purpose of this project could be defined as follows:

**Problem statement:** “The design and development of a standard-based quality measurement tool, using HL7 standards, providing scalable quality measurements in interoperable health data.”

To develop such a tool it is also imperative the investigation of the already available tools and the study of their possible use. The possible limitations that these tools may bring could be:

- Specific technology choices for web-application solutions
- Dependency on their data base technologies
- Dependency on their design choices based on their priorities, like performance

On the other hand, the needs of Philips business groups are:

- Exploration on the usage of the available technologies and promotion of the HL7 standards added value
- Exploration of the suitable database technologies
- Need for new indicators in Philips Research business groups
- Need for an architecture which will be the input to the HSDP business group

Regarding the above statements, the project entitled “A Unified Quality Measurement Engine for the Health Suite Digital Platform.”

It is called:

1. “Unified,” because it uses HL7 standards providing scalability and consistency in business groups’ measures.
2. “Quality Measurement Engine,” because its goal is to measure the quality of healthcare delivery in terms of executing quality eMeasures on clinical and claims data.
3. “For the Health Suite Digital Platform,” because we aspire that this engine would be part of the HSDP and it can be used by different business groups to develop quality measurement dashboards for their customers.

UQMEE for HSDP is a web-based application, which makes use of open source libraries.

### 3.4 *Design Opportunities*

In this project, the following design opportunities are described, which aim at the enchantment of the quality of this project.

**Ease of use:** This concerns two dimensions of the project. The one concerns user-friendliness of the application. The end-user may reform quality measurements through a user friendly UI. The other dimension concerns the elegance of the design.

**Backward Compatibility:** This concerns the compatibility to previous releases of the HQMF standards. The usefulness of the solution depends on its capacity of being able to accept previous releases of the standards.

**Extensibility:** This concerns the ability of the solution to be easily extensible for a new functionality that HL7 standards may bring for the measurement in the future.

■

# 4. Domain Analysis

This chapter contains the analysis of the domains that are related to UQMEE for HSDP. This analysis gives insights in the used technologies for the design and implementation of this project.

## 4.1 Introduction

The technology of the Unified Quality Measurement Engine for HSDP covers the following areas:

1. Healthcare standards
2. HL7 tools
3. Health data

These areas need to be analyzed in order to comprehend the proposed solution.

Healthcare standards are designed to allow healthcare providers and systems to exchange information by relying on a common set of concepts. These standards provide a set of functionalities on transferring and documenting health data. The healthcare standard information is usually stored in files using special representation languages. These languages are defined through data structure formats, like XML or JSON. An international healthcare standards organization is the Health Level 7 (HL7). HL7 is a U.S.-based, American National Standards Institute (ANSI)-accredited health information standards development organization.

Already existing HL7 quality measuring tools are also described in the following section. Specifically, Cypress is described, a testing measuring tool which uses HQMF indicators on a test health data set. This solution is a web application which depends on technologies such as the Ruby on Rails framework. .

In addition, the investigation on the available data storage technologies is necessary.

## 4.2 Healthcare standards

### 4.2.1. Introduction to HL7

HL7 International is an ANSI [5] organization focusing on healthcare standards development. This kind of organization produces standards which are called specifications or protocols oriented to various healthcare domains such as pharmacy, medical devices and claims processing. Specifically, the HL7 target is to standardize clinical and administrative data. Its name derives from the seven levels of the Open Systems Interconnection (OSI) model [7], the Level Seven is the Application model.

The objective of HL7 is to provide interoperable standards to improve healthcare delivery in terms of its workflow and its transferring knowledge between different providers, such as private institutes, vendors and agencies. Its vision is to establish the best and most widely used standards in healthcare.

In general, interoperability gives many advantages to business systems. There are three main aspects of interoperability which make it important, the technical, the semantic and the process. By technical, we mean the support of moving data from

one system to another. By semantic, we mean that the different systems are able to understand the data in the same way. By process, we mean that various business processes can work together.

The target groups applied the HL7 standard are the following:

- Clinical and Public Health Laboratories
- Clinical Decision Support Systems Vendors
- EHR, PHR Vendors and Health Care IT Vendors
- Lab Vendors
- Healthcare Institutions (hospitals, long term care, home care, mental health)
- Stand-alone family health history applications, family history sections in personal health records, or family history modules within electronic health records
- Primary Care and Specialty Physicians

There are two particular versions of HL7 standards, Version 2 and Version 3.

The HL7 Version 2 (V2) Messaging Standard-Application Protocol for Electronic Data Exchange in Healthcare Environments- supports the data exchange in healthcare and so far is the most widely implemented healthcare standards worldwide. Especially, HL7 Version 2.6 attempts to enhance the unified format for messaging between different healthcare information systems by adding new messages and domains to provide syntactic interoperability.

The HL7 Version 3 is an improved version of V2, which strives to improve the V2 process and its outcomes providing a fully specified standard. Some of the capabilities that HL3 V3 brings are the following:

- Top-down message development with emphasis to the reusability of multiple contexts and semantic interoperability
- Representation of complex relationships
- Formalisms for vocabulary support
- Solving re-use and interoperability across multiple domain contexts
- A uniform set of models
- Expanded scope to include community medicine, epidemiology, veterinary medicine, clinical genomics, and security

The HL7 V3 development model is enhanced by the use of a well-defined information model the Reference Information Model (RIM). RIM is an object model which represents clinical data and identifies the life cycle of events that messages can carry. It determines a robust set of data types and a complete set of healthcare domain terminology. RIM is essential for HL7 standards development, since it increases precision and reduces implementation cost.

Figure 4 shows the primary subject areas of the HL7 RIM.

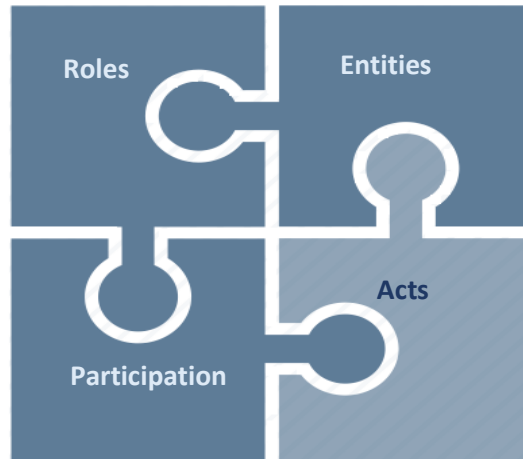


Figure 4:HL7 RIM primary subjects

In order to analyze the importance of the HL7 RIM for interoperability issues, it is vital to describe the interoperability aspects in detail, as well as the specific contribution of the HL7 RIM.

Interoperability is the ability of two or more systems or components to exchange information and reuse the information that has been exchanged. This definition presents the functional and semantic interoperability. Functional interoperability implies the capability to reliably exchange information, and semantic interoperability implies the ability to interpret and make effective use of the exchanged information. By “effective use,” we mean that the transferred information can be applied to any type of computable algorithm which is appropriate to this health data type. In other words, it is underlying that if the semantic interoperability depends on standard models, then more systems which also apply these models are able to reuse the information.

The HL7 V3 RIM supports the extension of semantic interoperability across all types of clinical and related information systems such as

- Care provider systems which support electronic health records
- Healthcare delivery
- Patient administration
- Patient finance
- Clinical decision support
- Clinical research

For highlighting the role of the RIM, the following example is presented [1].

Back in 80s, when distributed healthcare application systems were developed the number of their interfaces was increasing rapidly. For instance, two systems need one interface, three systems need three interfaces and four systems need six interfaces for exchanging information. Based on the formula of “number of combinations of n elements taken r at a time” Eq 4.1, an estimation of the required interfaces given by the number of systems is presented to Table 2.

$$n!/(n-r)! r! \quad \text{Eq 4.1}$$

$$\text{For } r = 2 \text{ and arbitrary Eq 4.1 gives } \frac{n(n-1)}{2} \quad \text{Eq 4.2}$$

Table 2: Interface number based on system number

Systems:	Interfaces:
<b>3</b>	3
<b>4</b>	6
<b>5</b>	10
<b>10</b>	45
<b>20</b>	190
<b>30</b>	435
<b>50</b>	1225

However, this number of interfaces cannot be maintainable. Large U.S. organizations like Mayo Fdn or Duke and national systems in Europe obtain typically between 50-100 such interfaces. The cost of 50-100k USD per custom interface indicates as cheaper solution to apply an interface standard. This reduces the number of interfaces for n systems to the cost of (n-1) interfaces.

In case each system complies to a standard interface, then 6 systems requires 6 interface, 30 systems requires 30 and 50 systems require 50, this results bring maintainable systems.

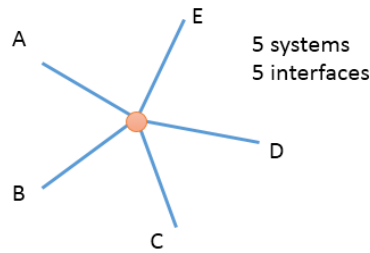


Figure 5: Example of 5 vendor systems with 5 interfaces

This actually means that any change that vendor “C” makes, Figure 5, their internal lab data structures and vocabulary are mapped into a common semantic structure. Systems A, B, D and E also map the standard-defined semantic lab structures into their internal lab data structures. In other words, interfacing means mapping to/from standard semantic structures.

Moreover, at a particular healthcare site, which consists of systems A1, B1, C1, D1, and E1, systems can be developed based on a local lab standard or reference information model. In case this site needs to interoperate with other sites, there needs to be an overall lab reference model that each site can map its information.

In conclusion, HL7 RIM can tackle this need and provide a mature version of a common reference healthcare applications information model.

Regarding Figure 4, a more detailed representation of HL7 RIM is shown in Figure 6. The blue colored classes are the core classes and the orange colored shows the first level of subclasses. Since RIM attempts to be a reference model that encloses the entire healthcare domain, its knowledge can be useful for understanding any healthcare application.

The following are examples of actions: a request or an order for a medical test is an action; the report of the test result is an action, creating a diagnosis based on test results and prescribing treatment based on the diagnosis. In simple terms, a medical record is a record of each individual action of the diagnosis, the treatment and the care of a patient.

More details for RIM concepts are depicted through the following:

- Each event in healthcare is called an Act, such as procedures, observations, medications, supply or registration.
- Acts are related through an Act Relationship, such as composition, preconditions, and revisions.
- Participations define the context of an Act, such as author, performer, subject or location.
- The participants are Roles, such as patient, provider, practitioner and employee.
- Roles are played by Entities, and subtypes of them can be persons, organizations, materials, places or devices.
- Role Link represents relationships between individual roles

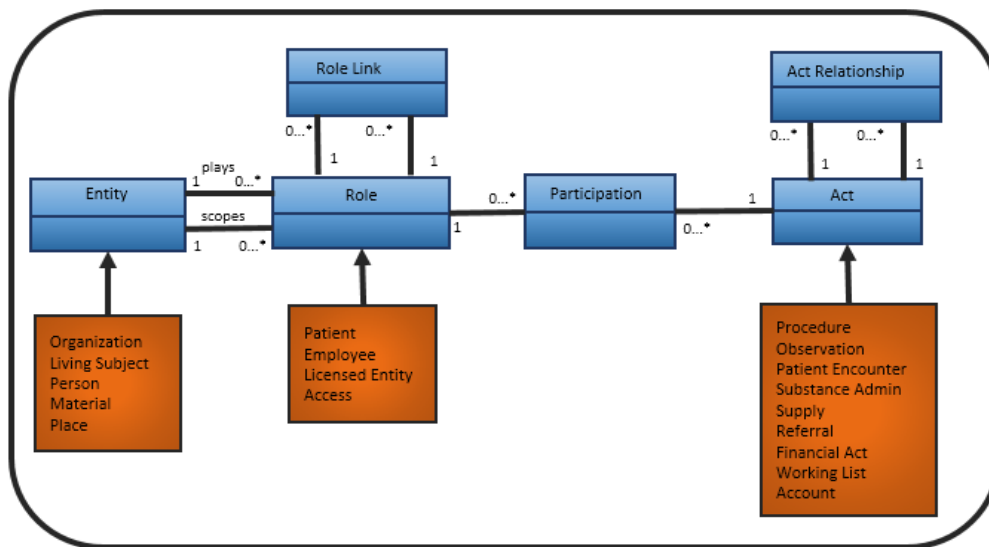


Figure 6: HL7 RIM

From the classes of Figure 6 only Act, Entity and Role represents the exact class or concept intended. In HL7 representation, a sub-type is addition to the RIM in case it requires one specific attribute which cannot be inherited by its parent classes. Classes that express unique concepts without the need of further attributes are represented with distinct codes. Accordingly, the following coded attributes serve to further define the concept being modeled:

- **classCode (in Act, Entity and Role):** represents the exact class intention
- **moodCode (in Act) and determinerCode (in Entity):** represents an attribute which defines whether the class represents an instance or a kind of Act or Entity
- **code (in Act, Entity and Role):** represents a particular classified type within a particular classCode

The rest RIM classes, Participation, ActRelationship and Role Link represents various concepts such as different forms of participation or different kinds of relationships between acts. These concepts are represented by **typeCode** attribute.

Specifically, the specific state of an action is described by “mood” code. The “mood” code specifies whether the Act is an activity that has happened, can happen, is happening, is intended to happen, or is requested to happen.



An act mood code could be:

Definition (DEF):	Definition of an Act
Intent (INT):	An intention to plan or perform an act
Request (RQO):	A request or order for a service
Promise (PRMS):	Intent to perform
Confirmation (CNF):	Promise that has been solicited via an order
Event (EVN):	An Act that actually happened

For instance, considering an Act which defines the occurrence of “Room Cleaning,” the related mood codes could be defined as:

<b>Mood code</b>	<b>Statement</b>
Proposal (PRP)	Why don't you clean your room?
Order/Request (RQO)	Clean your room!
Promise (PRMS)	I will clean my room tomorrow!
Event (EVN)	Room is cleaned.

Moreover, Acts happen at a specific time which is defined by the Act.effectiveTime. Act's effective time is a time expression specifying the operative time of the Act, the primary time for which the Act holds and the time of interest from the perspective of the Act's intention.

Furthermore, some of the Act relationships could be defined as following:

<b>Type</b>	<b>Definition</b>
COMP	Has component
PERT	Has pertinent info
RSON	Has reason
INST	Instantiates
PRCN	Has precondition
OCCR	Occurrence

As Entity plays a particular Role, Participation expresses the context for an act, in terms of who perform it, for whom it was done or where it was done. For example, Joe Smith plays the role of the doctor. The doctor role participates in an act that can be the ordering in which doctor is participating as author of an order. Some types of participations can be author (AUT), data entry person (ENT), call back contact (CBC), patient subject (PATSBJ), admitter (ADM), discharger (DIS), location (LOC), consultant (CON), device (DEV) and responsible provider (RESPROV).

Taking everything into consideration, HL7 has defined a large amount of data types; some of them are basic data types such as Booleans, but they can also specify encapsulated, coded, numerical or quantity data types.

HL7 documents are stored as data structures in xml or JSOM files. The next section provides a reference to data structures. In latter chapters, on the basis of HL7 Reference Information Model, also other models are described, such as the Health Quality Measure Format (HQMF) and Quality Report Document Architecture (QRDA).

#### 4.2.2. Data Structure

##### 1. Extensible Markup Language

XML data structure stands for Extensible Markup Language [8]. Markup languages are used for the representation and processing of a text. Special codes define special

formatting styles for text layout and style. The code which is used for specifying the formatting is called tags. An example of Markup Language is HTML.

The data of XML document is modeled into a linearization tree structure. This means that several character strings are placed in each node in the tree. The tree structure and the character strings together form the information content of a XML document. Some of the characters in the document support the linearization and others consist the information content.

Moreover, XML can be used in a variety of different contexts transferring data. It is also used in Web Services sending requests back and forth without human interaction.

## 2. JavaScript Object Notation

JSON stands for JavaScript Object Notation. It is a standard that uses human readable text for transferring data objects. Data objects are structured as key-value pairs. JSON is derived from the JavaScript scripting language, but it does not depend to any programming language specification [9].

JSON is promoted as a lightweight alternative version of XML. Both of these formats have support of creating, reading and decoding data. In addition, other examples of data structure format could include OGD, YAML and CSV.

In the following part, Figure 7 and Figure 8, there is an example of XML and its corresponding JSON script format.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

Figure 8: JSON format

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>
      <type>home</type>
      <number>212 555-1234</number>
    </phoneNumber>
    <phoneNumber>
      <type>fax</type>
      <number>646 555-4567</number>
    </phoneNumber>
  </phoneNumbers>
  <gender>
    <type>male</type>
  </gender>
</person>
```

Figure 7: XML format

### 4.2.3. Health Quality Measure Format

The HL7 Health Quality Measure Format document was first initiated in 2010 and so far it spans several releases. HQMF R1 was the first attempt to represent internationally the clinical quality measure metadata, data elements and logic. As HQMF R1 has complex structure, then HQMF R2 was established to enhance its use. Its latest release is HQMF R2.1, issued in 2014, which structures the HQMF R2, into discrete modules such as metadata layer, data layer and expression layer [10].

In this section, we describe the basic concepts and the use of the latest release HQMF R2.1.

#### 1. Use of HQMF standards

The HQMF represents a health quality measure in electronic format. HQMF measures derive from clinical guidelines and its purpose is to evaluate the performance of healthcare delivery by the comparison of the actual actions in healthcare to the expected ones. The HQMF standard provides also a consistent and a formulated way for querying patients' data.

#### 2. General Concepts

Figure 9 presents a high level introduction to the HQMF concepts is presented. And the following XML snippet shows the high level XML structure of the HQMF standard.

```
<!-- Start of an HQMF R2 eMeasure. An eMeasure is enveloped by the QualityMeasureDocument element. -->
<QualityMeasureDocument>
  <!-- Header attributes including Title, Narrative, Author, Custodian etc. -->
  <templateId />
  <title />
  <text />
  <author />
  <custodian />
  <verifier />

  <!-- defining the time period that this eMeasure applies to -->
  <controlVariable>
    <measurePeriod />
  </controlVariable>
  <!-- Miscellaneous metadata for an eMeasure -->
  <subjectOf>
    <measureAttribute />
  </subjectOf>

  <!-- Sections -->
  <!-- Measure Description Section -->
  <component>
    <measureDescriptionSection />
  </component>

  <!-- Data Criteria Section, containing actCriteria, etc. -->
  <component>
    <dataCriteriaSection />
  </component>

  <!-- Population Criteria Section containing an Initial Population, numeratorCriteria, denominatorCriteria
  exclusions, exceptions, stratifier Criteria etc. -->
  <component>
    <populationCriteriaSection />
  </component>
</QualityMeasureDocument>
```

```

<!-- Measure Observation Section containing expression language expressions for evaluation using Data
Criteria-->
  <component>
    <measureObservationsSection />

  </component>
</QualityMeasureDocument>
<!--end of eMeasure -->

```

In the above XML snippet, with blue color we indicate the introductory document concepts for the eMeasure, such as its id, the author, the measure period of the HQMF and its metadata. The red color presents the information of data criteria, population criteria and sections related to the measurement of the HQMF.

Specifically, the header contains information about the measurement such as name, author, description, id, references and measure type, and the body contains all the necessary sections which participated in the quality measure computation. The body also contains a human readable text, which is used for the HTML representation of the eMeasure.

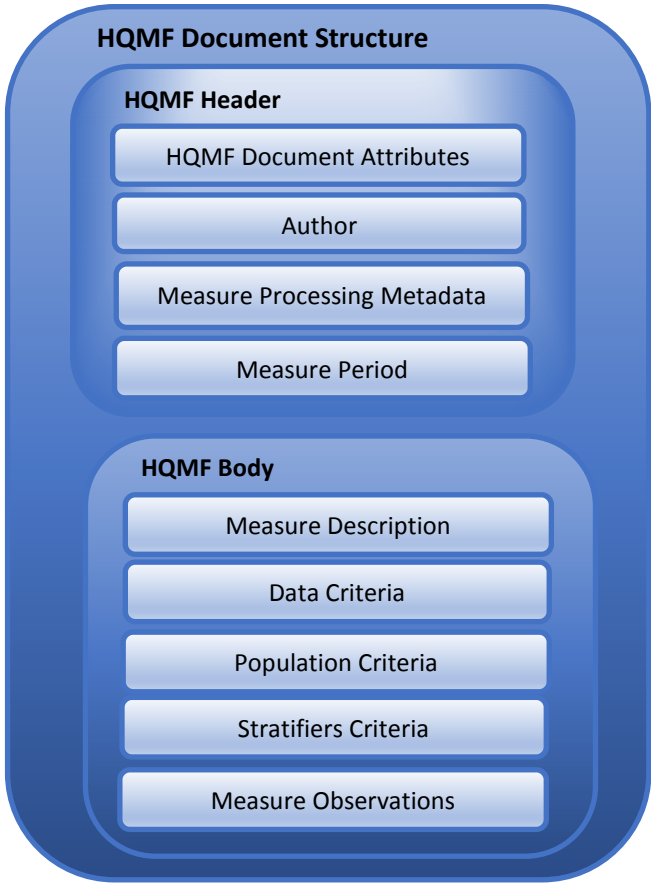


Figure 9: HQMF high level document structure

3. *Measure Period*

Every HQMF document has a Measure Period. The Measure Period is a required element for the proper definition of a quality measure. The Measure Period desig-

notes the time period in which the measured data are determined, collected and inspected in this HQMF document.

The measure period is defined inside a control variable element, as this variable controls the time period that the eMeasure is computed.

The Measure period class code is defined as observation (OBS), its mood code is event (EVN), and its unique code is defined as MSRTP from the HL7 ActCode vocabulary.

The measure period value could be defined by interval properties such as low, high, and width values. The following XML snippet shows an example of measure period definition.

```
<!-- measure period is 2011-01-01 to 2011-12-31 -->
<controlVariable>
  <measurePeriod>
    <code code="MSRTP" codeSystem="2.16.840.1.113883.5.4">
      <displayName value="Measurement period"/>
    </code>
    <value xsi:type="IVL_TS" highClosed="true" lowClosed="true">
      <low value="20110101"/>
      <high value="20111231"/>
    </value>
  </measurePeriod>
</controlVariable>
```

#### 4. Data Criteria Section

This is the core of the HQMF standard. Data criteria determine the specific content that the HQMF document includes. To illustrate the usage of data criteria section the definition of a criterion should be given:

*“A criterion is a condition or a set of conditions that can be evaluated as TRUE or FALSE for a given item.”*

Especially, the data criteria of a quality measure identify a set of conditions that define if a data item is included in the measured population or not.

For example:

- Patient’s age is greater than 18 years old
- Patient’s diagnosis is set equal to Obesity

The types of data criteria vary over a large list of healthcare information. Some of them are defined as:

- Patient Demographics
- Encounters
- Medications
- Lab Results
- Vital Signs
- Problems
- Procedures
- Allergies
- Immunizations

In addition, data criteria can express a nested time constraints to a patient population number. This means that data criteria can be related to other data criteria or time constrains through time relationships. To illustrate this kind of relationships, the following examples are presented.

1. The patient had a blood measure lab test, one year before the measure period
2. The patient has an encounter during the measure period having a required type of medication

Another important thing for the data criteria is the order they are applied. A different order can give a different result.

Typically the data criteria section may include a definition section and several entries for each data criterion.

### Data criteria definition attributes

The definition element in data criteria section is not required for the proper definition of an eMeasure. However, definitions may be used as a link between the data criteria and the derived implementation data model such as the corresponding tables in healthcare storage. For instance, if healthcare data model consists of tables such as encounters or procedures, then the definition can be defined based on this information, determining the general concepts in which data criteria are used. The definitions may contain act, encounter, observation, procedure or supply.

The ability to specify the related implementation model allows the translation of the measure into a computable form. It is only required to uniquely refer the specific data model element to the particular corresponding information inside the data criteria.

In case of an act definition, the eMeasure should contain attributes like class code which is set "ACT", and the mood code is set "DEF". In addition, an id uniquely identifies the definition within the data criteria section, and it can be used to a data criterion as a reference to its definition. The following XML snippet presents an example encounter definition.

```

<definition typeCode = "INST">
  <encounterDefinition classCode = "ENC" moodCode = "DEF">
    <id root="12345678" extension="encounter"/>
  </encounterDefinition>
</definition>
```

In the above snippet, the following attributes are defined:

- **typeCode**: instantiates the object
- **classCode**: defines the type of object, in this example is an encounter
- **moodCode**: defines the definition of this object
- **id-root**: identifies the specific id of this object
- **extension**: identifies the name of the data type

### Data criteria entry attributes

Inside an entry a criterion can be defined. In this entry a local variable name is set with the unique name of the criterion. This name is also used to refer the particular criterion to other parts of the document like the population section.

It is aforementioned that the RIM objects are trying to cover conceptually all the elements of healthcare delivery, For instance, Acts can include observations such as lab result, blood pressure or infection. In the eMeasure for an Act criterion the attribute "isCriterionInd" indicates that the specific act may have occurred and attributes like code identifies the specific information that the criteria presents.

The following example presents a valid entry of a data criterion for a completed weight observation.

```
<localVariableName value = "Observation_weight">
<observationCriteria classCode="OBS" moodCode="EVN" isCriterionInd="true">
  <id root="f92aa450-73c0-11de-8a39-0800200c9a66"/>
  <code code="27113001" codeSystem="2.16.840.1.113883.6.96"
    codeSystemName="SNOMED CT">
    <displayName value="weight"/>
  </code>
  <statusCode code="completed"/>
</observationCriteria>
```

Regarding the previous snippet, we can see that the observation criterion for weight is presented. The observation criterion consists of a unique id and code elements which determine the type of data. Usually, data criteria are defined using codes that derive from the National Medicine Library. This library includes a huge amount of codes which represents diseases, procedures, patients' characteristics and medications. In this particular example, it is defined the code which represents the weight. The code for weight is set using the code system SNOMED CT. Many code systems are also used like LOINC and RXNORM.

The definition of data criteria usually contains more complex logic and relationships, as a data criterion may depend on other criteria or time constrains. The following XML snippet presents a criterion which includes temporally related information which indicates that the criterion takes place during the measure period of the eMeasure.

```
<entry typeCode="DRIV">
  <localVariableName value="EDEncounter" />
  <encounterCriteria classCode="ENC" moodCode="EVN" isCriterionInd="true">
    <code code="2.16.840.1.113883.3.117.1.7.1.292" codeSystem="2.16.840.1.113883.3.560.101.1">
      <displayName value="Emergency Department Visit SNOMED-CT Value Set" />
    </code>
    <title value="Occurrence A of Encounter, Performed: Emergency Department Visit" />
    <statusCode code="completed"/>
    <participation typeCode="LOC">
      <role classCode="SDLOC"></role>
    </participation>
    <temporallyRelatedInformation typeCode="DURING">
      <criteriaReference classCode="OBS" moodCode="EVN">
        <id root="2.16.840.1.113883.3.100.100.123" extension="MeasurePeriod"/>
      </criteriaReference>
    </temporallyRelatedInformation>
  </encounterCriteria>
</entry>
```

### 5. Population Criteria Section

The population criteria section identifies a population using one or more data criteria elements. The elements that this section contains are based on the type of quality measures which could be:

- Proportion - Ratio: the result of a fraction, between 0 and 1
- Continuous Variable: a number which results an average, summary, a median or any other computational statement
- Cohort: a number representing the requested group

Table 3 presents the different population types in HQMF.

Table 3: Population Types

Population Types	Rules
<b>IPP</b>	The initial patients' population that is included in the measure
<b>DENOM</b>	The denominator can be the same as the IPP in case there is an additional data criteria, it is added to the IPP
<b>NUMER</b>	The numerator population is a sub set of the DENOM
<b>NUMER Exclusions</b>	The numerator exclusion population is a subset of NUMER which has to be removed
<b>DENOM Exceptions</b>	In case the NUMER is resulted empty, then the DENOM Exceptions take place this is a subset of the DENOM
<b>DENOM Exclusions</b>	The denominator exclusions is a subset of DENOM which has to be removed from DENOM
<b>MSRPOPL</b>	The measured population is the same as the IPP, including additional conditions and continuous variables
<b>MSRPOPLEX</b>	The measured population exclusion is a sub set of MSRPOPL, which has to be removed from MSRPOPL

For each measure type, the corresponding population types are defined in Table 4.

Table 4: Measure Types

Measure Type	Population types
<b>Proportion</b>	IPP, DENOM, NUMER, DENOM Exclusions, DENOM Exceptions, NUMER Exclusions
<b>Ratio</b>	IPP, DENOM, NUMER, DENOM Exclusions, NUMER Exclusions
<b>Continuous Variable</b>	IPP, MSRPOPL, MSRPOPLEX
<b>Cohort</b>	IPP

Population criteria are constructed using data criteria. In the population criteria, data criteria are connected to each other by logical operations like AND, OR, NOT or XOR.

The type of logical operators between the data criteria have the following format shown in Table 5.

Table 5: Logic Operator types

Type	Logic Operator	Logic Statement
<b>AllTrue</b>	AND	A AND B
<b>AllFalse</b>	NOR = NOT(OR)	NOT (A OR B)
<b>AtLeastOneTrue</b>	OR	A OR B
<b>AtLeastOneFalse</b>	NAND = NOT(AND)	NOT(A AND B)
<b>OnlyOneTrue</b>	XOR	(A OR B)AND(NOT(A AND B))
<b>OnlyOneFalse</b>	XNOR	(A OR B)AND(NOT(A AND B))

To illustrate the usage of logical operators an example is given.

To identify the Initial Population that consists of information related to patients' characteristics such as male patients between the ages of 16-74, two data criteria



elements are used. The first contains information about the patient date of birth and the second contains conditions related to patients' gender.

- Data Criteria Element 1: "Patient is between the ages of 16-74"
- Data Criteria Element 2: "Patient is male"
- Inclusion of both criteria is done using the "AllTrue" operator, which is a logical AND, "AllTrue" {Data Criteria Element 1, Data Criteria Element 1 }

The next XML snippet shows the structure of this example criteria included to initial population. It is shown that the initial population criterion has its unique id, and code which derives from HL7 ActCode. Each reference to a criterion and the use of logical operations are defined within precondition attributes.

```

<initialPopulationCriteria classCode="OBS" moodCode="EVN" isCriterionInd="true">
  <id root="c75181d0-73eb-11de-8a39-0800200c9a66"/>
  <code code="IPOP" codeSystem="2.16.840.1.113883.5.4" codeSystemName="HL7 ActCode">
    <displayName value="Included in Initial Population"/>
  </code>
  <precondition>
    <allTrue>
      <id root="9ea5d63f-f794-4b5d-ae33-e1091cb31d38"/>
      <precondition typeCode="PRCN">
        <!-- Age 16-74 years-->
        <criteriaReference classCode="OBS" moodCode="EVN">
          <id root="f92aa450-73c0-11de-8a39-0800200c9a66"/>
        </criteriaReference>
      </precondition>
      <precondition typeCode="PRCN">
        <!-- Gender male -->
        <criteriaReference classCode="OBS" moodCode="EVN">
          <id root="42e2aef0-73c4-11de-8a39-0800200c9a66"/>
        </criteriaReference>
      </precondition>
    </allTrue>
  </precondition>
</initialPopulationCriteria>

```

Moreover, population criteria can included deep nested logical operations, for instance an AllTrue operation can contain AtLeastOneTrue conditions.

## 6. Stratifiers

Stratifiers are constructed using Data Criteria and determine the way the population criteria should be grouped. In a quality measure computation, strata are usually demographic information of patients like age, gender, ethnicity, race and payer.

Stratifier criterion can also have logic operations, as it can be defined by various category combinations. The following stratifier example captures patients who are less than 50 years old or female.

```

<stratifierCriteria>
  <id root="aebb3a81-74da-21de-7a23-0800200c9a65"/>
  <precondition>
    <atLeastOneTrue>
      <precondition>
        <!-- Less Than 50 years of Age -->
          <criteriaReference classCode="OBS" moodCode="EVN">
            <id root="aebb3a51-73da-11de-8a39-0800200c9a55"/>
          </criteriaReference>
        </precondition>
      <precondition>
        <!-- Female Gender -->
          <criteriaReference classCode="OBS" moodCode="EVN">
            <id root="aebb8a52-73da-11de-8a39-0300200c9a11"/>
          </criteriaReference>
        </precondition>
      </atLeastOneTrue>
    </precondition>
  </stratifierCriteria>

```

## 7. Continuous variable eMeasures

In case of continuous variable eMeasures additional attributes are defined in HQMF document. Inside the section `measureObservationDefinition`, information related to the applied computation of the measure score is set. Its code should be defined as “AGGREGATE”. Specific type methods identify the computations over data.

To clarify the usage of `measureObservationDefinition` the following example XML snippet is provided.

```

<!-- the resultant measure observation, defining the CV calculation -->
<measureObservationDefinition classCode="OBS" moodCode="DEF">
  <code code="AGGREGATE" codeSystem="2.16.840.1.113883.5.4">
  </code>
  <value xsi:type="PQ">
    <expression value="EDEncounter.Participation.Role.effectiveTime.high -
      EdEncounter.Participation.Role.effectiveTime.low"/>
  </value>
  <methodCode>
    <item code="MEDIAN" codeSystem="2.16.840.1.113883.5.84" />
  </methodCode>
  <component>
    <criteriaReference>
      <id root="c75181d0-73eb-11de-8a39-0800200c9a66"
        extension="measurePopulation"/>
    </criteriaReference>
  </component>
</measureObservationDefinition>

```

In this measure observation definition, we calculate the median visiting time of an emergency department. In that case, it is necessary to subtract the discharge (high) time from admission (low) time of each emergency department encounter. Then the calculated value should take place to the median measurement method.

Additional type methods are count, sum, average, standard variance, minimum, maximum, median and mode.

All in all, HQMF provides a wide range of computational capabilities, and helps standard developers to use healthcare information effectively.

### 4.2.4. Quality Report Document Architecture Category III

The Quality Report Document Architecture Category III is a template healthcare standard document which is used to report the results of indicators which are com-

puted by HQMF documents [11]. The QRDA category III document consists of the following parts:

- CDA Header, Clinical Document Architecture
- Reporting Parameters
- Measure Section

In the first part, the header, there is the document id, the date that the document was created, the author and other information related to the document information.

In the Reporting Parameters section, there is information about the reporting period and in the Measure Section, there is information about the quality measures, their number id, title and their version.

In the report body, the results of each population are presented and for each population type the corresponding strata results. In the next figure, an example of the form of the QRDA Cat III report is presented, after the Measure Section, the measure results are presented, in this case only for the Initial Population.



Figure 10: QRDA category III overview

In more details, the count results of each aggregate group contain:

- Measure data
- Reporting stratum
- Race supplemental data element
- Ethnicity supplemental data element
- Sex supplemental data element
- Payer supplemental data element

Measure data implies to the computed population type such as initial population, denominator, and numerator. For this particular population, the group results are documented which are specified in strata in HQMF. Finally, the counts related to race, ethnicity, sex and payer are exhibits.

The next Figure 11 is a micrograph of a QRDA document format.

<b>Document id:</b> <b>Document created:</b> <b>Performer:</b> <b>Author:</b> <b>HER Certification Number:</b> <b>Legal authenticator:</b> <b>Document maintained by:</b>	B467c6c7f623 May 13, 2012 Person id, Organization id SOME Data Aggregator Transform Tool dev Medical record, device 1a2b3c Good Health Hospital signed at August 11, 2012 Good Health Hospital
<p><b>Table of Contents</b></p> <ul style="list-style-type: none"> <li>• Reporting parameters</li> <li>• Measure Section</li> </ul> <p><b><u>Reporting Parameters</u></b></p> <ul style="list-style-type: none"> <li>• Reporting period: 01 January 2012-31 March 2012</li> <li>• First encounter: 05 January 2012</li> <li>• Last encounter: 24 March 2012</li> </ul> <p><b><u>Measure Section</u></b></p>	
<b>eMeasure Title</b>	<b>eMeasure Version Number</b>
Anticoagulation Therapy for Atrial Fibrillation/Flutter	1
<ul style="list-style-type: none"> <li>• <b>Performance Rate:</b> 80% (Predicted 62%)</li> <li>• <b>Initial Population:</b> 1000           <ul style="list-style-type: none"> <li>• <b>Male:</b> 400</li> <li>• <b>Female:</b> 600</li> <li>• <b>Black:</b> 300</li> <li>• <b>White:</b> 400</li> <li>• <b>Asian:</b> 300</li> </ul> </li> <li>• <b>Denominator:</b> 500           <ul style="list-style-type: none"> <li>• <b>Male:</b> 200</li> <li>• <b>Female:</b> 300</li> <li>• <b>Black:</b> 100</li> <li>• <b>White:</b> 150</li> <li>• <b>Asian:</b> 250</li> </ul> </li> <li>• <b>Numerator:</b> 400           <ul style="list-style-type: none"> <li>• <b>Male:</b> 170</li> <li>• <b>Female:</b> 230</li> <li>• <b>Black:</b> 80</li> <li>• <b>White:</b> 120</li> <li>• <b>Asian:</b> 200</li> </ul> </li> </ul>	

Figure 11: QRDA Category III document report

## 4.3 HL7 quality measuring tools

### 4.3.1. Web based open source quality measure tools

This section gives an overview of several open source projects and libraries which are used for generating, parsing, and calculating HQMF queries against tested patient records [12].

### 4.3.2. Health-data-standard

The Health-Data-Standard is an open source library for generating and consuming several standardized healthcare related data formats. It includes libraries for parsing HQMF documents and dealing with National Medical Library value sets.

In projects such as Cypress, which is a testing tool for quality measurements, the need to parse the HQMF document leads to the use of the HQMF parser by the Health-data-standard library. The following Figure 12 shows the activity diagram parsing a HQMF file.

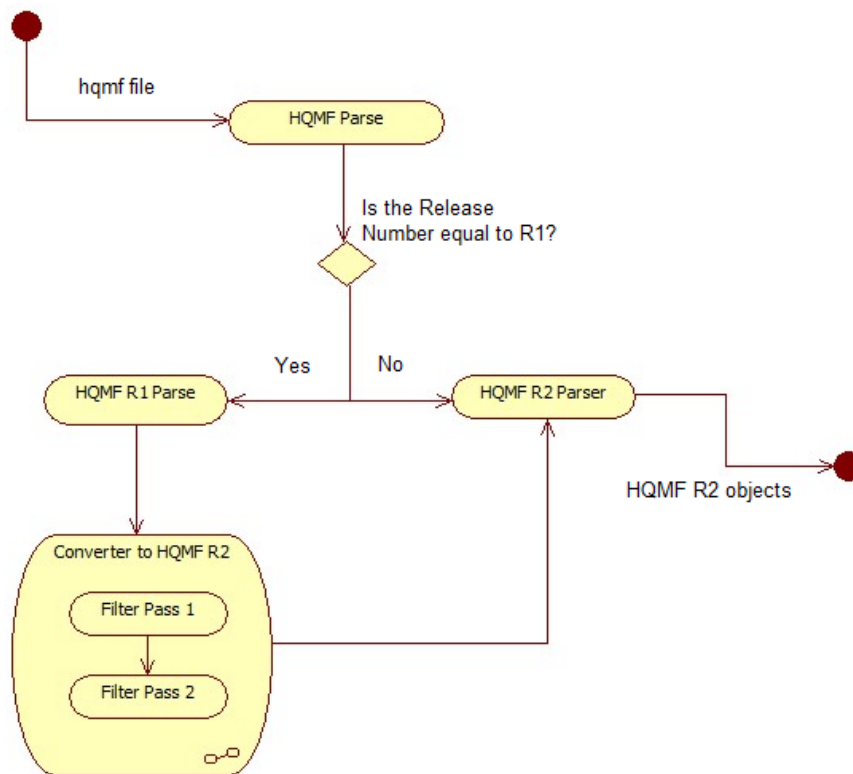


Figure 12: Activity diagram of HQMF parser of Health Data Standards

The HQMF Parser and converter are used by the Meaningful of Use stage 2, which publishes measures defined using HQMF R1. HQMF R1 is a complicated format with a deeply nested structure. The HQMF R2 was developed to address the complexity of the HQMF R1 standard. The internal model of the HQMF in Cypress is based on HQMF R2 RIM. Figure 13 presents the HQMF-model in HQMF R2 RIM model. All HQMF files in xml format should be translated in this internal form. The Parser supports files of HQMF R1, R2 and R2.1 releases.

All files should be aligned to this internal HQMF-model. In case the input to Parser is a HQMF R1 file, it is parsed and it is converted to the internal HQMF R2 by passing through a two-pass filter. Actually, the converter consists of two passes into the HQMF R2 format. The first pass converts most of elements like data criteria and population criteria, and the second pass applies some changes regarding comparisons between some kinds of population criteria, such as an observation criteria. In the end, the parser outputs a JSON file, which is actually a representation of the HQMF objects in one document.

The HQMF-model is used as a reference model in which every HQMF document should be aligned. The parser can search and extract the required elements from the HQMF xml file and map it to the HQMF corresponding element that the HQMF model represents.

The next sections describe more open source projects and libraries which are used to create queries for various technologies. One of these projects is Cypress, an open source testing measuring tool based on Mongo DB and its map-reduce framework. The Cypress data is stored as patients' oriented documents.

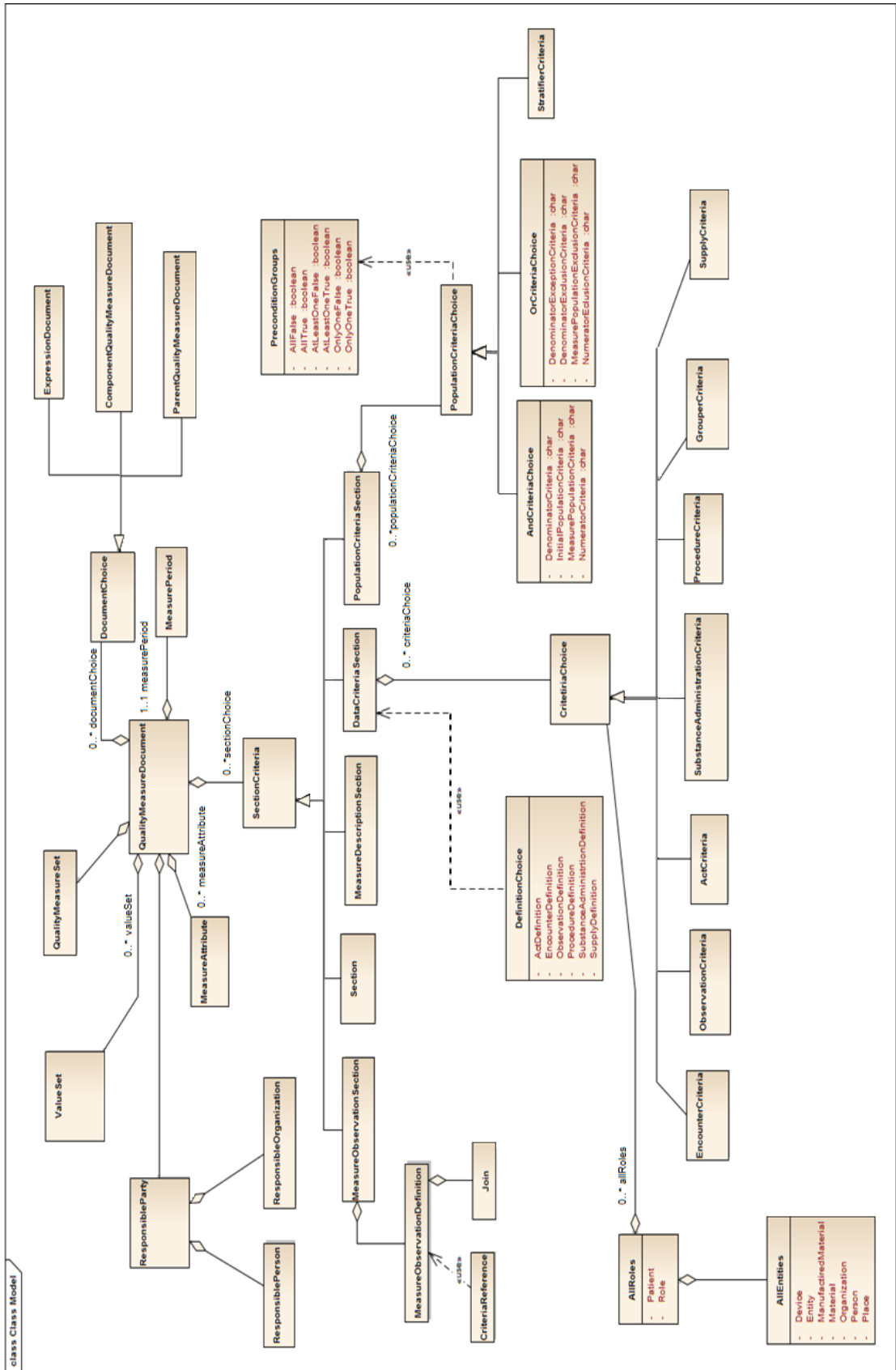


Figure 13: HQMF R2.1 class model, based on RIM model



### 4.3.3. HQMF2JS

The HQMF2JS project includes the Health-data-standard library. This project provides the JavaScript code for each indicator. In fact the JSON file (the output of the parser) is translated into JavaScript code. This JavaScript code is generated using other custom-built libraries, such as the patient-API, the HQMF library and the Logging library.

*What is the patient-API?*

The patient-API provides an object-oriented wrapper for test patients' data. Its class structure is based on the Green CDA for C32 standard and it also supports additional data items for the use of electronic health record systems by hospitals and health care providers. In addition there are functions that filter all of the patients encounters based on value sets.

*What is the HQMF library?*

The HQMF library implements some of HL7 data types and all of the required Quality Data Model - QDM functions. It also extends the patient API to provide conversions between patient API primitives and the HL7 data types. For instance, the patient API event base class is extended to provide a method to yield an HL7 IVL\_TS that captures the start and end timestamps of the event.

*What is the Logging library?*

The logging library wraps the patient API functions and the HQMF library methods to generate a log of the execution of an indicator for each test patient record. It is used to determine what data and population criteria a patient record satisfies.

Figure 14 illustrates a structure of the HQMF2JS project. There are several JavaScript templates for each data types such as data criteria, measure period, population criteria and precondition. Based on the kind of data, the corresponding JavaScript template is used to generate the analogous JavaScript object. During the conversion, common utility functions of library assets and the map reduce process are applied to JavaScript objects. In that way, the required functions are available for use in the map reduce framework during the Mongo DB database connection.

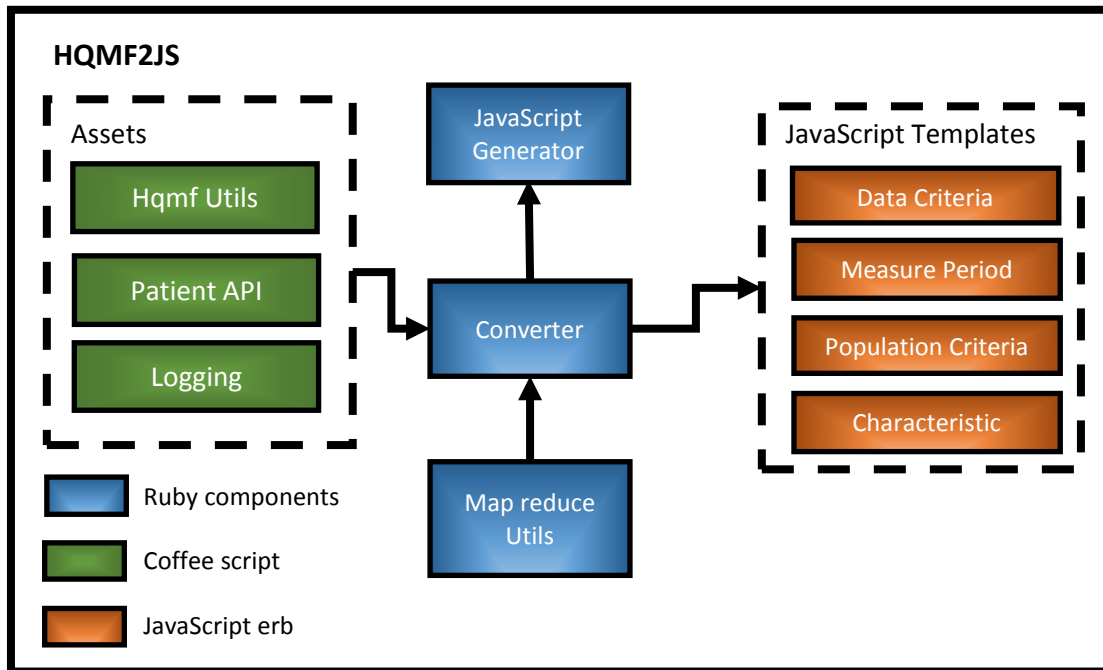


Figure 14: HQMF2JS components

Each of the generated JavaScript function of a data criterion should run against a patient record. Figure 15 shows a part of a patient record, in QRDA Category 1 format. Every patient record consists of all the events that a patient participated in, referring its code and its values.

```

First Name: John
Last Name: Doe
DOB: March 24th, 1973

Conditions:
  Name: Diabetes Type 2
  Code: 111552007
  Date: June 21, 2012

Lab Results
  Name: LDL Cholesterol
  Code: 12773-8
  Date: April 4, 2012
  Value: 175

Vitals:
  Name: Systolic BP
  Code: 12929001
  Date: May 22,2012
  Value: 131

```

Figure 15: Patient records in QRDA category I format

This JavaScript code is executable by the Quality Measure Engine, which has been developed by MITRE, and which is used by Cypress and Pophealth.

#### 4.3.4. Quality Measure Engine

The Quality Measure Engine makes use of the following parts:

1. The generated JavaScript map functions which are derived from HQMF2JS project
2. The value sets by each measure
3. The patient API
4. The type of the measure, for example proportion, cohort etc. In each case a suitable reduce function is selected to execute.

The Quality Measure Engine is based on a bundle of indicators that MITRE published and that are used in Cypress as testing measures. This bundle contains all the precomputed indicators, indicators' results, the test patient data, and the value sets for each criteria. Then, after the Quality Measure Engine connects to the database, where the bundle is loaded, it utilizes the map-reduce framework of the Mongo DB. The map-reduce framework is based on JavaScript language.

##### Map-Reduce process

###### Step 1 – map function

Each map function can be run in a particular patient record and calculates if the patient includes a specific data criterion or not. In addition, the result of the map function is classified in the corresponding type of population where the criteria belongs.

###### Step 2 – reduce function

Based on the type of the measure, the corresponding reduce function takes place and sums the results in each population.

This is the whole process that is executed using the above open source project and library approach.

#### 4.3.5. Ruby on Rails Applications

Due to the fact that the previous open source projects are web applications developed on the Ruby on Rails framework, as well as that all the libraries are written in the Ruby language, one of the related domains of UQMEE for HSDP relies on this framework. The key point of the UQMEE for HSDP is to implement a prototype application which gives insights into an architecture using HL7 standards.

##### *Ruby on Rails features overview*



Rails is a framework for web developed applications. It is popular for its ease of use as it simplifies many common repetitive tasks in web development [13].

Figure 16: Ruby on Rails logo

Rails is written in Ruby. Ruby is used to Rails as Python to Django. However Ruby is a new and very appealing language because of its elegance and its clarity.

One of the greatest principles of Ruby on Rails development is the convention over configuration. In simple words, the developers do not have to spend a lot of time configuring files in order to set up the application. Rails provides a set of conventions assisting in speeding up the development.

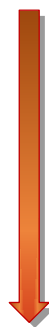
Another principle of Rails is the emphasis on RESTful application design. Rest (Representational State Transfer) is a software architecture based on the client-server relationship. As web applications require many request types to web servers, Rails gives simplicity over the handling of request. GET serves for viewing things, POST serves for creating things, PUT serves for updating and DELETE serves for deleting things.

Over the last few years, Ruby on Rails is expanded and obtains more followers, due to its ease of use and the fast developing.

#### *Advantages and Disadvantages*



1. The development time is lower than other frameworks allows. Ruby is an object-oriented language. The fact that Ruby is supported by a large variety of open source code available within the Rails community its availability of its resources is underlining.
2. The Rails conventions allows developers to move between different Rails projects easily, as each project will tend to follow the same structure and code practices.
3. Ruby code is very readable and mostly self-documenting.
4. Rails has developed a strong focus on testing and has good testing frameworks.
5. Rails and most of its libraries are open source and cost free.



1. Not all website hosts can support Rails. One of the reasons is that Rails is more resource intensive than PHP. However, Rails-friendly hosts do exist, for example, Heroku and Engine Yard. Alternatively, the Rails application can be hosted on a Virtual Private Server (VPS) with Amazon EC2, or Linode.
2. Java and PHP are more widely used. As a result, there are more developers in these languages. However the number of Ruby developers is growing fast every year and its community is active.
3. As for the performance and scalability, Rails applications are not as fast as Java or C, However the library JRuby in a Rails application improves the performance characteristics at the same level as Java.

## **4.4 Health data storage**

Another important domain for the implementation of the UQMEE-4-DHP is the data base technologies [14].

### **4.4.1. Relational Databases**

Relational databases are organized based on the relational data model. In this model data is organized into one or more tables or relations of rows and columns identifying a unique key for each row.

In principle, each object type described in a database is represented by its own table, the rows represent instances of that type and the columns represent values attributed to that instance. As each row obtain a unique key, various rows of one table can be linked to rows of another table by mapping the corresponding keys to each other.

Software systems used to maintain relational databases are known as Relational Database Management Systems (RDBMS), and virtually all relational database systems use SQL, Structured Query Language, a querying language for updating and moderating the database.

When developing a data storage system, usage of a relational database is good to be applied for the following reasons:

1. There are relations between data.
2. There is reusability of data tables.
3. There is a need for data normalization, in terms of decreasing the dependency between the data.

#### 4.4.2. Document Schema-less databases

In case the data storage system requires a high rate insert or select for big data, the relational databases can provide high performance. In addition, when data fields should be dynamically be configured then a document schema-less database could give a better solution [15].

In that case, each document is a self-contained piece of semi-structured data, and the data is de-normalized.

In general, based on the data type, if there is need to have document collections without links between the different documents collections, the document schema-less database can be suitably used. Starting applying links between documents, the use of document schema-less database is misused.

One popular document schema-less database is Mongo DB, [16]. Mongo DB is suitable for big data and high rate inserts in a database. Moreover, it is a documentation database, in other words, every record can be printed in a paper without being depended on additional references to other documents.

Hadoop assists Mongo DB's big data, [17]. Hadoop is a data ware house which provides massive data storage and faster processing.

An example of efficient querying patient data against a set of criteria using Mongo DB identifies the following requirements:

1. Data is stored as patient record documents having various fields.
2. Map and reduce functions are written in JavaScript.
3. Depending on the data criteria, we have to provide the proper map JavaScript functions:
  - a. Instantiation of methods and variables
  - b. Providing the appropriate map function
4. Reduce Function sums the results of the map functions based on the querying criteria.

Example:

In this example [18], it is assumed that patient records consist of the following elements

```
{id, name, age, city, activities :{ physical, healthy eating}}
```

The requested query is defined: Find the number of activities that patients older than 20 years old are subscribed.

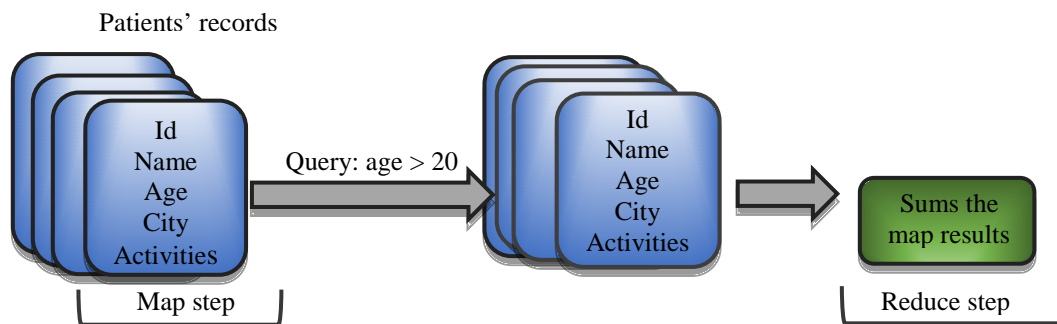


Figure 17: Map-Reduce process

The first map step finds the patients who are older than twenty years old. Then, the second step sums the results from the map step.

Map Reduce framework is used for big data but not for very complicated queries. In addition it is well used when there is need for parallel data processing. ■

# 5. Requirements

In this chapter the requirements of this project are presented. The requirements represent a more analytical approach towards the definition of the problem and the expected solution. They depend on the stakeholder analysis, Chapter 2, and problem analysis, Chapter 3.

## 5.1 Introduction

In section 1.3, the initial expectations state that this software solution should be able to:

1. Provide a User Interface (UI) to quality analysts, in which they can select a specific indicator for measuring
2. Provide indicators using HL7 standards, like HQMF
3. Provide a quality measure execution engine based on HQMF standards
4. Provide the results of the HQMF according QRDA Category 3 format
5. Show the results for analysis using graphs through the UI

In light of these points, the set of the requirements are categorized into:

- User requirements
- Functional requirements
- Non-functional requirements

## 5.2 User requirements

The end-user of the software solution could be medical practitioners, CFOs, ACOs, and CEOs. These kinds of users do not always possess the knowledge of quality measurement using HQMF. Their need is to obtain a tool in which they are able to choose a specific indicator and request the corresponding results for analysis. In that case the user requirements can be defined as follows in Table 6:

Table 6: User Requirements

ID	Description
UR_1	The user should be able to select from a given list the indicator he needs to measure
UR_2	The user should be able to submit a request for the measurement of a specific indicator
UR_3	The user should be able to receive the resulting measurements through graphs
UR_4	The user should be informed from the system about the possibly not successful measurement of a specific indicator.

In UR\_1, the indicators could be either clinical or claims indicator.

Regarding the user requirements, the following use case scenarios are defined in the sub-function level:

- Select indicator type
- Request the measurement of a specific indicator

In the use case description, we assume that the end user is a medical practitioner. The Use Case Scenario 1 (UCS 1) describes the interaction between the user and the system where the user selects the indicator for measurement. The Use Case Scenario 2 (UCS 2) describes the process where the user submits the selected indicator for measurement.

UCS 1:

1. The medical practitioner chooses the indicator types he needs to select from a list, clinical or claims indicators
2. The system provides a list of indicators to the user
3. The medical practitioner selects the specific indicator for measurement

UCS 2:

1. The medical practitioner submits the selected indicator (UCS 1)
2. The system calculates the quality measure of the specific indicator
3. The system displays to the medical practitioner the corresponding results

Extension 2.1 of UCS 2:

Title: Invalid indicator

2-a The system informs the user when the indicator is not possible to be executed, because it is an invalid HQMF document

Extension 2.2 of UCS 2:

Title: Unsuccessful measurement

2-a The system informs the user in case the selected indicator is not possible to be measured due to data inconsistency

Extension 2.3 of UCS 2:

Title: Unsuccessful querying

2-a The system informs the user in case of time out of the querying execution time, the session expires in 10 seconds

### 5.3 *Functional requirements*

In this project, the functional requirements are defined based on the needs of Philips Research, as described in Chapter 3. Table 7 presents the extracted functional requirements of UQMEE. These are identified by the Philips Research stakeholders according to their needs.

Table 7: Functional Requirements for UQMEE for HSDP

<b>Id</b>	<b>Name</b>	<b>Description</b>
F1	HQMF standard compliant	The UQMEE for HSDP should provide a HQMF based measurement engine for a set of indicators formulating by HQMF standards
F2	Data parsing	The HQMF documents should be parsed using the HQMF parser from Health data Standards library set
F3	Data storage	The data should be stored in a structured way based on HQMF data criteria definitions
F4	Reporting results	The quality measurement results should be presented in a QRDA Cat 3 document
F5	Supporting future use	The UQMEE should provide extensible data model regarding the HQMF elements, such as new procedures or problems



## 5.4 *Non-Functional requirements*

The non-functional requirements describe the qualities of the system. The non-functional requirements are listed in Table 8.

Table 8: Non-Functional Requirements for UQMEE for HSDP

<b>Id</b>	<b>Name</b>	<b>Description</b>
NF1	Ease of Use	The design of UQMEE should clearly illustrate the way the HQMF information is translated to executable queries on a set of health data. In that way the HSDP business group can acquire proper insights that it has to take into consideration for their environment
NF2	Backwards compatibility	The UQMEE for HSDP should support all the previous versions of HQMF releases, (R1, R2, R2.1)

■

# 6. System Architecture

This chapter describes the system architecture of the UQMEE. It starts by outlining the relation between the UQMEE for HSDP and its concepts are presenting the system as a black box. Then, it gives an overview of its components and its data model that are enclosed in this software solution.

## 6.1 UQMEE for HSDP overview

The UQMEE for HSDP is a tool to be deployed on top of HSDP which provides access to health data of various sources. Figure 18 shows the UQMEE as a black box.

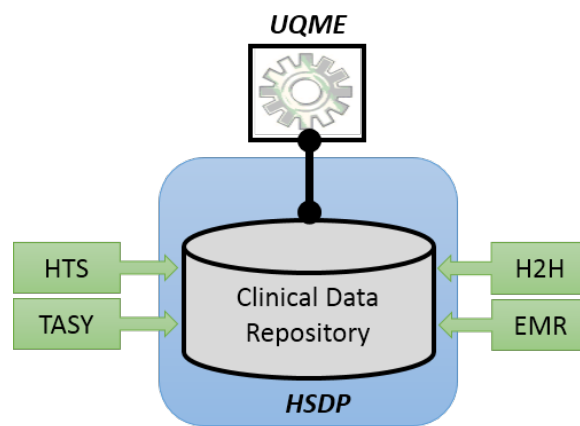


Figure 18: UQMEE for HSDP as a black box

As medical practitioners should be able to choose the indicators for quality measurement and view their results, HQMF for HSDP should also provide them the interactivity to do that. Figure 19 shows the UQMEE, its UI and the Clinical Data Repository within HSDP.

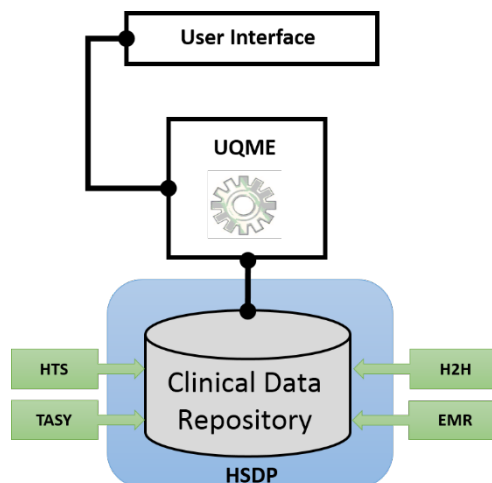


Figure 19: UQMEE for HSDP

However, in this software solution, there is no integration between HSDP and UQMEE. Data is stored in a data repository to which UQMEE connects directly, Figure 20.

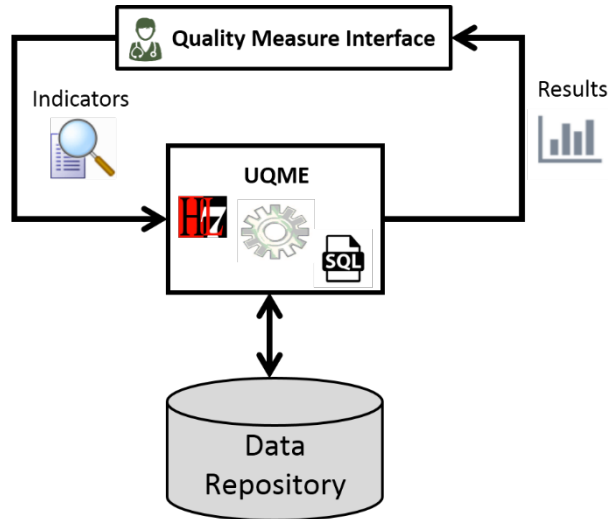


Figure 20: UQMEE for HSDP overview diagram

Regarding the fact that the UQMEE for HSDP needs a UI which we can call Quality Measure Interface, the project can be divided into two main sub systems, the Quality Measure Interface (QMI) and the Quality Measure Engine (QME), Figure 21.

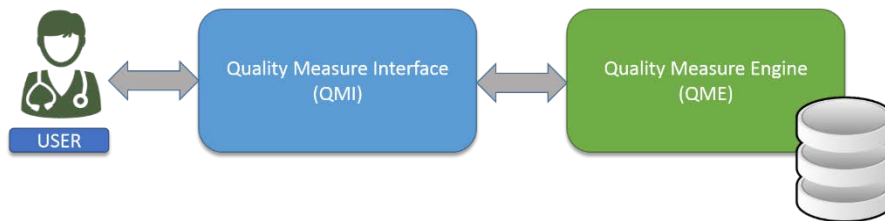


Figure 21: QMI and QME sub systems

Specifically, in the QMI the following actions take place:

- capture input and select the specific HQMF document, which is stored in QMI
- the extraction of the measurement results from QRDA Cat 3
- the display of the results

Respectively, in the QME the following actions are taking place:

- The HQMF parsing
- The translation of the extracted HQMF objects into a query structure
- The execution of the queries to data model

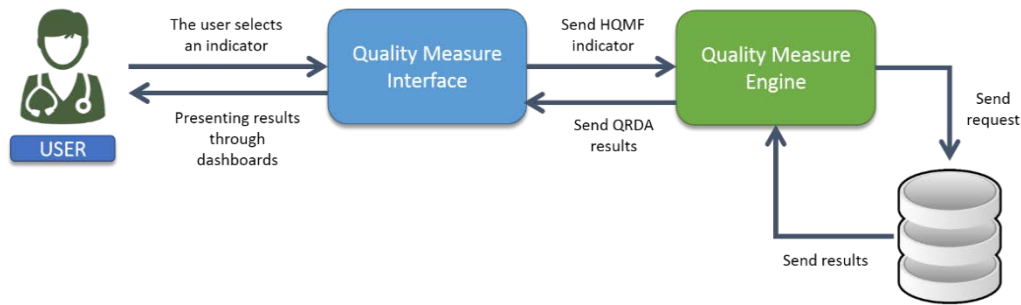


Figure 22: Quality measurement process

The role of these two sub-systems is important, as the first one is focusing on the interface and the second one on the measurement execution. In that way, the dependency between the QMI and the QME technologies is loosely coupled. A loosely coupled system is more flexible, it is easier to modify and adapt for new purposes. This approach exhibits the ease of use of this software solution, one of the design non-functional requirements.

Figure 22 presents an overview of the process involving the QMI and the QME. The user may select the indicator for measuring through the UI on QMI, where the corresponding HQMF xml document is selected and is provided to the QME. In the QME, the received HQMF xml document is transformed into queries which are executed on the data model. Then, the results are placed into QRDA Cat 3 document and are sent back to the QMI. In QMI, the results are extracted and are applied into suitable graphs to be presented through the UI to the user.

The following sections present more details of the internal architecture and process of QMI and QME.

## 6.2 Quality Measure Interface

This section presents an overview of the QMI architecture. In the QMI, the following processes are included:

- User interaction
- Preparing and displaying the results to the user

The architecture pattern of QMI is the Model View Controller (MVC). It is based on the interaction between the UI and the data model orchestrated by the controller.

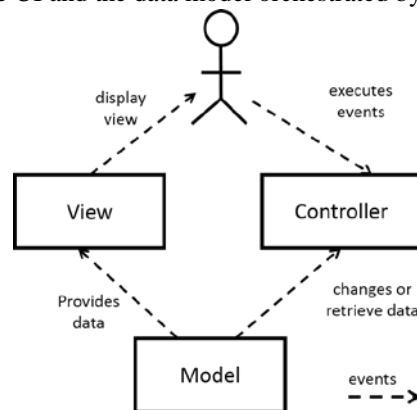


Figure 23: MVC

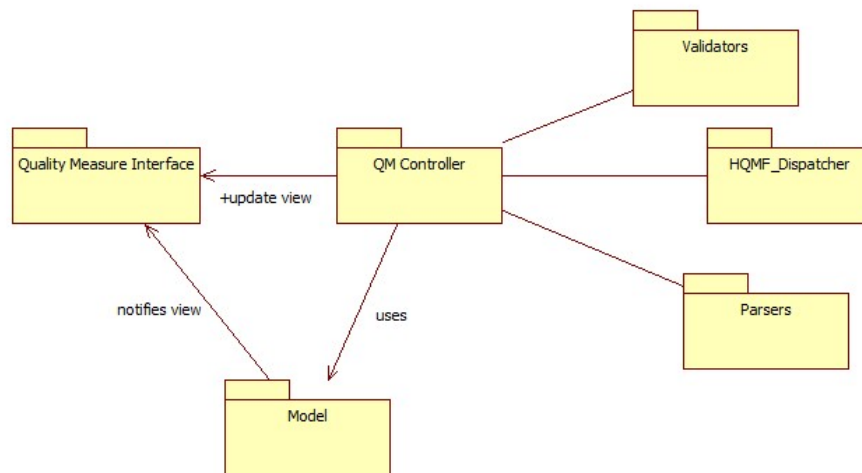


Figure 24: MVC package diagram of QMI

The components of MVC architecture are the View, the Model and the Controller. The Controller is the one which handles the interaction between the user through the View and the Model. User actions can create events related to the Model. The Controller also renders the new upcoming information towards the View. Figure 24 shows a package diagram of MVC architecture of QMI, in which are also presented the related components of the QM\_Controller, such as Validators, the HQMF\_Dispatcher, and the Parsers.

Clarifying the flow process of QMI, Figure 25 presents a diagram where the red number depicts each step that is taken, as follows:

- 1:** the user selects from the UI, a specific indicator title for measuring
- 2:** the parameters of the selected indicator provides the input to the QM\_Controller which requests from the model the corresponding HQMF document from the model
- 3:** the HQMF xml document is transferred to HQMF Dispatcher
- 4:** the HQMF document is sent to QME
- 5-6:** the results from QME are received in the form of a QRDA xml document and sent to the QM\_Controller
- 7:** the QRDA cat 3 document is validated by QRDA Validator
- 8:** the QRDA Validator sends a True or False indication to QM\_Controller, indicating whether the QRDA document is well-formed based on its QRDA schema
- 9:** the validated QRDA xml document is the input for the QRDA parser
- 10:** the extracted results, such as the aggregated numbers of population (like IPP, DENOM and NUMER), are sent to QM\_Controller, where they are presented in suitable charts, like bar charts for display
- 11:** the results are presented through the UI to the user

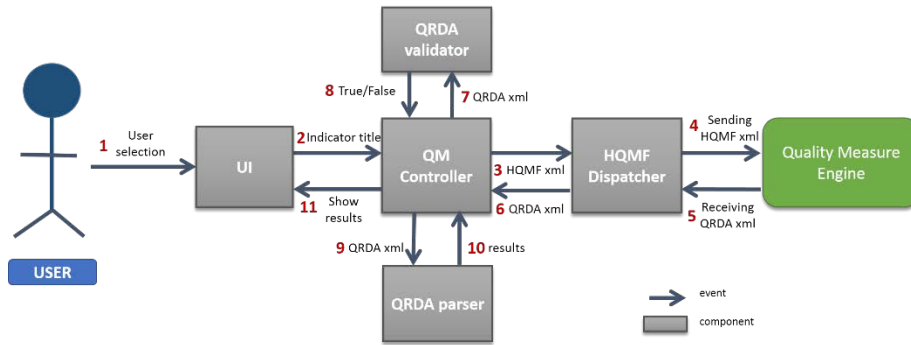


Figure 25: Flow process of QMI

Figure 26 shows a high level class diagram of QMI. The **QM\_Controller** is connected to the model by two types of models **Indicators** and **Reports**. The model class **Indicator** includes attributes of the used indicators, such as its title, and id. The class model **Reports** consists of the elements of the received results derived from **QRDA** documents such as the population types counts.

The **QM\_Controller** is also connected to **QRDA\_Validator**, the **HQMF\_Dispatcher** and the **QRDA\_Parser**. The **HQMF\_Dispatcher** is a SOAP web service provider, through this the **HQMF xml** document is sent to **QME** and the **QRDA xml** document is received to **QMI**. The **QRDA Validator** checks the validity of the schema of the received **QRDA xml**.

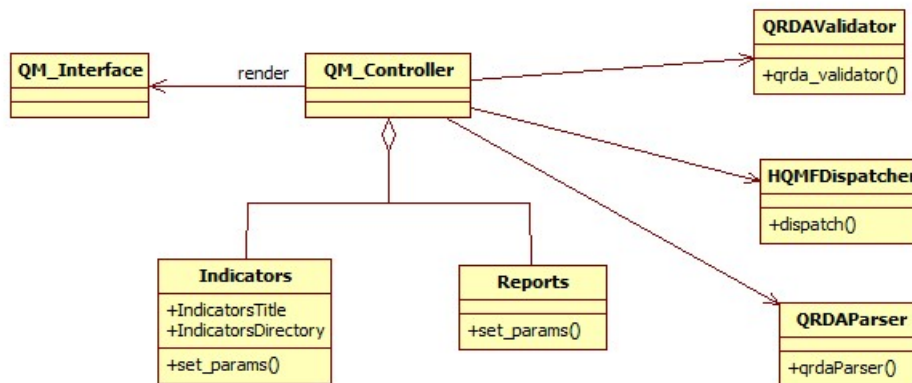


Figure 26: Class diagram of QMI

Regarding the QMI class diagram, two sequence diagrams are shown in Figure 27 and Figure 28. The first is set from the moment the user selects an indicator towards the moment the indicator is sent to the **QME**. The second is set from the moment the resulting **QRDA** document is received from the **QME** till the moment the results are shown through the **UI**.

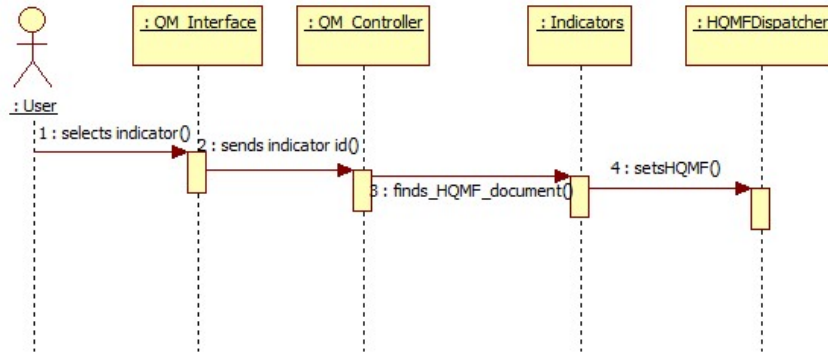


Figure 27: Sequence diagram from the moment the user selects an indicator until the moment the corresponding HQMF is sent to QME

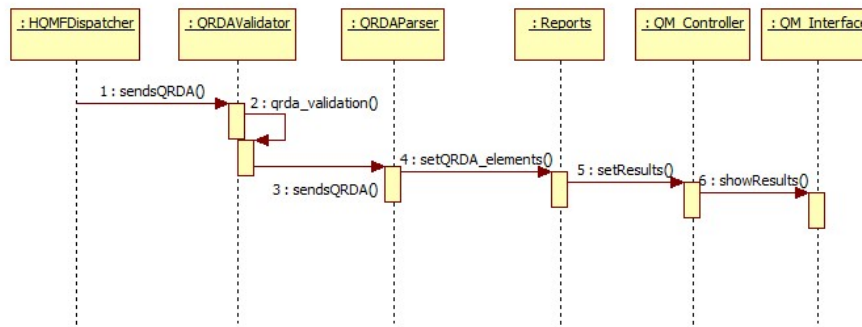


Figure 28: Sequence diagram from the moment the QRDA is received till the moment the results are shown to QM\_Interface

### 6.3 Quality Measure Engine

This section presents the QME architecture and its main components. The QME system presents the following three processes:

1. HQMF parsing
2. Query generation
3. Execution of queries on the data model

The following Figure 29 gives a general overview of the main components of QME. In this case, there is no user interaction: the QME encloses an engine controller, which uses other components such as validators, parsers, query generation and QRDA generator.

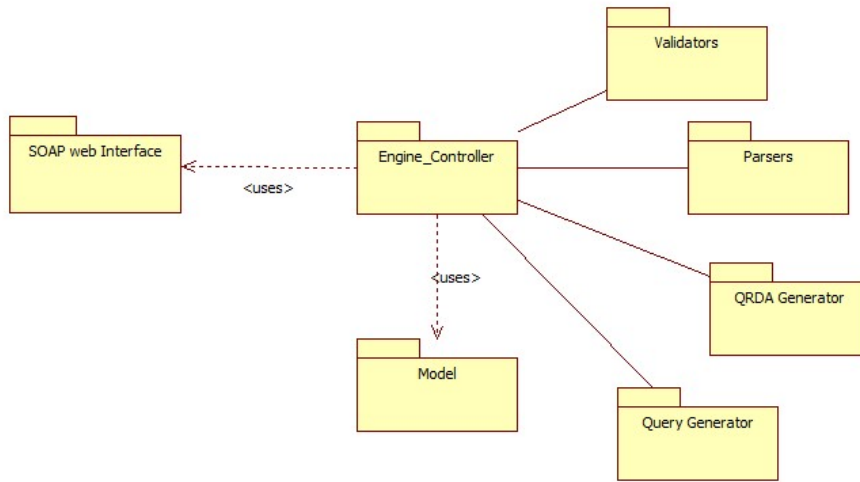


Figure 29: QME package diagram

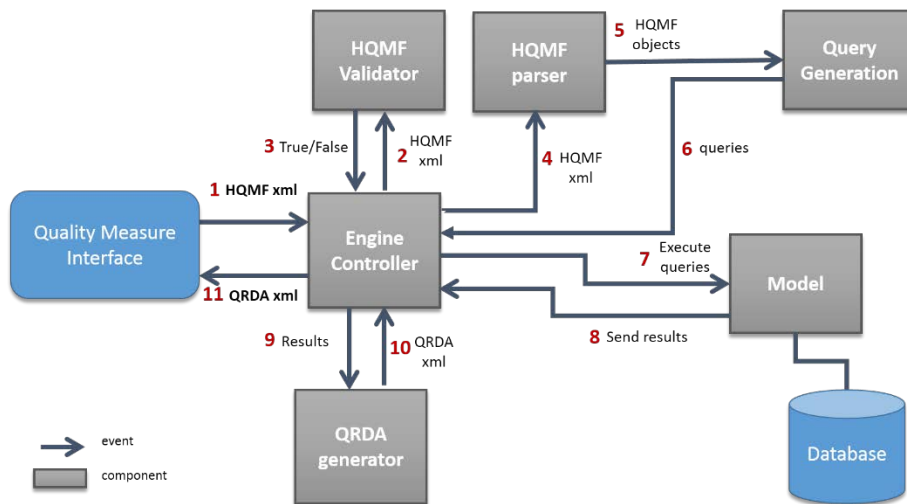


Figure 30: QME flow process overview

To clarify the flow process in this system the Figure 30 presents its steps.

- 1:** QMI sends a HQMF xml document to the Engine Controller
- 2:** the received HQMF xml is validated through the HQMF Validator
- 3:** the HQMF Validator sends a True or False indication to Engine Controller, indicating whether the HQMF document is well-formed based on HQMF schema
- 4:** the valid HQMF xml is parsed by the HQMF parser
- 5:** the extracted HQMF objects are used for query generation
- 6:** the generated queries are transferred to Engine Controller
- 7:** the Engine Controller sends the generated queries to the data model for execution



**8:** the results for each population type are returned by the Model to the Engine Controller

**9:** the received results are sent to QRDA generator for conversion into a QRDA document

**10:** the QRDA in xml format is sent back to the Engine controller

**11:** the QRDA document is sent to QMI

Regarding the QME process, the validation process is similar to QRDA Validator. It checks for syntactic correctness of the HQMF xml schema of the received document. Both in QMI and QME the validation attempts to ensure that the received documents are complete and valid, complying with their schema. The HQMF parser is also part of the Health data –standard library, as well as the QRDA generator, see Chapter 4. In the query generation process, the extracted HQMF objects are structured in such a way that they can be mapped to queries that are suitable for the chosen relational data-base management system e.g. SQL-based.

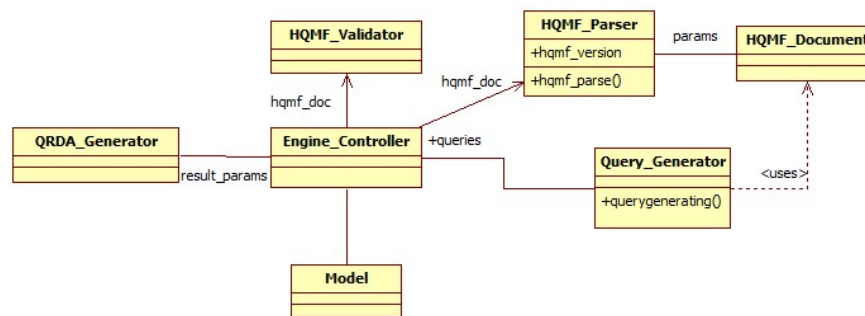


Figure 31: QME class diagram

The class diagram of QME is presented in Figure 31. This class diagram does not include design details of QueryGeneration: the analysis of the query generation is presented in next Chapter 8.

## 6.4 Data model

In the preceding sections a general overview of the major components of the UQMEE for HSDP is given. This section describes the type of the used data, as well as its structure.

The HQMF model supports a big variety of health data types. Some of these could be encounters, patient demographics, medications, procedures, physical exams, allergies, lab results, problems, immunizations, vital signs and diagnostic results. However, in this software solution some of them are used. It is not possible to include all these data elements. The HQMF elements that are used are:

- Characteristics (patients' demographics)
- Encounters
- Diagnosis
- Medications
- Procedures
- Transfers
- Activities

These data types highlight two main concepts: 1) the patient characteristics and 2) the encounters. These two form the core of the data schema ( Figure 32), as each patient participates in an event in healthcare delivery.

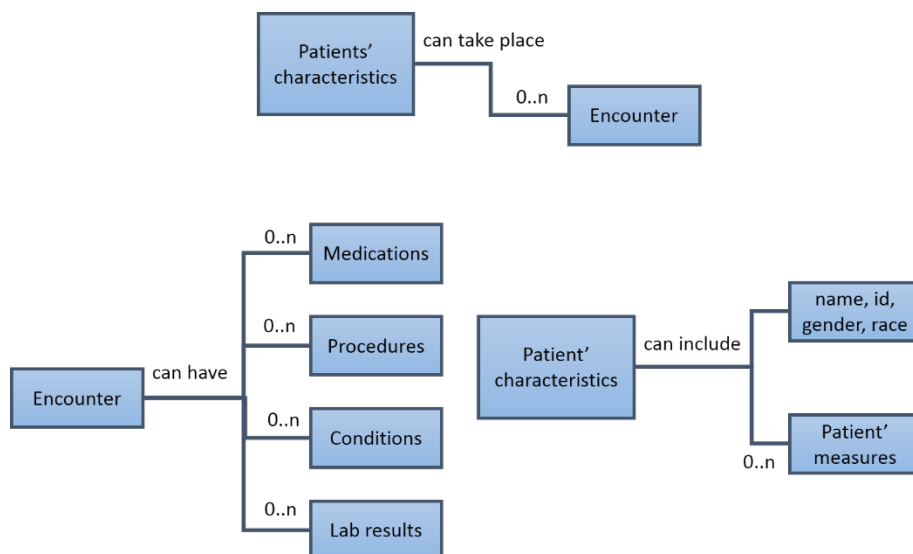


Figure 32: Example of HQMF data elements' relations

In the patient characteristics, all the necessary information of the people dealing with healthcare institutes and hospitals is included. This kind of information concerns personal attributes of patients such as their birthdate, gender, race, ethnicity, risk profile and payer type.

The encounter is also a core data type, as each process in a healthcare department occurs within an encounter. For instance, during an encounter the patient may receive services such as a procedure like a surgery, a medication, a diagnosis or a consultation. Due to this fact, the data model centers around the encounters, which can have multiple conditions, costs, medications, transfers or procedures. In addition each patient can be subscribed to multiple encounters. Figure 33 below presents the data model of the UQMEE for HSDP.

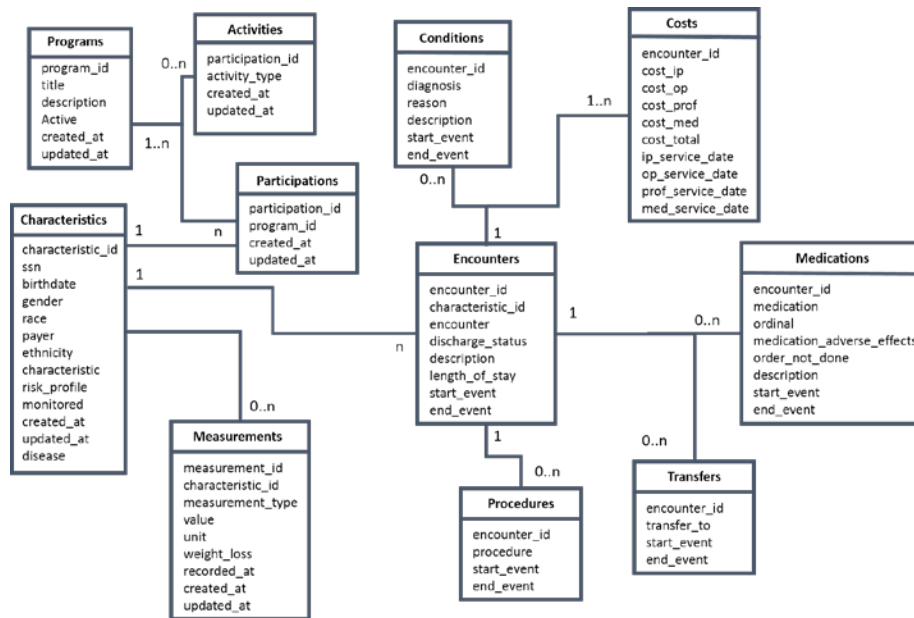


Figure 33: Data model of UQMEE for HSDP

Based on this data model, the following indicators are used, and also some of their data elements are presented, Table 9.

Table 9: Indicators

Indicator Title	Data elements
1 Obese Patients who achieved a weight loss of more than 2%	characteristics, conditions, measurements, encounters
2 Obese Patients who participated in the Physical Activity Group and performed weight loss more than 5%	characteristics, conditions, measurements, activities, encounters
3 Acute myocardial infarction (AMI) patients who are prescribed aspirin at hospital discharge	characteristics, conditions, medications, encounters
4 Ischemic stroke patients with atrial fibrillation/flutter who are prescribed anticoagulation therapy at hospital discharge	characteristics, conditions, medication, procedures, encounters
5 Patients with elective vaginal deliveries or elective cesarean sections at $\geq 37$ and $< 39$ weeks of gestation	characteristics, conditions, procedures, transfers, encounters
6 Median Time from ED Arrival to ED Departure for Discharged ED Patients	characteristics, conditions, encounters
7 Hearing Screening Prior To Hospital Discharge	characteristics, conditions, diagnostic results
8 The average cost per occurrence for inpatient encounter during the year 2012	Encounters, costs
9 The average length of stay for encounters during 2012	Encounters, costs

Regarding the functional requirement F5, the data extensibility in this data model can be achieved by adding new processes during an encounter or increasing the attributes in each table. For example, an encounter can be connected to diagnostic results, vital

signs or physical exams. In addition, a condition is possible to contain a primary and secondary diagnosis.■

# 7. System Design

Chapter 6 gives an overview of the system architecture, while this chapter provides more details of the process defining the components, modules and data for the UQMEE system.

## 7.1 QME design introduction

The QME system performs the transformation of HQMF information into the respective queries. The package diagram of the Engine is given in Figure 34, and it consists of the HQMF\_Parser, the HQMF\_Document and the Query\_Generation.

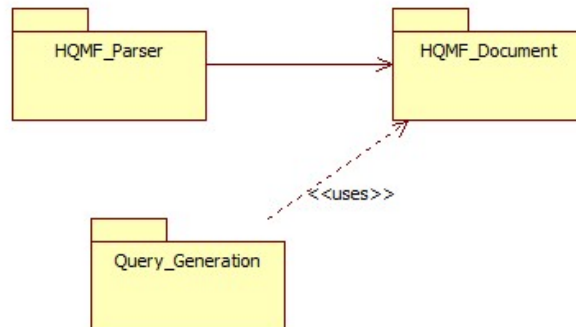


Figure 34: Package model of Engine

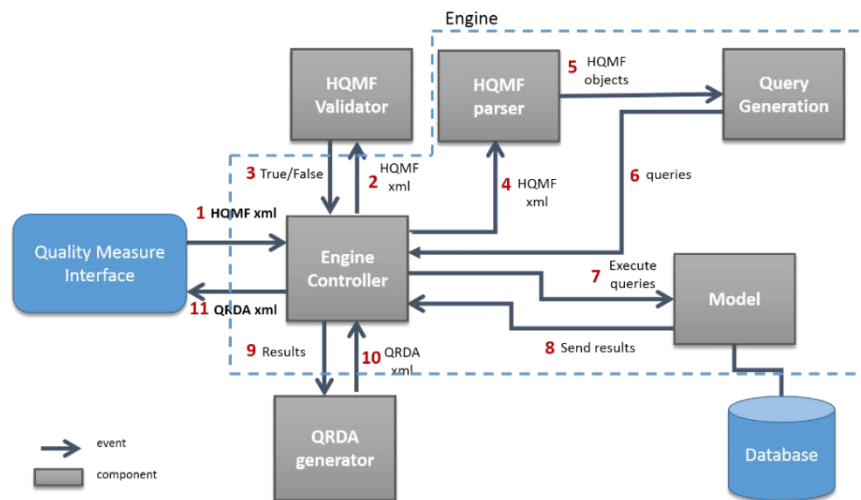


Figure 35: Highlighting the Measure Execution Engine process

In the Query\_Generator component, the transformation of the HQMF information to SQL-based queries is performed. The query generation may be represented as a text-to-text transformation between HQMF and SQL statements.

In the following sections is described 1) the main process of mapping the HQMF elements to SQL-based queries and 2) a description of the Engine components and design.

### 7.1.1. HQMF to Query transformation

It is important to point out that SQL commands may differ depending on each vendor Database Management system. In general, SQL-based queries consist of the following elements:

- Relations (Tables)
- Attributes (Columns)
- Select Statements
- Where Clauses
- Logical Operators
- Comparison Operators
- Aggregation Operators
- Joins

The simplest form of a SQL query may be a declaration of a *SELECT* statement such as “*SELECT column\_name FROM table\_name.*” In case we want to query using a specific criterion, the *WHERE* clause is used. For example, “*SELECT column\_name FROM table\_name WHERE column\_name operator value.*”

In case we need to combine multiple criteria to narrow data in a SQL statement, we can use Logical Operators such as “*SELECT column1, column2, columnN FROM table\_name WHERE [condition1] AND [condition2] ...AND [conditionN].*”

Data criteria elements define the data of interest included in the measure. Data criteria elements can be mapped to *SELECT* statements using *WHERE* clauses.

Population is defined consisting of one or multiple combined data criteria. They specify the required population for measurement. Population criteria may be mapped as a unified group of *WHERE* clauses using Logical Operators.

For example, we can construct two Data Criteria (DC) elements and combine them as follows, Table 10:

Table 10: Data criteria elements examples of their query form

DC	Description	Query
DC 1	“Patient is older than 16”	<i>SELECT * FROM patients WHERE age &gt;16</i>
DC 2	“Patient is male”	<i>SELECT * FROM patients WHERE gender = “male”</i>
DC 2 & 3	Combine the criteria using the “AllTrue” operator, mapped to logical AND	<i>SELECT * FROM patients WHERE age &gt;16 AND gender = “male”</i>

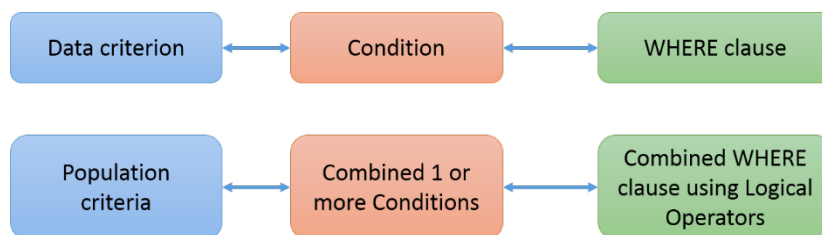


Figure 36: Relation between HQMF information to SQL-based queries

Other quality measures such as continuous variable measures define variable or calculations used to score a particular aspect of performance. For example, a continuous variable measure could calculate the median or average length of stay in hospital during a year.

This kind of calculations is also mapped to SQL queries. The following table shows an example of structuring a query, Table 11.

Table 11: Example of a continuous variable measure calculation

Type	Description	Query
Measure Population (MP)	“Encounter is done during 2013”	<i>SELECT * FROM encounters WHERE encounter.admission_date ≥01/01/2013 AND encounter.discharge_date ≤31/12/2013</i>
Computation Type (CT)	“Average length of stay of patients in hospital”	<i>SELECT AVG(length of stay) FROM encounters</i>
Combination of MP and CT	Combine the two criteria using the “AllTrue” operator, which is a logical AND	<i>SELECT AVG(length of stay) FROM encounters WHERE encounter.admission_date ≥01/01/2013 AND encounter.discharge date ≤31/12/2013</i>

More details regarding the data criteria and the population criteria are presented in Chapter 4 – section 4.2

Additional calculations can be defined, such as sum, find minimum or maximum. More descriptive mappings among the HQMF elements and SQL-based functions is set in Appendix A.

### 7.1.2. HQMF\_Parser – HQMF\_Document

Figure 37 presents the class diagram of the engine. The following part describes in sequence the presenting classes and their functionality.

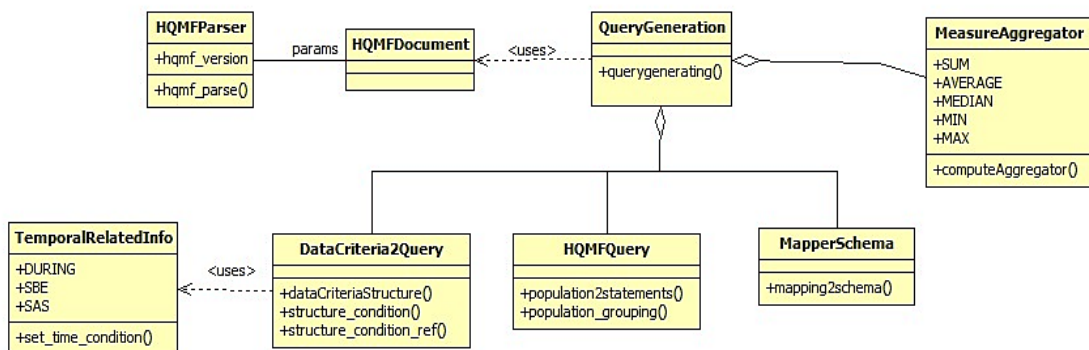


Figure 37: Class diagram of Engine

The execution of the engine starts from the moment the HQMF is parsed. Then the extracted objects are set based on the HQMF\_Document model. This model includes only the computable information of HQMF, Figure 38.

As mentioned in Chapter 5, the quality measuring tool should be backwards compatible to new HQMF versions and easy extensible in case of new HQMF functionalities. The QME uses the HQMF parser of Health-data standards library which supports R1, R2 and R2.1 releases and is backward compatible. The HQMF parser is described in Chapter 4 – section 4.3 . The parser encloses the HQMF R2 as its in-

ternal reference model, in case of parsing a HQMF R1 xml document; it transfers the R1 to R2 model.

HQMF is composed by data and populations criteria, which are stored into sections. The different types of populations and data criteria are distinguished by equivalent names. Different data criteria are set in the same structure, the data criteria elements are presented in Table 12. The different population types such as initial population or denominator have similar population elements, Table 14.

Each data criteria type includes the following main extracted elements:

Table 12: Data criteria elements

Data criteria elements	Description
local variable name	The unique name of the criteria within the HQMF document
title	The title of the data criteria
description	A description of data criteria
type	The general health data type to which the criteria belongs, such as encounters, medications, and conditions
definition	The specific definition of the equivalent data type such as encounter, diagnosis, and patient characteristic race
field values	Additional values that are included in data criteria
temporal references	References to related criteria or time constraints using a logical operator such as AND, OR, or DURING

For better understanding an example is given of the data criteria elements. This example derives from the quality measure *“Acute myocardial infarction (AMI) patients who are prescribed aspirin at hospital discharge.”* It shows the elements of a data criteria in which is defined the diagnosis of acute myocardial infarction (AMI).

Table 13: Example Data Criteria elements

Data criteria elements	Example elements
local variable name	DiagnosisActiveHospitalMeasuresAmi_precondition_5
title	Hospital Measures - AMI
description	Diagnosis, Active: Hospital Measures - AMI
type	Conditions
definition	Diagnosis
field values	Ordinal
temporal references	Diagnosis is defined during the Inpatient Encounter

Each population criteria type includes the following main extracted elements:

Table 14: Population criteria elements

Population criteria elements	Description
type	Indicates the population type like IPP, DENOM etc.
preconditions	In the precondition sections is included a local variable name which refers to the data criteria which take place to the definition of the population
id	The id code of the included data criterion
reference	The local name of the included data criterion
conjunction_code	The logical operator in which the data criterion is included

As above, for better understanding an example is given of the population criteria elements. This example derives from the quality measure *“Acute myocardial infarction (AMI) patients who are prescribed aspirin at hospital discharge.”* It shows the elements of the initial population criteria which includes a reference to a data criteria in which is defined the diagnosis of acute myocardial infarction (AMI).



Table 15: Example Population Criteria elements

Population criteria elements	Example population criteria elements
type	IPP
preconditions	id, reference
id	Id: 5
reference	reference: DiagnosisActiveHospitalMeasuresAmi_precondition_5
conjunction_code	allTrue

Each criteria type is enclosed into a section in the class HQMF document. In that way the section can be an abstract class and based on the criteria types specific classes are instantiated providing different functionality. To apply this approach the Factory Method design pattern is used. For each type of criteria section is instantiated the corresponding class object.

This design provides a simple way of extending the section types with minor changes in the software code. In case there is a change on data criteria, the only change will be applied in the corresponding code part. In addition, supposing that HQMF standard provides a new section, a new class will be added inhering the Section\_Criteria\_Factory, without influence the preceding functionality.

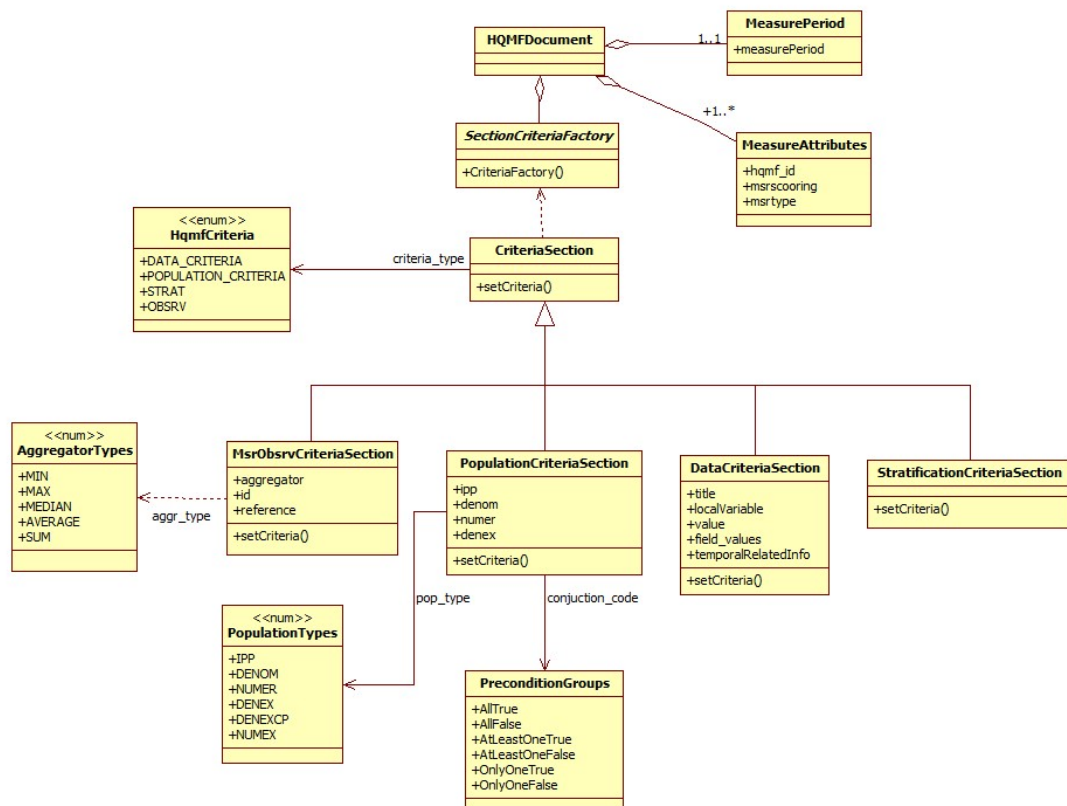


Figure 38: HQMF\_Document class diagram

### 7.1.3. Mapper2Schema

In this class, there are functions which act as filters. To execute a measure, measure's HQMF elements should be present in the dataset of the database. In this part, it is checked whether HQMF elements are mappable to the data model attributes. In that way, before the query generation, it is verified whether the HQMF can be executed. In case one of the HQMF elements is not contained in the data model, the query execution is not processed any further.

### 7.1.4. Query Generator

This class makes use of the HQMF objects (HQMF\_Document) specifying the generation of the queries. The final query generation is executed into three steps:

1. Each data criterion is transformed into a SQL condition statement.
2. Population criteria are combined into SQL statements.
3. Measure aggregation types such as sum, median, average are transformed into SQL aggregation constructs.

These preparation steps are implemented by Data\_Criteria2Query, Population2Query and Measure\_Aggregator, respectively.

### 7.1.5. Data\_Criteria\_2Query

Based on the data criteria structure, Table 16, the following transformation is taking place in Data\_Criteria2Query.

Table 16: Data criteria - SQL elements mappings

Data Criterion element	SQL elements
type	table name
definition	column name
title	value that corresponds to the definition
field_values: {display name, value, type such as intervals}	display name corresponds to column name, the type defines the logical operators such as greater or less
temporal_references: {reference to time constrains or other criterion}	additional condition to the current query. It is connected by "AND" operator and based on its elements it expresses a condition

Based on the above description, an example is given in the Figure 32.

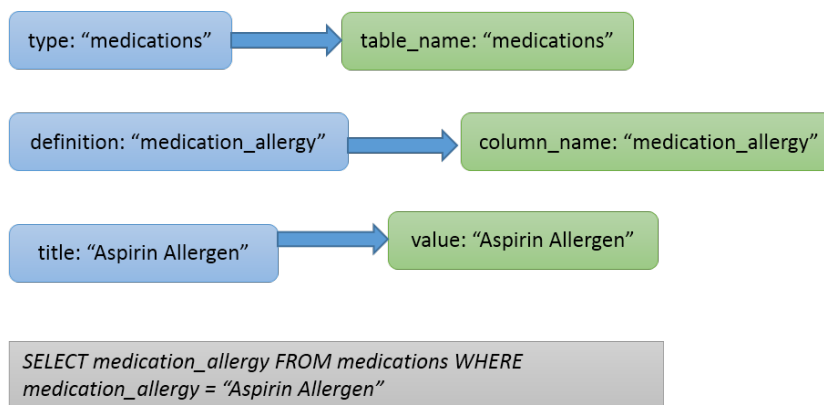


Figure 39: Example mapping between data criteria and query

More mapping examples are described in **Appendix B**.

### 7.1.6. Population\_Query

The population criteria include a nested combination of various data criteria. Each population criteria type such as IPP or DENOM has the form of a tree. Each root of the tree is a logical operator and each leaf a data criteria Figure 40. The tree traversing is done in order [19].

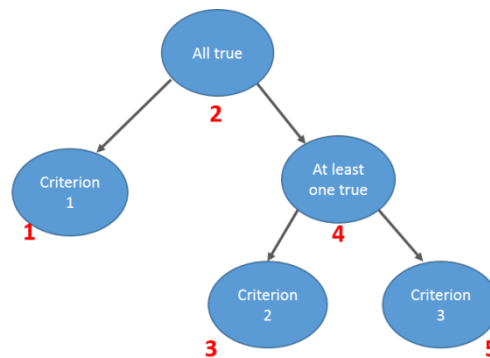


Figure 40: In order tree-traversal

Regarding Figure 40, if “All true” equals to the logical operator “AND” and “At least one true” equals to the operator “OR” then the produced statement is: Criterion 1 AND (Criterion 2 OR Criterion 3). This statement defines the required population that should be retrieved. Each criterion contains its own condition based on the analysis in the previous section Data\_Criteria\_2\_Query.

In case of a Measure Observation Definition section in HQMF, the included computation type is extracted. In this class the computation in SQL format takes place.

In the end in the **Query\_Generator** class the produced conditions and statements are combined into one statement per population. In that way the query is ready and is sent for execution to the Model through the Controller (EngineController), MVC architecture - Figure 23.

The next sequence diagram, Figure 41, illustrates the process flow from the moment the HQMF is parsed till the moment the generated query is sent to the Engine Controller.

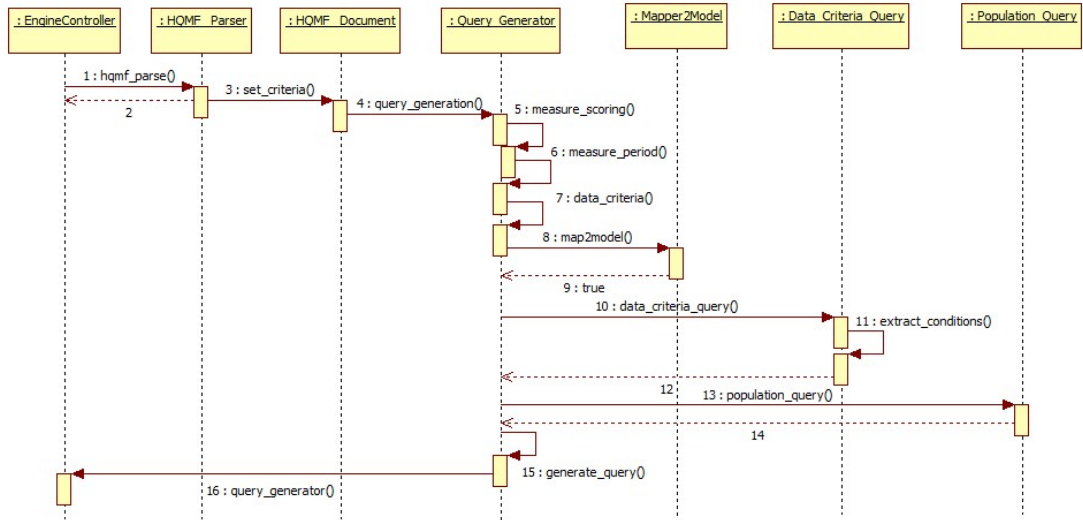


Figure 41: Sequence diagram of query execution

■

# 8. Implementation

In Chapter 6 and 7, the system architecture and design of UQMEE are presented. This chapter describes the implementation of the UQMEE. It starts by presenting the different components of UQMEE, followed by the technologies that are being used to realize each of them. Then it continues analyzing each technology choice, as well as some possible alternatives.

## 8.1 UQMEE Implementation

The UQMEE for HSDP is a web application, which consists of two sub-systems the QMI and QME. It is developed using the Ruby on Rails framework. QMI provides the ability to a user to select a specific indicator for quality measurement and displays results of executing this indicator. QME provides the following functions: 1) parsing HQMF and QRDA documents 2) query generation.

Table 17 presents the main components of UQMEE for HSDP and its corresponding used technology.

Table 17: Technologies used to realize each component

Components	Technology
QME, QMI	Ruby on Rails framework
HQMFParse, QRDAParser	Ruby libraries
Database	SQLite
QME-QueryGenerator	Object Relational Mapping

The next paragraph analyzes the technology choices for each of these components.

## 8.2 Technology choices for implementation of UQMEE components.

### 8.2.1. QMI and QME on Ruby on Rails framework

The UQMEE solution is a web application. The Ruby on Rails framework is chosen for web development since many open source libraries used for HQMF and QRDA are using this framework. The Ruby on Rails framework is an ideal environment for quick development of mini applications. For example, developers do not need to spend much time configuring the connection between models, views and controllers. Ruby on Rails conventions provide an easy navigation inside the application and because there are plenty of open source Ruby libraries called gems the development time is decreased. More information about Ruby on Rails framework can be found in Chapter 4 section 4.3.5.

The Rails framework is based on the MVC architecture and it paves the way for implementing Models, Views and Controllers. Some of the major Rails components are:

**Model:** It provides a base for the data models in a Rails application. Specifically, it maps the data models to Ruby objects and contains the corresponding business logic.

This sub-system is implemented using the ActiveRecord library which provides the connection between the tables in a relational database and the Ruby program code.

**Controller:** It coordinates the interaction between the Views and the Model. It processes incoming requests of querying the models for specific data and organizes the data into a form that fits the needs of a given View. This sub-system is implemented by the Action Controller library.

**View:** It gives a representation of the data in a specific form. This sub-system is implemented in the ActionView library providing representation templates for data display.

In our implementation, we used Rails release 4.2.1 and Ruby 2.0.

As mentioned before the UQMEE consists of the QMI and QME components. QMI contains 1) the UI implementation, 2) QRDA parsing and validation and 3) other interconnection processes between model and controller. Figure 42 shows the relation between the Quality Measure Interface where the UI implementation is contained and the Action View, the QM Controller which uses the Action Controller and the Model which makes use of the Active Record.

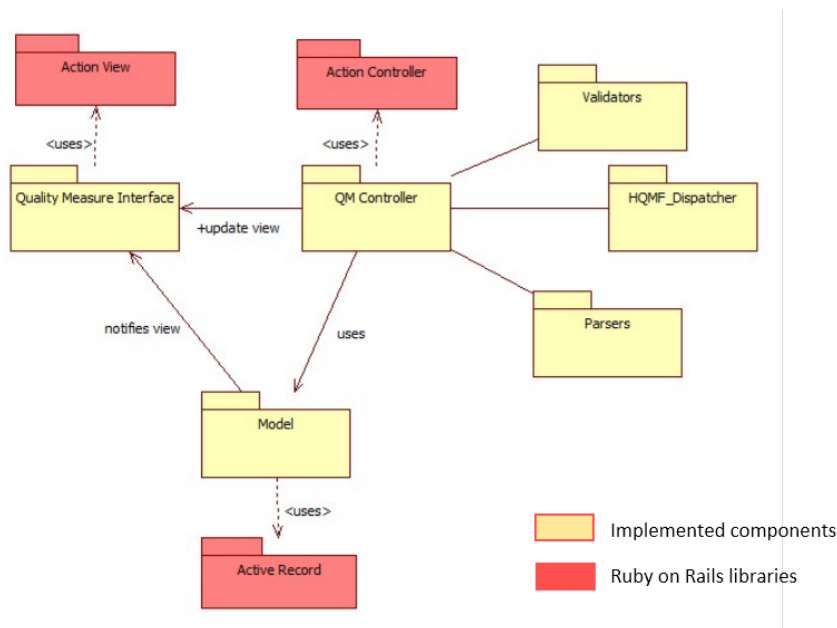


Figure 42: Relation between QMI and Ruby on Rails components

The UI implementation is based on HTML 5, CSS and JavaScript technologies. The QRDA Parser and Validators are using Ruby libraries which are going to be described in the following section 8.2.2.

Specifically for the UI implementation basic examples are studied from Twitter Bootstrap and suitable templates are used [20]. Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML, CSS and JavaScript design templates for typography, forms, buttons and other interface components.

For results rendering via graphs, an open source HTML 5-based library, the Chartjs is used [21]. This library also provides good documentation for several types of charts. Its community is active to the users of a library who are asking solutions over their specific needs.

### 8.2.2. Parsing HQMF and QRDA libraries

The Health - data standard library is used for consuming HQMF and QRDA standards. The Health - data standard is a Ruby-based library. I used the 3.5.3 release for this implementation. A new, green field implementation of a parser was not feasible due to the lack of time. In addition, these libraries have been already used and tested in various applications of quality measuring and testing procedures. Our research showed that there are no other open source libraries for supporting HQMF or QRDA parsing.

### 8.2.3. Relational data management system and other alternatives

The UQMEE for HSDP needs a suitable database for the computation of different indicators. In general there are two options of database technologies, 1) the relational database technology and 2) the schema-less database technology.

The database selection is based on the data relations and the required performance of the UQMEE application. Chapter 4 – paragraph 4.4 provides an overview of the available database technologies including their pros and cons. For this implementation, the chosen technology is the relational data model.

The reasons we choose the relational data model are the following:

1. Most healthcare providers use relational database systems
2. The current implementation depends only on HQMF, and QRDA open source parsing libraries
3. The learning curve was lower compared to schema less database technologies during the required period

In case we use a schema-less database, like MongoDB, we may follow an implementation similar to Cypress, which is described in Chapter 4 - section 4.3 . The corresponding QME component using a schema-less approach is illustrated in the following Figure 43.

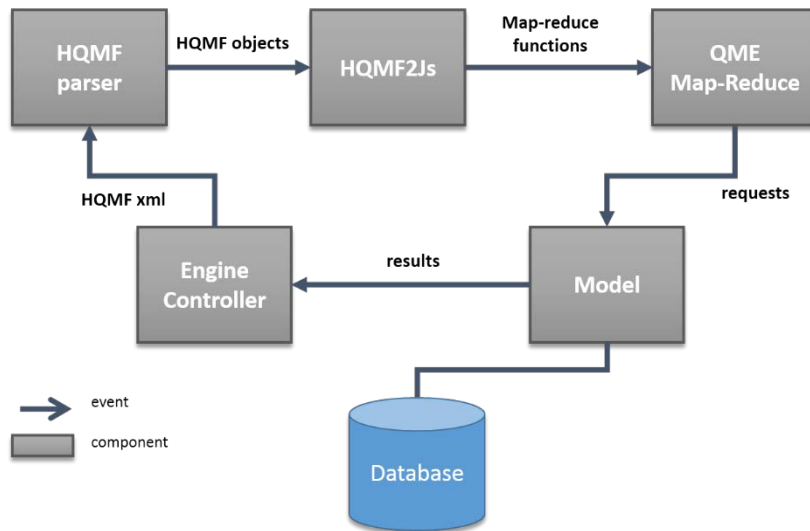


Figure 43: QME component using schema less database

The advantages of using a schema less database technology could be: 1) less response time, 2) data independency. Yet, Mongo DB does not offer the level of security and transaction lacking intelligence offered by most known vendor Relational DB systems.

The technology chosen for data management is SQLite [22], specifically the SQLite 3 version [23]. It is actually a light version of SQL supporting many of its features. It is used for applications in small devices, such as mobile phones or tablets. SQLite is not a client-server system, it is implemented as a library that can be included in an application. As a result, the required configuration is simple. It also stores the entire database into a disk file, which allows SQLite to process data that fit into the main memory, extremely fast.

#### 8.2.4. Data

During this project, there was an attempt to come up with a dataset which can be used in this implementation. Unfortunately, none of the stakeholders had real health dataset available to provide us. However, we obtained useful information to create our own dataset based on the following three sources:

1. Clinical Measures
2. The Preventive Medicine indicators
3. Claims Indicators

There are already international clinical measures which are defined using the Clinical Practice Guidelines. These cover a wide area of clinical issues such as medication, diagnosis, encounters, problems and procedures.

The Preventive Medicine gave us some descriptions of the used indicators. These indicators are related to obesity, diabetes, or quitting smoking. As a consequence the created dataset also contains records regarding the Preventive Medicine indicators.

In addition, the last months I was given a dataset sample from an H2H project, the Banner claims Reporting Project. This dataset includes information of the total inpatient cost of patient encounters. Appropriate HQMF documents are created for requesting data such as the average inpatient cost during a month.



### 8.2.5. QME – Object Relational Mapping

In the QueryGeneration component of QME system, the query generation takes place for the measure specified in the HQMF document input.

During the development the queries are not produced as pure SQL code. In Ruby on Rails queries are built in a higher level query language, namely the Object Relational Mapping (ORM).

ORM is a programming technique in which a data Model is transformed into data objects. For instance, a DB table *patients* is mapped to the Model class *Patient*. As a result, we have to create an object and assign it to a variable. In addition, objects can be cached in memory, reducing load on the database.

The biggest advantage of using ORM is the fact that there is no need for developers to write code specific to a particular database. It allows the developers to start a project using SQLite and later on migrate to MySQL or PostgreSQL.

On the other hand, working with ORM frameworks requires a learning curve for developers. Moreover, some actions such bulk insert, update, or delete are slower when implemented using ORM. In that case, it is more efficient to use native SQL queries.

All in all, using ORM simplifies a lot the process of query generation. In addition an open source tool, is called the Algebra of Relational query operators (Arel) [24], is used. It is a SQL Abstract Syntax Tree (AST) manager for Ruby, developed on top of ORM. It is a framework which simplifies the generation of complex SQL queries.

## 8.3 Conclusions

Regarding the technology choices for this implementation there were no specific limitations. The SQLite manages 12 tables in this implementation and the maximum number of joins in a query was 5, both of which are below the limitations of SQLite. However the response time varies based on the complexity of the query and the execution of groupings. The response time range is between 0.134 msec and 10 sec. The application runs in a system model of Intel(R) Core (TM) i5-4570 CPU @ 3.20 GHz.

■

# 9. Verification & Validation

This chapter discusses the verification and the validation of UQMEE for HSDP. Firstly it describes the main aspects that are verified. Secondly, it presents the validation of the system during the query generation process.

## 9.1 Introduction

The UQMEE main functionality is the execution of quality measures. A quality measure is executed on a known dataset in a database. For the execution of a quality measure, we have to assert two main points 1) that the execution of quality measure gives correct results, 2) the engine delivers the expected behavior. The first point implies the verification of the system and the second its validation.

In the verification process of the system, it has to be assured: 1) the correct usage of the data objects and 2) the correct query execution on the known dataset. For the validation, a number of test cases are applied in order to prove the engine's proper operations. The following section gives details for the aforementioned processes.

## 9.2 Verification

In this section, the verification process is described. This process verifies that the executed results of the quality measures are as it is expected to be. In other words, necessary conditions are checked, based on HQMF measures.

### 9.2.1. Verification of the executed measures

The UQMEE provides measures over a data set with elements that are relevant to these measures. The context of the data model in this project is created regarding the used indicators. We assume the number of patients belonging to a specific category. In that way, we can compare the computed results with the expected ones, Figure 44.

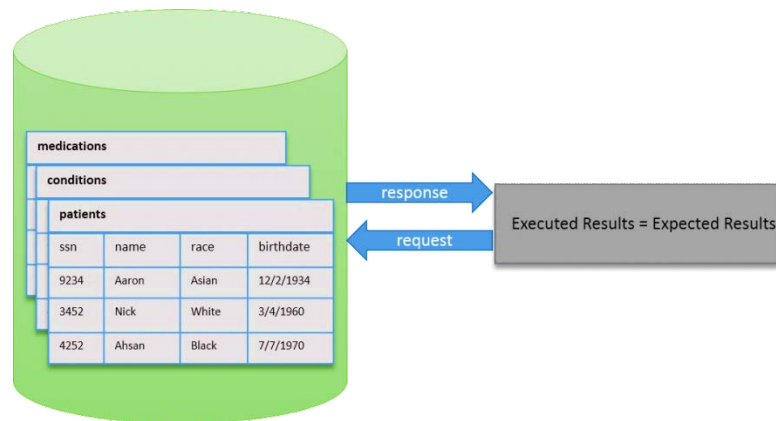


Figure 44: Verification of executed measures

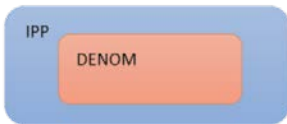
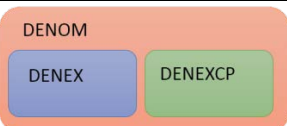
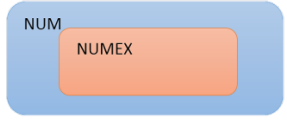
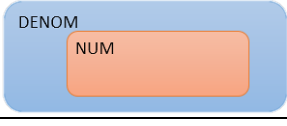

The computed results are given by each population type based on HQMF, Table 18.

Table 18: Population types

Population type	Description
Initial Patient Population (IPP)	It is the initial group of patients
Denominator (DENOM)	It is the same or a subset of initial patient population
Denominator Exclusion (DENEX)	It is a subset of the denominator with patients to be excluded from denominator
Denominator Exception (DENEXCP)	It is a subset of the denominator excluded patients from denominator, this set is used as numerator in case the numerator is equal to zero
Numerator (NUM)	It is a subset of the denominator
Numerator Exclusions (NUMEX)	It is a subset of the numerator

After the quality measure results are extracted as population types, there is need to compute a ratio or a proportion using these populations. Based on HQMF standards, there is a set of conditions that can be applied for checking, Table 19.

Table 19: Condition checks of executed measures

Conditions sets		
1	$IPP \cup DENOM = IPP, \quad IPP \neq \emptyset$	
2	$DENOM \cup DENEX \cup DENEXCP = DENOM, \quad DENEX \geq \emptyset \text{ and } DENEXCP \geq \emptyset$	
3	$NUM \cup NUMEX = NUM, \quad NUMEX \geq \emptyset$	
4	$DENOM \cup NUM = DENOM, \text{ proportion measure}$	
5	$NUM \cap DENOM = \emptyset, \text{ ratio measure}$	

Regarding Table 19 for each case, we note the following situations:

1. The Initial Patient Population set should be always equal to or greater than Denominator. The Denominator is a subset of Initial Population.
2. The Denominator Exclusion and Denominator Exceptions are subset of the Denominator.
3. The Numerator Exclusion is a subset of the Numerator
4. For a proportion measure the Numerator is a subset of the Denominator
5. For a ratio measure the Numerator and Denominator intersection is empty

In the current implementation, there is no case that the numerator or initial patient population equals to zero. However, the above conditions are applied to ensure that

the numeric results are appropriate to compute a ratio. As a result the cases below are covered by the dataset for the used indicators.

- $IPP > 0$
- $DENOM > 0$
- $NUM > 0$
- $NUMEX \geq 0, DENEX \geq 0, DENEXP \geq 0$

What are not covered in the verification phase are the cases that the IPP or DENOM or NUM equals to zero. In case the result of one of these populations is zero, there are two cases; 1) the quality measure cannot be computed as a ratio, 2) in case the NUM is zero and the quality measure includes a DENEXCP, in that case the DENEXCP population is used as the NUM, and then the quality measure is able to be executed.

### 9.3 Validation

The validation is performed executing the test cases (indicators) to the system demonstrating the system behavior to a client or user who understands the HQMF input and the expected output. These test cases should check the following functions:

1. A valid HQMF indicator should be executed
2. An invalid HQMF indicator should not be executed
3. A valid QRDA document should be executed
4. An invalid QRDA document should not be executed

Table 20 lists possible states and their expected behavior.

Table 20: HQMF-QRDA possible states

State	Expected behavior
(1) HQMF valid	The HQMF syntax is valid. The system sends the HQMF document to HQMF parser
(1.1) HQMF executable	After parsing, it is checked whether the elements of HQMF are available on the data set or not. In case the elements are available the HQMF is executable.
(1.2) HQMF not executable	After parsing, it is checked whether the elements of HQMF are available on the data set or not. In case the elements are not available the HQMF is not executable.
(2) HQMF invalid	The system informs the user that it is not possible to execute the specific indicator because the HQMF is not in proper syntax.
(3) QRDA valid	The system sends the QRDA document to QRDA parser
(4) QRDA invalid	The system informs the user that the results cannot be extracted because the QRDA syntax is not proper

Regarding the state (1.1) and (1.2), a quality measure can be executed when the required data elements are present in the data model. For instance, in case a criterion refers to an attribute that is not present in the data model, the executed query will give a SQL exception. In this case, the user will get a message indicating that the current indicator is not able to be executed and that a specific element is missing.

Table 21 presents the test case indicators.

Table 21: Indicator titles

Indicator Title	
A	Obese Patients who achieved a weight loss of more than 2%

B	Obese Patients who participated in the Physical Activity Group and realized a weight loss of more than 5%
C	Acute myocardial infarction (AMI) patients who were prescribed aspirin at hospital discharge
D	Ischemic stroke patients with atrial fibrillation/flutter who were prescribed anticoagulation therapy at hospital discharge
E	Patients with elective vaginal deliveries or elective cesarean sections at $\geq 37$ and $< 39$ weeks of gestation
F	Median Time from ED Arrival to ED Departure for Discharged ED Patients
G	Hearing Screening Prior To Hospital Discharge
H	The average cost per occurrence for inpatient encounter during the year 2012
I	The average cost per patient for inpatient encounter during the year 2012
J	The average length of stay for encounters during 2012

In order for the execution of an indicator, all its elements should be mapped to the data elements in the data model. This is indicated by True in Table 22. It is shown that the indicator G cannot be executed as the element *diagnostic result* is not included in the data model.

Table 22: Indicators and their corresponding HQMF elements

Indicators/ data types	A	B	C	D	E	F	G	H	I
characteristics	True	True	True	True	True	True	True	True	True
medications			True	True	True				
conditions			True	True	True		True		
encounters	True	True	True	True	True	True			
costs								True	True
activities		True							
diagnostic results							False		
procedures				True	True		True		
measurements	True	True							
Transfers					True				
	✓	✓	✓	✓	✓	✓	✗	✓	✓

In the current implementation, all the above indicators except G, produces a valid HQMF document. However, for indicators A and B, it is checked what is the behavior of the system in case the HQMF is not valid. In this case the computation is not done and the user can select a new indicator.

For the validation of the QRDA, the syntax of the document is checked based on its schema. In this implementation, all the QRDA documents are passed as syntactically correct. There is no test case that checks the behavior of the system in case the document is not valid or an element of QRDA is missing.

■

# 10. Conclusions

This chapter gives an overview of the results of UQMEE for HSDP project. It presents possible alternatives and future extensions.

## 10.1 Results

The UQMEE is a web-based software solution providing a proof of concept quality measure execution engine using healthcare standards. It constitutes an input to the HSDP business group. The HSDP is a central data storage platform designed to become accessible from all the Philips Research groups in the future. The final goal of the HISS group is to build a unified quality measure execution engine on top of the HSDP. In that way, various health data analytics groups may apply their measures using one unified measure execution engine based on international healthcare standards.

### 10.1.1. Results based on the requirements

Based on the requirements in Chapter 5, this section describes the way how the software solution meets the corresponding requirements. The requirements are presented again in the next tables, Table 23, Table 24, and Table 25.

Table 23: User Requirements

ID	Description
UR_1	The user should be able to select from a given list the indicator he needs to measure
UR_2	The user should be able to submit a request for the measurement of a specific indicator
UR_3	The user should be able to receive the resulting measurements through graphs
UR_4	The user should be informed by the system about the possibly unsuccessful measurement of a specific indicator.

The UQMEE provides the Quality Measure Interface sub system (QMI), in which the UI is developed. The user is able to select one of the indicators by selecting the title of the corresponding quality measure. After the submission, the user is able to view the results through graphs. Figure 45 shows the UI of QMI presenting the aggregated results of the quality measure “Acute myocardial infarction (AMI) patients who are prescribed aspirin at hospital discharge.”

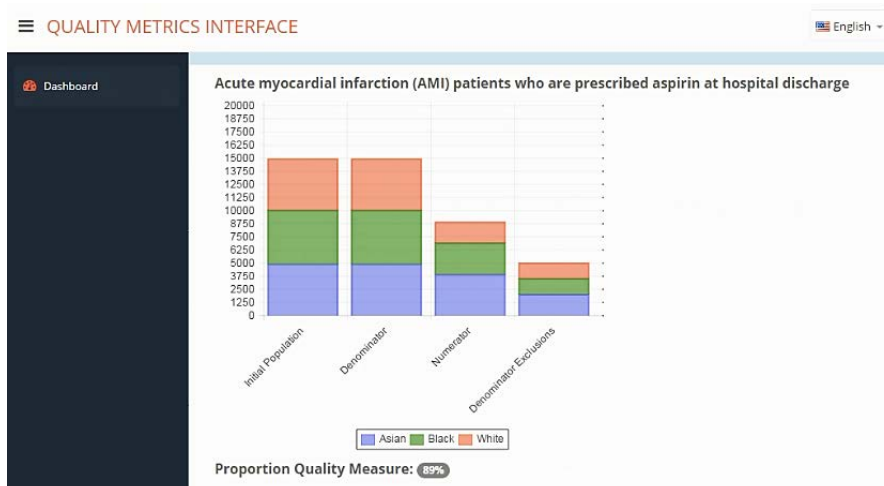


Figure 45: UI of QMI

A quality measure can be executable or not based on the data elements that are stored in the data model. In case an element is missed, for instance vital signs, it cannot be mapped to the data model. As a result, the system does not execute queries but sends a message to the user indicating the missing elements.

Table 24: Functional Requirements

ID	Description
F1	The UQMEE for HSDP should provide an HQMF based measurement engine for a set of indicators formulated in terms of HQMF standards
F2	The HQMF documents should be parsed using the HQMF parser from Health data Standards library set
F3	The data should be stored in a structured way based on HQMF data criteria definitions
F5	The UQMEE should provide an extensible data model regarding the HQMF elements, such as new procedures or problems

Regarding the functional requirements in Table 24, the F1 requirement is met by using HQMF documents in which the quality measures are defined. The documents are used in release 1, 2 or 2.1 from Centers for Medicare & Medicaid Services (CMS). The F2 requirement is met by using the HQMF parser of Health-data-standards, an open source library which is backwards compatible to HQMF R1, R2 and R2.1. The F3 requires the data to be compatible with HQMF data. For this case, data is stored in groups of medications, diagnosis, procedures and patients' characteristics see data model in Chapter 6 - section 6.4.. The extensibility of the data model (F5 requirement) is a matter of the data elements relations. For this case, the data model has as central relational table the encounters, each procedure, medication or new health data element can be defined as a new relation to the encounter, see Chapter 6 - section 6.4 .

Regarding the non-functional requirements, Table 25, the UQMEE tries to give a clear design giving two sub systems the QMI and QME. In that way, the system is flexible and decoupled. The first sub-component is focusing on the UI and the second in the quality measure engine (NF1). As it is mentioned, the NF2 requirement is met by using a compatible HQMF parser of Health-data-standards.

Table 25: Non-functional requirements

ID	Description
NF1	The design of UQMEE should clearly illustrate the way the HQMF information is translated to executable queries on a set of health data. In that way the HSDP business group can acquire proper insights that it has to take into consideration for their environment
NF2	The UQMEE for HSDP should support all the previous versions of HQMF releases, (R1, R2, R2.1)

### 10.1.2. Conclusions

UQMEE exposes the value of using healthcare standards in Philips Research through its software solution. The set of healthcare standards gives a uniform way for the exchange and use of health data. It is analogous to a common health language. It supplies a basis with the same semantics, forms and context on health data. Health data is explicitly organized into groups based on its functions. For example, data related to medications are organized including the reason for prescription, adverse effects or allergies. As a result, the extensibility of quality measures is easier, and multiple groups can execute their measure using the UQMEE.

UQMEE is a unified quality measure execution engine in which quality measures are structured using HQMF health standards issued by International HL7. It allows a medical practitioner to select a specific clinical or claims indicator to calculate the performance of a healthcare provider. For instance, a clinical practitioner can see how many heart disease patients were prescribed with aspirin after discharge from a hospital, how many of them were allergic to aspirin, how many left the hospital against medical advice or how many were transferred to another institute.

To execute a measure, we need to apply the measure on a suitable dataset. In this solution the following achievements are done, 1) a dataset was created for testing the functionality that can be reused, 2) a working prototype is built for an engine to compute measures and display results of the computation.

Specifically the dataset is created based on: 1) clinical indicator examples from Centers for Medicare & Medicaid Services (CMS) [25], an organization developed health and human services, 2) clinical indicator examples of Preventive Medicine project, and 3) examples of claims indicators.

The working prototype provides an end-to-end functionality. The solution consists of steps in which a HQMF document can be transformed into executable SQL queries. The system is modular and other technology can be applied. For instance, in case another technology is applied for building queries, in the system only the package Query Generation needs to be changed, see chapter 7, section 7.1.

In this project, other similar tools were investigated. UQMEE uses open source parsers for HQMF and QRDA, which are Ruby libraries. These tools can bring insights regarding alternative approaches. In addition, in this solution a relational data model is used, as most of healthcare providers use RDBMS. The UQMEE generates queries based on a transformation of HQMF to SQL-based queries.

An alternative technology of query generation could involve a domain specific language. The HQMF to SQL-based queries transformation can be also implied as a text to text transformation, text HQMF elements to text SQL statements. For text to text transformation other tools can be also used, such as Xtext [26]. This is a framework for developing domain specific languages. This tool allows developers to create their own parser using a model to model transformation, HQMF to SQL model. It can be used in Eclipse environment.

Another alternative technology choice could refer to the database system. The database system is a choice which depends on the performance needs and the data relations. The schema-less database can be ideal in case data is stored independently as patient records. It is also suitable for handling big data, due to owning efficient map-reduce functions. Patient records could be structured using the QRDA Cat 1 health standard. QRDA Cat 1 contains information such as medications, measurements, diagnosis, or encounters on patient level.

All in all, based on future HSDP needs, there is a broad area for selecting a proper technology. Since HSDP is expected to serve as a common data storage platform of Philips Research business groups, new performance needs will be exposed.



## 10.2 *Future work*

Taking a further step of QMEE, possible future work could be:

1. Integration to HSDP

As the main goal of this project is to initiate a measuring tool on top of HSDP, further work can be done in this direction. The HSDP group could provide its available interfaces used for connection to the engine. It is also investigating over the proper technologies to use. For instance, a new engine could be developed in Java. In that way, various groups inside Philips can compute their quality measures using one standard-based engine. Their measures can be scalable and compliant with international standards, as well as they can use a tested and valid tool for their measures.

2. Providing an authoring measurement tool

Another future task could be the development of a measurement authoring tool. The authoring of measures capability allows users to generate new indicators through the UI. The authoring tool gives the freedom to users such as clinicians or data analysts to produce quality measures based on their needs of their group and the aspects they are interested in measuring.

In the current solution, the available indicators are fixed. The authoring tool should be based on HQMF and independent of the data set available. The user should be able to define what they want to measure. The engine should also be parametrized by the HQMF – Schema mappings and so be quite generic. Whether results are obtained or not, depends on the available data set.

Developing a measurement authoring tool requires time and a proper UI design, so a user without having knowledge of HQMF standards can create through simple concrete steps an indicator based on his need.

3. Enabling of security features

In the current implementation, the UQMEE poses no restrictions on user access. In this software solution we built a prototype application for demonstrating a standard based quality measure engine. In real life situations, the UQMEE could be enhanced with security features controlling the level of user access on the data and protecting potentially sensitive patient information.

■

# 11. Project Management

Whereas the previous chapters describe the whole UQMEE architecture and implementation process, this chapter discusses the organization and the management of the project.

## 11.1 Introduction

This section outlines the management of this project. In order to present it, we give an overview of its main phases and the people involved.

The UQMEE delivers a unified quality measure execution engine for a known health dataset using HL7 - HQMF standards within 9-months. During this period, the main challenge was the understanding and the correct use of this kind of standards. To understand the health standards, it was necessary to study the HQMF and QRDA and experiment with various examples of using them. In this domain the only professional with a related background is my company supervisor, Asim Muhammad. He is considered to be a HL7 Specialist and his main concern was to ensure that we are using the standards properly. As a consequence, the communication with the supervisor was really crucial for the progress of this project.

The development of a quality measure execution engine requires an initial period of investigation and trying out existing technologies. Then the requirements of the project can be set, based on our needs and later on, we can continue with the design and implementation of the engine.

The main phases of this project can be roughly set as follows:

1. **Study and investigation of the domain:** study of the HL7 standards and investigation of proper technologies
2. **Requirements analysis:** define the scope of the project and its requirements
3. **Design:** create the architecture and design
4. **Implementation and testing:** development of UQMEE
5. **Documentation:** document and manage the work

These phases during the nine-month period are presented in the next section

## 11.2 Project time line

Figure 37 presents the time line of the project based on the aforementioned five phases. Each phase is described following the time line order.

**Study and investigation of the domain:** In the initial phase, there was a period of studying the domain of health standards. Specifically, the studied aspects are: 1) the value and use of quality measures, 2) HL7 main concepts, 3) HQMF concepts, 4) QRDA concepts and use. In order to reuse some of the existing tools for health standards, we investigated a set of open source projects, such as Cypress, popHealth, Bonnie and MAT. In addition, we also explored possible technology solutions for data storing and web development.

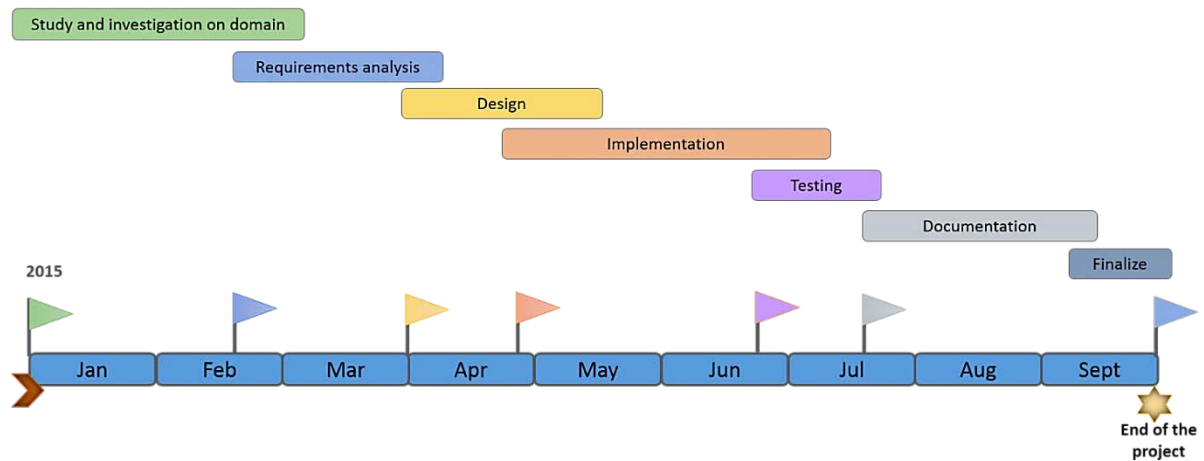


Figure 46: Project time line

**Requirements analysis:** In this phase, the scope and the requirements of the project are defined. At the beginning of the project, the initial requirements were different. For instance, the first requirements included the integration of UQMEE with HSDP. However, the HSDP was not available and later on this requirement was set out of scope. Also the possibility of developing an authoring measurement tool with the engine was investigated. This task was also set out of scope for this current implementation due to the lack of time. Finally, the first version of UQMEE is developed as a Ruby on Rails web-application.

**Design:** In the design phase, we modeled the components of the system trying to meet the expectations of the defined system requirements. We also sketched the architecture of the system.

**Implementation and testing:** The development of the UQMEE takes place in this phase. In order to choose the proper tools, investigation and trials of different technologies were applied. It was a real challenge to learn the Ruby on Rails framework and Ruby libraries, from scratch at such a fixed timeframe. The implemented engine was tested based on a known dataset.

**Documentation:** We documented initial concepts for our own understanding. We also wrote some draft documents for presenting progress in front of the supervisors. Preparation of presentations was also conducted in weekly and monthly periods.

**Finalize:** In the end of the project, some remaining tasks were conducted. These tasks were administrative work, cleaning code, and documentation, as well as preparing final presentation and demonstration.

### 11.3 *Communication*

During this nine-month project, once per week we scheduled a progress update meeting with the two company supervisors. Especially during the first months, we communicated daily, as they tried to guide on understanding the use of healthcare standards. In our weekly meetings we performed two main tasks 1) report on the progress of the current deliveries, and 2) presentation of results or findings.

Once per month, we scheduled a monthly project steering group meeting (PSGM) with both TU/e and company supervisors. In these meetings the subject was set in high- level description of the project progress. The main discussed points were:

- Presentation of the current tasks
- Presentation of the next monthly tasks
- Discussion over the project state

■

# 12. Project Retrospective

This chapter finalizes the documentation of the UQMEE project by presenting a reflection of the author on the project.

## 12.1 *Introduction*

This project was very interesting, as it introduced the vast domain of healthcare technologies over data analysis. During these months, we learned a lot of aspects regarding the quality of health care delivery. This project attempts to provide insight into healthcare quality. In that way, it is known which aspects of delivery is good, and which needs to improve in order to achieve good health.

## 12.2 *Design opportunities revisited*

In chapter 3 the following design criteria were considered for UQMEE.

- Ease of use
- Backward Compatibility
- Extensibility

We conclude that the above design opportunities are met in the UQMEE by the following statements:

**Ease of use:** we provide a UI which any non-technical user can use to measure quality. As far as for the usage of the components, the UQMEE is decomposed into two parts the QMI and QME, providing flexibility and loose coupling to the system. As a result, possible changes on the system can be done easily. Different technologies for front and back end can be applied without dependencies between them.

**Backward Compatibility:** we use for parsing the HQMF standard a HQMF parser which is backward compatible to R1, R2 and R2.1. This parser is already used from several health providers.

**Extensibility:** To further extensible of functions, the Factory Method design pattern has been applied. In this way, we can define an interface for creating different objects and functions that can be instantiated through sub classes. As a result, in case there is a new extension in the HQMF functionality a new object can be instantiated.

## 12.3 *Strong Points*

We present the following strong points for our approach:

**Balancing the exploration of a huge domain with a concrete prototype.** We believe that we singled out concepts of the domain that are relevant for this project.

**Fast learner:** In this project we worked with different tools. We had to learn Ruby on Rails framework and the usage of various Ruby libraries in a short period.

**Working incrementally:** We created a basic version of our system. Then we enhanced the proto-type incrementally focusing on different components per iteration

## ***12.4 Improvements Points***

We identified the following points that we could improve:

Organizing better the process of studying the HQMF, QRDA standards as well as the HL7 concepts. I think that we spent a lot of time to understand them theoretically. The organization of practical examples from the beginning of the project would be a better approach.

We underestimated the effort needed for the documentation. In the end of the project, even though we tried to provide an early documentation, it still took a lot of time to provide a clear description of all the UQMEE aspects.

■

# Appendix A

In this section the mappings between HQMF elements and SQL-based functions using Arel-ORM are described. This section is organized into four parts. The first part gives a generic description of the HQMF elements needed to be mapped to SQL expressions. The second part includes the SQL expressions using Arel/ORM syntax. The third part presents the HQMF elements and its expressions and the fourth part includes the mappings between the HQMF elements and Arel-ORM functions.

## Part 1

- **Temporally Related Information type Code**

The temporal related information element allows two acts to be related by the moment they occur with respect to each other. In terms of data criteria this implies that two data criteria can be related to each other using the section of temporally related information. Temporal calculations could be timing relations such as DURING or CONCURRENT.

For example, in case an A act (represented on data criterion) occurs during B, then:

**A DURING B** is true if all of the following are true:

- A.effectiveTime.low must be non-null
- A.effectiveTime.high must be non-null
- B.effectiveTime.low must be non-null OR null with a nullFlavor of NINF
- B.effectiveTime.high must be non-null OR null with a nullFlavor of PINF
- B.effectiveTime.low <= A.effectiveTime.low <= B.effectiveTime.high
- B.effectiveTime.low <= A.effectiveTime.high <= B.effectiveTime.high

If any of the above is false, DURING is false

Table 26: Temporal relations types

	<b>Concept Code</b>	<b>Print name</b>	<b>Definition</b>
1.	CONCURRENT	concurrent with	A relationship in which the source act's effective time is the same as the target act's effective time.
2.	DURING	occurs during	including end points, as defined in the act's effective times
3.	EAE	ends after end of	A relationship in which the source act's effective time ends after the target act's effective time.
4.	EAS	ends after start of	A relationship in which the source act's effective time ends after the start of the target act.
5.	EBE	ends before end	The source act ends before the end of the target act
6.	EBS	ends before start of	A relationship in which the source act's effective time ends before the start of the target act.
7.	ECW	ends concurrent with	A relationship in which the source act's effective time ends with the end of the target act's effective time.
8.	ECWS	ends concurrent with start	The source act ends when the target act starts

9.	EDU	ends during	A relationship in which the source act ends within the target act's effective time (including end points, as defined in the act's effective times)
10.	OVERLAP	overlaps with	A relationship in which the source act's effective time overlaps the target act's effective time in any way.
11.	SAE	starts after end of	A relationship in which the source act starts after the end of the target act.
12.	SAS	starts after start of	The source act starts after the start of the target act.
13.	SBE	starts before end	The source act starts before the end of the target act.
14.	SBS	starts before start of	A relationship in which the source act starts before the start of the target act.
15.	SCW	starts concurrent with	with A relationship in which the source act's effective time starts with the start of the target act's effective time
16.	SCWE	starts concurrent with end	The source act starts when the target act ends.
17.	SDU	starts during	A relationship in which the source act starts within the target act's effective time (including end points, as defined in the act's effective times)

- **Value**

The value element is used for representing a specific required value of a HQMF element. For instance, a value can represent the measure period, the length of stay in hospital or the age of a patient.

Table 27: Value element

<b>Value</b>			
Value Type	QTY(INT, AL,MO,PQ,RTO,TS,CO)	RE-	In this case it shall contain an expression which indicates how the value is computed

QTY: Abstract type quantity  
 INT: Integer  
 REAL: Real number  
 MO: Monetary amount  
 PQ: Physical quantity  
 RTO: Ratio  
 TS: Point in time  
 CO: Coded ordinal  
 IVL: Interval  
 ST: Character string

Table 28: Attributes of Value element

<b>Value</b>	<b>Description</b>	<b>Additional elements</b>
--------------	--------------------	----------------------------



Value Type	Type can be IVL_TS or PIVL_TS or any other kind	
low	The actual value	LowClosed, indicates whether the low value is included into the interval
high	The actual value	HighClosed, indicates whether the high value is included into the interval
width	The actual value	Type, indicates the type of the value and the unit the unit kind

- **Excerpt – Type Code**

This is used to indicate how the target of the relationship will be a filtered subset of the total related set of targets. It is used when there is a need to limit the number of components to the first, the last, the next, the total, the average, or some other filtered or calculated subset.

<b>Value Constraints (used when the target is Observation Criteria only)</b>	
MAX maximum	Selects the observation with the largest value
MIN minimum	Selects the observation with the smallest value
<b>Time Constraints</b>	
LAST, expected last	Selects the act that is expected to occur the farthest in the future
FIRST, first known	Selects the first known occurrence of the act
<b>Summaries</b>	
SUM, summary	Represents a summary of all acts. The effective Time represents the outer boundary of all occurrences, repeat Number represents the total number of repetitions, etc.

## Part 2

Table 29: Arel-ORM functions

<b>Comparison Operators</b>		
	Arel/ORM function	Symbol
1a	.eq	=
2a	.not_eq	!=
3a	.lt	<
4a	.gt	>
5a	.lteq	<=
6a	.gteq	>=
7a	.in	IN
8a	.created_at	IN / BETWEEN
<b>Aggregation Functions</b>		
	Arel/ORM function	Expression
10a	.having	HAVING
11a	.sum	SUM
12a	.average	AVG
13a	.minimum	MAX
14a	.maximum	MIN
15a	.count	COUNT
16a	.order	ORDER
<b>Logical Operators and other conditions</b>		
	Arel/ORM function	Expression
17a		

18a	.and	AND
19a	.not	NOT
20a	.or	OR
21a	.where	WHERE, it is used as an AND Statement
22a	.group	GROUP BY
23a	.first	Choose the first table item
24a	.last	Choose the last table item

### Part 3

Table 30: HQMF elements

<b>Comparison Operators</b>		
	HQMF Comparison of 2 events, A and B	Expression
	Checking the following statements should be taken in any case A.effectiveTime.low must be non-null A.effectiveTime.high must be non-null B.effectiveTime.low must be non-null OR null with a nullFlavor of NINF B.effectiveTime.high must be non-null OR null with a nullFlavor of PINF	
1b	CONCURRENT	B.effectiveTime.low = A.effectiveTime.low B.effectiveTime.high = A.effectiveTime.high
2b	DURING	B.effectiveTime.low <= A.effectiveTime.low <= B.effectiveTime.high B.effectiveTime.low <= A.effectiveTime.high <= B.effectiveTime.high
3b	EAE	A.effectiveTime.high < B.effectiveTime.high
4b	EAS	A.effectiveTime.high > B.effectiveTime.low
5b	EBE	A.effectiveTime.high < B.effectiveTime.high
6b	EBS	A.effectiveTime.high < B.effectiveTime.low
7b	ECW	A.effectiveTime.high = B.effectiveTime.high
8b	ECWS	A.effectiveTime.high = B.effectiveTime.low
9b	EDU	A.effectiveTime.high = B.effectiveTime.low
10b	OVERLAP	A.effectiveTime.high > B.effectiveTime.high A.effectiveTime.low > B.effectiveTime.low
11b	SAE	A.effectiveTime.low > B.effectiveTime.high
12b	SAS	A.effectiveTime.low > B.effectiveTime.low
13b	SBE	A.effectiveTime.low > B.effectiveTime.low
14b	SBS	A.effectiveTime.low < B.effectiveTime.low
15b	SCW	A.effectiveTime.low = B.effectiveTime.low
16b	SCWE	A.effectiveTime.low >= B.effectiveTime.high
17b	SDU	A.effectiveTime.low >= B.effectiveTime.low A.effectiveTime.low <= B.effectiveTime.high
<b>HQMF Aggregation Functions</b>		
		Expression
18b	COUNT	Count
19b	SUM	Summary
20b	AVERAGE	Average
21b	STDEV.S	Standard deviation Sample
22b	Deviation.	Deviation
23b	VARIANCE.S	Variance Sample
24b	STDEV.P	Standard deviation Population
25b	VARIANCE.P	Variance Population
26b	MIN	Minimum
27b	MAX	Maximum
28b	MEDIAN	Median
<b>HQMF Logical Operators</b>		
		Expression

29b	AllTrue	AND
30b	AllFalse	NOT OR
31b	AtLeastOneTrue	OR
32b	AtLeastOneFalse	NOT AND
33b	OnlyOneTrue	XOR
34b	OnlyOneFalse	XOR

<b>HQMF Value element</b>		
35b	Low and lowClosed= true	<=
36b	Low and lowClosed = false	<
37b	high and highClosed = true	>=
38b	high and highClosed = false	>

#### Part 4

Table 31: Comparison operators' mappings

<b>HQMF</b>	<b>Arel/ORM</b>
1b	1a
2b	5a
3b	3a
4b	4a
5b	3a
6b	3a
7b	1a
8b	1a
9b	1a
10b	3a
11b	3a
12b	3a
13b	3a
14b	4a
15b	1a
16b	5a
17b	5a and 6a

Table 32: Aggregation operators' mappings

<b>HQMF</b>	<b>Arel/ORM</b>
18b	15a
19b	11a
20b	12a
21b	No ready functions
22b	No ready functions
23b	No ready functions
24b	No ready functions
25b	No ready functions
26b	13a
27b	14a
28b	No ready functions

Table 33: Logical operators' mappings

<b>HQMF</b>	<b>Arel/ORM</b>
29b	18a
30b	19a and 20a
31b	20a
32b	19a and 18a
33b	No ready functions
34b	No ready functions

Table 34: Value elements mappings

<b>HQMF Value</b>	<b>Arel/ORM</b>
35b	5a
36b	3a
37b	6a
38b	4a

■

# Appendix B

## Data Criteria Mappings

The following section presents the flow process of the generation of SQL queries. The inserted xml file is parsed through the HQMF parser from the Health Data Standards library. The results of the parser are JSON objects, which have been mapped to the HQMF model R2. Then the extracted JSON objects are mapped into ORM-Arel objects which generate the final SQL query.

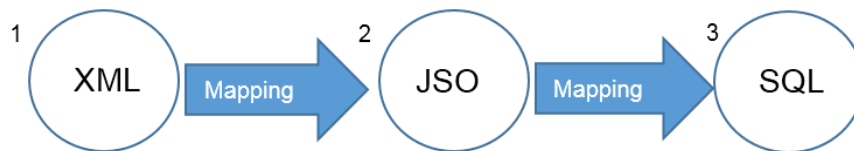


Figure 47: Flow process of queries execution

To clarify the way the mappings are being executed three example cases are illustrated.

The first example shows how a data criterion declares the required population group using only the basic elements for structuring a data criterion.

The basic elements for structuring a data criterion are the elements such as the id, the title and definition which can define a criterion without further references to another criterion.

The second and the third example shows a declaration of data criterion using an out-bound information from other criteria and temporally related information to other criteria using timing relations like during, concurrent etc. These three cases can be combined declaring a criterion in any case.

	<b>Data criterion in xml format</b>
<ol style="list-style-type: none"> <li>1.</li> <li>2.</li> <li>3.</li> <li>4.</li> <li>5.</li> </ol>	<pre> &lt;entry typeCode="DRIV"&gt;   &lt;localVariableName value="Obese"/&gt;   &lt;observationCriteria moodCode="EVN" classCode="OBS"&gt;     &lt;templateId&gt;       &lt;item root="root_templateId number" identifierName="root_identifierName"/&gt;     &lt;/templateId&gt;     &lt;id root="number_root_id" extension="Obese"/&gt;     &lt;code code="test" displayName="Diagnosis" codeSystem="2.16.840.1.113883.6.1"/&gt;     &lt;text value="Diagnosed Obese "/&gt;     &lt;statusCode code="active"/&gt;     &lt;definition&gt;       &lt;criteriaReference moodCode="EVN" classCode="OBS"&gt;         &lt;id root="test" extension="diagnosis"/&gt;       &lt;/criteriaReference&gt;     &lt;/definition&gt;   &lt;/observationCriteria&gt; &lt;/entry&gt; </pre>
	<b>Mapping between xml and parsed objects</b>
<ol style="list-style-type: none"> <li>1.</li> <li>2.</li> <li>3.</li> <li>4.</li> <li>5.</li> </ol>	<pre> id extension =&gt; title text value =&gt; description status code =&gt; status definition-criteria reference- id extension =&gt; definition In the parser there is a mapping for the case that definition = "diagnosis"   "diagnosis":{     "title":"diagnosis",     "category":"conditions",     "definition":"diagnosis",     "status":"",     "sub_category":"",     "hard_status":false,     "patient_api_function":"allProblems",     "not_supported":false},   } category =&gt; type </pre>
	<b>Data criterion objects after be parsed</b>
<ol style="list-style-type: none"> <li>1.</li> <li>2.</li> <li>3.</li> <li>4.</li> <li>5.</li> </ol>	<pre> "Obese": {   "title": "Obese",   "description": "Diagnosed Obese",   "children_criteria": [   ],   "type": "conditions",   "definition": "diagnosis",   "status": "active",   "hard_status": false, } </pre>
	<b>Mapping between parsed criterion objects and SQL objects</b>
	<pre> Table =&gt; type Field table =&gt; definition Field content =&gt; title </pre>
	<b>ORM – Arel script</b>
	<pre> type = Arel::Table.new(:type) query = type.project(Arel.sql('*')).where(type[:definition].eq('title')) For this example, the corresponding query will have the following form: query = conditions.project(Arel.sql('*')).where(conditions[:diagnosis].eq('Obese')) </pre>

	ORM to SQL
	⇒ SELECT * FROM conditions WHERE conditions.diagnosis = 'Obese'

The second example shows how a data criterion declares the required population group included an out-bound relationship with another criteria defining related values.

In the example, the data criterion consists of an encounter criteria declaring the Inpatient Encounter event, and an outbound relation with an observation criterion in which the encounter length of stay is defined.

Query Generation Example 2	
Data criterion in xml format	
	<pre> &lt;entry typeCode="DRIV"&gt;   &lt;localVariableName value="OccurrenceAEncounterInpatient1_precondition_9"/&gt;   &lt;encounterCriteria moodCode="EVN" classCode="ENC"&gt;     &lt;templateId&gt;       &lt;item root="2.16.840.1.113883.10.20.28.3.5" identifierName="encounter, performed template"/&gt;     &lt;/templateId&gt;     &lt;id root="2.16.840.1.113883.3.100.1" extension="OccurrenceAEncounterInpatient1_precondition_9"/&gt;     &lt;code xsi:type="CD" valueSet="2.16.840.1.113883.3.666.5.307"&gt;       &lt;displayName value="Encounter Inpatient"/&gt;     &lt;/code&gt;     &lt;text value="Encounter, Performed: Encounter Inpatient"/&gt;     &lt;definition&gt;       &lt;criteriaReference moodCode="EVN" classCode="ENC"&gt;         &lt;id root="2.16.840.1.113883.3.100.1" extension="encounter"/&gt;       &lt;/criteriaReference&gt;     &lt;/definition&gt;     &lt;outboundRelationship typeCode="SUBJ"&gt;       &lt;observationCriteria classCode="OBS" moodCode="EVN"&gt;         &lt;id root="OccurrenceAEncounterInpatient1_precondition_9_LENGTH_OF_STAY"/&gt;         &lt;code code="183797002" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT"&gt;           &lt;displayName value="Length of Stay"/&gt;         &lt;/code&gt;         &lt;value xsi:type="IVL_PQ" highClosed="true" &gt;           &lt;low nullFlavor="NINF"/&gt; &lt;high value="120" unit="d"/&gt;         &lt;/value&gt;       &lt;/observationCriteria&gt;     &lt;/outboundRelationship&gt;     &lt;outboundRelationship typeCode="OCCR"&gt;       &lt;localVariableName controlInformation- Root="ENCOUNTER_PERFORMED_ENCOUNTER_INPATIENT" controlInformationExtension="A"/&gt;     &lt;/outboundRelationship&gt;     &lt;criteriaReference classCode="OBS" moodCode="EVN"&gt;       &lt;id root="2.16.840.1.113883.3.100.1" extension="OccurrenceAEncounterInpatient1"/&gt;     &lt;/criteriaReference&gt;   &lt;/encounterCriteria&gt; </pre>
1.	
2.	
3.	
4.	
5.	
6.	
7.	

	<code>&lt;/encounterCriteria&gt;</code>
	<b>Mapping between xml and parsed objects</b>
<ol style="list-style-type: none"> <li>1. Code -&gt; Display name value =&gt; title</li> <li>2. text value =&gt; description</li> <li>3. definition-criteria reference- id extension =&gt; definition</li> <li>4. In the parser there is a mapping for the case that definition = "encounter" <pre> "encounter":{     "title":"encounter",     "category":"encounters",     "definition":"encounter",     "status":"",     "sub_category":"",     "hard_status":false,     "patient_api_function":"encounters",     "not_supported":false}, category =&gt; type </pre> </li> <li>5.</li> <li>6. outbound relationship -&gt; value =&gt; field values</li> <li>7. outbound relationship -&gt;local variable =&gt; specific_occurrence_const Out bound relationship -&gt; Criteria reference =&gt; source_data_criteria</li> </ol>	
<ol style="list-style-type: none"> <li>1.</li> <li>2.</li> <li>4.</li> <li>3.</li> <li>6.</li> <li>7.</li> <li>5.</li> </ol> <pre> "OccurrenceAEncounterInpatient1_precondition_9": {   "title": "Encounter Inpatient",   "description": "Encounter, Performed: Encounter Inpatient",   "type": "encounters",   "definition": "encounter",   "status": "performed",   "hard_status": false,   "negation": false,   "specific_occurrence": "A",   "specific_occurrence_const": "ENCOUNTER_PERFORMED_ENCOUNTER_INPATIENT",   "source_data_criteria": "OccurrenceAEncounterInpatient1",   "variable": false,   "field_values": {     "LENGTH_OF_STAY": {       "type": "IVL_PQ",       "high": {         "type": "PQ",         "unit": "d",         "value": "120",         "inclusive?": true,         "derived?": false       }     }   } } </pre>	
	<b>Mapping between criterion object and SQL objects</b>
	<p>Table =&gt; type  Field table =&gt; definition  Field content =&gt; title  Additional field value =&gt; field_values  Field value content =&gt; value  Comparison Operator =&gt; inclusive? &amp; low/high  Field unit =&gt; unit</p>
	The corresponding query from the extracted objects in ORM – Arel script
	<code>type = Arel::Table.new(:type)</code>



	<p>Arel comparison function = .lteq (less than equal to)  query = type.project(Arel.sql('*')).where(type[:definition].eq('title').and(type[:field_values].lteq(value)))  <i>For this example, the corresponding query will have the following form:</i>  query = conditions.project(Arel.sql('*')).where(encounters[:encounter].eq('Encounter Inpatient').and(encounters[:length_of_stay].lteq(120)))</p>
	ORM to SQL
	⇒ SELECT * FROM encounters WHERE encounters.encounter = ' Encounter Inpatient' AND encounters.length_of_stay <=120

In the third example, a data criterion is defined having a temporally timing related information with the measure period.

Query Generation Example 3	
	Data criterion in xml format
1. 2. 3. 4. 5. 6.	<pre> &lt;entry typeCode="DRIV"&gt;   &lt;localVariableName value="OccurrenceAOccurrenceAEncounterInpatient1_11"/&gt;   &lt;encounterCriteria moodCode="EVN" classCode="ENC"&gt;     &lt;templateId&gt;       &lt;item root="2.16.840.1.113883.10.20.28.3.5" identifierName="encounter, performed template"/&gt;     &lt;/templateId&gt;     &lt;id root="2.16.840.1.113883.3.100.1" extension="OccurrenceAEncounterInpatient1_11"/&gt;     &lt;code xsi:type="CD" valueSet="2.16.840.1.113883.3.666.5.307"&gt;       &lt;displayName value="Encounter Inpatient"/&gt;     &lt;/code&gt;     &lt;text value="Encounter, Performed: Encounter Inpatient"/&gt;     &lt;definition&gt;       &lt;criteriaReference moodCode="EVN" classCode="ENC"&gt;         &lt;id root="2.16.840.1.113883.3.100.1" extension="encounter"/&gt;       &lt;/criteriaReference&gt;     &lt;/definition&gt;     &lt;temporallyRelatedInformation typeCode="EDU"&gt;       &lt;criteriaReference moodCode="EVN" classCode="OBS"&gt;         &lt;id root="2.16.840.1.113883.3.100.1" extension="MeasurePeriod"/&gt;       &lt;/criteriaReference&gt;     &lt;/temporallyRelatedInformation&gt;     &lt;outboundRelationship typeCode="OCCR"&gt;       &lt;localVariableName controlInformation- Root="ENCOUNTER_PERFORMED_ENCOUNTER_INPATIENT" controlInformationExtension="A"/&gt;       &lt;criteriaReference classCode="OBS" moodCode="EVN"&gt;         &lt;id root="2.16.840.1.113883.3.100.1" extension="OccurrenceAEncounterInpatient1"/&gt;       &lt;/criteriaReference&gt;     &lt;/outboundRelationship&gt;   &lt;/encounterCriteria&gt; &lt;/entry&gt; </pre>
	Mapping between xml and parsed objects
1. 2. 3.	<p>1. Display name value =&gt; title  2. text value =&gt; description  3. In the parser there is a mapping for the case that definition = "encounter"</p> <pre> "encounter":{   "title":"encounter",   "category":"encounters",   "definition":"encounter", </pre>

<p>4. 5. 6.</p>	<pre> "status": "", "sub_category": "", "hard_status": false, "patient_api_function": "encounters", "not_supported": false}, category =&gt; type </pre> <p>definition-criteria reference- id extension =&gt; definition temporally Related Information =&gt; temporal_references outbound relationship -&gt; criteria reference =&gt; source_data_criteria</p>
<p>1. 2. 3. 4. 6. 5.</p>	<pre> "OccurrenceAEncounterInpatient1_11": { "title": "Encounter Inpatient", "description": "Encounter, Performed: Encounter Inpatient", "code_list_id": "2.16.840.1.113883.3.666.5.307", "type": "encounters", "definition": "encounter", "status": "performed", "hard_status": false, "negation": false, "specific_occurrence": "A", "specific_occurrence_const": "ENCOUNTER_PERFORMED_ENCOUNTER_INPATIENT", "source_data_criteria": "OccurrenceAEncounterInpatient1", "variable": false, "temporal_references": [ { "type": "EDU", "reference": "MeasurePeriod" } ] } </pre>
	<p>Mapping between criterion object and SQL objects</p>
	<p>Table =&gt; type Field table =&gt; definition Field content =&gt; title Additional temporal reference field (Event A) =&gt; temporal_references -&gt; reference Comparison Operator =&gt; temporal_references -&gt; type Additional temporal reference field (Event B) =&gt; source data criterion, which has the same attributes as the current criterion , which has start and end point</p>
	<p>The corresponding query from the extracted objects in ORM – Arel script</p>
	<pre> type = Arel::Table.new(:type) Comparison Operator =&gt; Ends During =&gt; using the &lt;= or &gt;= operators query = type.project(Arel.sql('*')).where(type[:definition].eq('title').and('encounters.end_point &gt;= MeasurePeriod.start AND encounters.end_point &lt;= MeasurePeriod.end')) </pre> <p><i>For this example, the corresponding query will have the following form:</i></p> <pre> query = conditions.project(Arel.sql('*')).where(encounters[:encounter].eq('Encounter Inpatient').and('encounters.end_point &gt;= MeasurePeriod.start AND encounters.end_point &lt;= MeasurePeriod.end')) </pre>
	<p>ORM to SQL</p>
	<p>⇒ SELECT * FROM encounters WHERE encounters.encounter = 'Encounter Inpatient' AND encounters.end_point &gt;= MeasurePeriod.start AND encounters.end_point &lt;= MeasurePeriod.end</p>

The type of the temporally related information can be: [DURING, OVERLAP, SBS, SAS, SAE, EBS, EAS,SDU,ECW,SCW, ECWS, SBCW, SBCWE, SACW,SACWE, SBDU,EBCW,EBCWS,EACW,EACWS, EADU, CONCURRENT]

■

# Glossary

HL7	Health Level Seven International is a not-for-profit, ANSI-accredited standards development organization
HQMF	Health Quality Measures Format is a HL7 standard for documents quality measurements
HSDP	HealthSuit Digital Platform is a cloud application which is developing in Philips Research for storing health data
QRDA	Quality Report Document Architecture is a HL7 standard for reporting the results of quality measurements
UI	User Interface is the designed space where interactions between human and machine occur
HISS	Health Informatics Solution Services Philips business group
H2H	Hospital to Home Philips business group
MAT	Measuring Authoring Tool
MITRE	MITRE is a non-for-profit organization that operates research and development centers sponsored by the federal government
RIM	Reference Information Model is a component of the HL7 V3 family of standards
HITSP C32	It is a summary document of medical consumers. The content may include administrative and clinical information
AREL	It is an algebra relational language. It simplifies the generation of complex SQL queries
ORM	Object Relational Mapper. It is a programming technique for converting data between incompatible type systems in object-oriented programming languages
UQMEE	Unified Quality Measure Execution Engine. It is software solution which transforms HQMF elements into executable queries conducting quality measures over a set of health data

# Bibliography

- [1] R. Suñol, "Avedis Donabedian," *International Society for Quality in Health Care 2000*, vol. 12, pp. 451-454, 2000.
- [2] "Open Clinical," Open Clinical, 1 July 2004. [Online]. Available: <http://www.openclinical.org/guidelines.html>.
- [3] Quality Forum, "Understanding Performance Measures: Anatomy and Types," Quality Forum, 2013.
- [4] "Unity Health Care," Unity Health Care, [Online]. Available: <http://www.unityhealthcare.org/>.
- [5] "Health Level Seven International," [Online]. Available: <http://www.hl7.org/>.
- [6] "MITRE," [Online]. Available: <http://www.mitre.org/>.
- [7] "webopedia," Webopedia, [Online]. Available: [http://www.webopedia.com/quick\\_ref/OSI\\_Layers.asp](http://www.webopedia.com/quick_ref/OSI_Layers.asp). [Accessed 22 April 2015].
- [8] "Extensible Markup Language (XML)," [Online]. Available: <http://www.w3.org/XML/>.
- [9] "JavaScript Object Notation," [Online]. Available: <http://json.org/>.
- [10] C. Q. I. W. Group, "HL7 V3 QM, DSTU R2.1," 2014.
- [11] "Quality Reporting Document Architecture - Category III, DSTU Release 1," 2012.
- [12] "Cypress: Meaningful Use Stage 2 Testing and Certification Tool," Cypress: Meaningful Use Stage 2 Testing and Certification Tool, 26 April 2013. [Online]. Available: [http://projectcypress.org/documents/cypress\\_software\\_design\\_20130422.pdf](http://projectcypress.org/documents/cypress_software_design_20130422.pdf).
- [13] "Ruby on Rails," [Online]. Available: <http://rubyonrails.org/>.
- [14] "Database," Wikipedia, the free encyclopedia, 2015.
- [15] "NoSQL," Wikipedia, the free encyclopedia, 2015.
- [16] "mongoDB," [Online]. Available: <https://www.mongodb.org/>.
- [17] "Apache Hadoop," [Online]. Available: <https://hadoop.apache.org/>.
- [18] "Map-Reduce, mongoDB," [Online]. Available: <http://docs.mongodb.org/manual/core/map-reduce/>.
- [19] "Tree traversal," wikipedia, [Online]. Available: [https://en.wikipedia.org/wiki/Tree\\_traversal](https://en.wikipedia.org/wiki/Tree_traversal).
- [20] "Bootstrap," [Online]. Available: <http://getbootstrap.com/>.
- [21] "Chart.js," [Online]. Available: <http://www.chartjs.org/>.
- [22] "SQLite," [Online]. Available: <https://www.sqlite.org/>.
- [23] "SQLite," [Online]. Available: <https://www.sqlite.org/version3.html>.
- [24] "Arel," [Online]. Available: <https://github.com/rails/arel>.
- [25] "Centers for Medicare and Medicaid Services," [Online]. Available: <https://www.cms.gov/>.
- [26] "eclipse," [Online]. Available: <https://projects.eclipse.org/projects/tools.xtend>.
- [27] "NLM," Value Set Authority Center U.S. National Library of Medicine, [Online]. Available: <https://vsac.nlm.nih.gov/>. [Accessed 21 August 2015].
- [28] "Measuring Authoring Tool," Measuring Authoring Tool, [Online]. Available: <https://www.emasuretool.cms.gov/>.
- [29] "django," [Online]. Available: <https://www.djangoproject.com/>.
- [30] PostgreSQL. [Online]. Available: <http://www.postgresql.org/>.

[31] "MySQL," [Online]. Available: <https://www.mysql.com/>.

## About the Author



**Pelagia Sykoudi Amanatidou** received her diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece in 2013. During her studies she focused on Software Technology and Computer Science. Her diploma thesis is titled "Navigation of a robot using potential fields and neural networks in a given place and design of methods for its efficient exploration." It was an opportunity to study many artificial intelligence methods and specialize in the field of genetic algorithms. In addition, during her studies, she was a member of PANDORA, the University's robotics team, where she took part on developing the robot's software development of interface for establishing reliable communication between the different types of modules for each part of the robot. The team won the 2nd place in the RoboCup World Competition 2013, in Eindhoven.

3TU.School for Technological Design,  
Stan Ackermans Institute offers two-year  
postgraduate technological designer  
programmes. This institute is a joint initiative  
of the three technological universities of the  
Netherlands: Delft University of Technology,  
Eindhoven University of Technology and  
University of Twente. For more information  
please visit: [www.3tu.nl/sai](http://www.3tu.nl/sai).