The Open Health Information Mediator: an Architecture for Enabling Interoperability in Low to Middle Income Countries

by

Ryan Crichton

Submitted in fulfillment of the academic requirements for the degree of Master of Science in the School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban.

March 2015

As the candidate's supervisors we have approved this dissertation for submission.

Signed:________________	Name:________________	Date:________________
Signed:________________	Name:________________	Date:________________

Abstract

Interoperability and system integration are central problems that limit the effective use of health information systems to improve efficiency and effectiveness of health service delivery. There is currently no proven technology that provides a general solution in low and middle income countries where the challenges are especially acute. Engineering health information systems in low resource environments have several challenges that include poor infrastructure, skills shortages, fragmented and piecemeal applications deployed and managed by multiple organisations as well as low levels of resourcing. An important element of modern solutions to these problems is a health information exchange that enable disparate systems to share health information.

It is a challenging task to develop systems as complex as health information exchanges that will have wide applicability in low and middle income countries. This work takes a case study approach and uses the development of a health information exchange in Rwanda as the case study. This research reports on the design, implementation and analysis of an architecture, the Health Information Mediator, that is a central component of a health information exchange. While such architectures have been used successfully in high income countries their efficacy has not been demonstrated in low and middle income countries. The Rwandan case study was used to understand and identify the challenges and requirements for health information exchange in low and middle income countries. These requirements were used to derive a set of key concerns for the architecture that were then used to drive its design. Novel features of the architecture include: the ability to mediate messages at both the service provider and service consumer interfaces; support for multiple internal representations of messages to facilitate the adoption of new and evolving standards; and the provision of a general method for mediating health information exchange transactions agnostic of the type of transactions.

The architecture is shown to satisfy the key concerns and was validated by implementing and deploying a reference application, the OpenHIM, within the Rwandan health information exchange. The architecture is also analysed using the Architecture Trade-off Analysis Method. It has also been successfully implemented in other low and middle income countries with relatively minor configuration changes which demonstrates the architectures generalizability.

Preface

The research work described in this dissertation was carried out in the School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban, from July 2011 to March 2015, under the supervision of Mr. Anban Pillay, Dr. Deshendran Moodley and Prof. Christopher Seebregts.

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any tertiary institution. Where use has been made of the work of others it is duly acknowledged in the text.

Declaration 1 - Plagiarism

I, Ryan Crichton, declare that:

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other person's data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other person's writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
   (a) Their words have been re-written but the general information attributed to them has been referenced.
   (b) Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References section.

Signed: ____________________

Declaration 2 - Publications

1. Crichton, R., Moodley, D., Pillay, A., Gakuba, R., & Seebregts, C. J. (2013). An Architecture and Reference Implementation of an Open Health Information Mediator: Enabling Interoperability in the Rwandan Health Information Exchange. In Foundations of Health Information Engineering and Systems (pp. 87–104). Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-39088-3_6

Signed: ____________________

Dedication.

I would like to dedicate this dissertation to my father, Graham Thomas Crichton. He will always be with us in memory. He helped me strive to achieve all that I could academically. I would also like to thank my family and especially my fiancée, Sarah Murray, for giving me the strength and support I needed to make it through this challenging process, as well as for her extensive editorial work.

Acknowledgements

I would like to thank my supervisors Anban Pillay, Deshendran Moodley and Christopher Seebregts for their never ending support and dedication to this process. Without their support I doubt I would have been able to produce this work.

I also wish to acknowledge the support of the Rwanda Ministry of Health (in particular, Gilbert Uwayezo and Daniel Murenzi) who with the National eHealth Coordinator, Dr Richard Gakuba, manage the national rollout of health IT as well as advisers, Elizabeth Peloso and Randy Wilson. Significant inputs were received from the Rwanda Health Enterprise Architecture (RHEA) and Rwanda Health Information Exchange (RHIE) project teams, including Wayne Naidoo, Carl Fourie, Hannes Venter, Mead Walker, Beatriz de Faria Leao, Paul Biondich, Shaun Grannis, Eduardo Jezierski, Dykki Settle, Odysseas Pentakalos and Bob Joliffe. Additional support was obtained from Mohawk College in Canada (in particular, Derek Ritz, Ted Scott, Justin Fyfe and Duane Bender) and eZ-Vida in Brazil (in particular, Dr Lincoln Moura, Dr Beatriz de Faria Leao and Ricardo Quintano Neira).

I would also like to acknowledge the funders for their generous support. The RHEA project is funded by grants from the IDRC (Open Architectures, Standards and Information Systems (OASIS II) - Developing Capacity, Sharing Knowledge and Good Principles Across eHealth in Africa. Grant Number: 105708), the Rockefeller Foundation (Open eHealth Enterprise Architecture Framework and Strategy Development for the Global South; Grant Number: 2009 THS 328) and the Health Informatics Public Private Partnership Project funded by the President's Emergency Plan for AIDS Relief (PEPFAR). The practical component of this research has been supported by funding from the President's Emergency Plan for AIDS Relief (PEPFAR) through a CDC cooperative agreement with Cardno Emerging Markets, Cooperative Agreement #PS002068. The research component was supported by the HEAL project, which is funded by grants from the Rockefeller Foundation (Establishing a Health Enterprise Architecture Lab, a research laboratory focused on the application of enterprise architecture and health informatics to low-resource settings, Grant Number: 2010 THS 347) and the IDRC (Health Enterprise Architecture Laboratory (HEAL), Grant

Number: 106452-001). The REACH (Research in Enterprise Architecture for Coordinating Healthcare) project was also funded by the IDRC through ecGroup (Derek Ritz).

Contents

Abstract **ii**

Chapter 1 Introduction **1**

1.1 Architectures for health information exchange 2

1.2 Problem statement, aim and objectives 3

1.3 Overview of the Research Design 5

1.4 Contributions 5

1.5 Thesis layout 6

Chapter 2 Literature review **8**

2.1 Information systems interoperability 8

2.2 Interoperability between health information systems 14

2.2.1 Technical interoperability 15

2.2.2 Syntactic interoperability 15

2.2.3 Semantic interoperability 17

2.2.4 Achieving all levels of interoperability with HIS standards 19

2.3 Architectural paradigms for distributed systems 23

2.3.1 Service-oriented architecture 25

2.3.2 Enterprise Service Bus 27

2.4 Interoperability in health care systems 29

2.4.1 Canada Health Infoway: Mohawk reference implementation 29

2.4.2 Health Service Bus by Ryan *et al.* 32

2.4.3 Xu *et al.* Mediator Services 33
2.4.4 NEHTA 35
2.5 Outstanding issues with ESB based Distributed Systems for HIS 36
Chapter 3 Research Design.. 39
3.1 The Rwandan Case Study 40
3.2 Deriving architectural concerns from the RHIE requirements 41
3.3 Architecture design and implementation 43
3.4 Analysis of suitability to LMIC environments 43
3.5 Analysis of architectural quality 44
3.6 Limitations and summary 46
Chapter 4 Design of the HIM Architecture.. 48
4.1 A national health information system for Rwanda 48
4.2 The architectural concerns for HIS interoperability 50
4.2.1 Concern #1: Facilitate interoperability between disparate and heterogeneous systems, both existing and future 51
4.2.2 Concern #2: Balance central governance with local autonomy 52
4.2.3 Concern #3: Adapt and scale within a changing environment 53
4.2.4 Concern #4: Prevent local changes from propagating through the system 53
4.2.5 Concern #5: Provide a low barrier to entry to connect new and legacy systems 54
4.2.6 Concern #6: Be secure and auditable 55
4.2.7 Concern #7: Be reusable across a multitude of environments 55
4.3 The Health Information Mediator 55
4.3.1 Overall HIM architectural paradigm: The ESB 56

4.3.2 Architectural viewpoints . . . 56
4.3.3 Architecture of the HIM . . . 58

Chapter 5 Implementation of the HIM Architecture in Rwanda . . . 69

5.1 The point of care interface . . . 69

5.2 Messaging format . . . 70

5.3 Mule ESB . . . 71

5.4 System Architecture of the OpenHIM . . . 72
5.4.1 Implementation of the point of care interface and access control components . . . 73
5.4.2 Implementation of the persistence component . . . 73
5.4.3 Implementation of the mediation component . . . 74

5.5 Implementation of the Rwandan HIE . . . 76
5.5.1 The RHIE workflow . . . 77

Chapter 6 Analysis and discussion . . . 82

6.1 Analysis of the HIM architecture . . . 82
6.1.1 Concern #1: Facilitate interoperability . . . 82
6.1.2 Concern #2: Balance central governance with local autonomy . . . 87
6.1.3 Concern #3: Adapt and scale . . . 87
6.1.4 Concern #4: Prevent propagation of local changes . . . 94
6.1.5 Concern #5: Provide a low barrier to entry . . . 95
6.1.6 Concern #6: Secure and auditable . . . 97

6.2 Re-usability of the HIM architecture in other LMICs . . . 97
6.2.1 MomConnect OpenHIM implementation . . . 97
6.2.2 Other uses of the HIM architecture . . . 100

6.3 Architecture quality analysis . . . 101
6.3.1 Identification of business drivers . . . 102

6.3.2 Utility Tree 103

6.3.3 Architectural analysis 104

6.3.4 Analysis of quality attributes 111

6.4 Comparison with existing approaches 115

Chapter 7 Conclusion 118

Bibliography 122

List of Figures

2.1 Dimensions of Interoperability 11

2.2 An HL7 v2 ORU_R01 message in ER7 format 16

2.3 Distributed System Architectures 24

2.4 The basic Service Oriented Architecture [54] 26

2.5 Canada Health Infoway: EHRS Conceptual View [9] 30

2.6 The Health Service Bus by Ryan *et al.* [59] 32

2.7 Architecture of the mediator service component [70] 34

2.8 NEHTA National eHealth Architecture. [50] 35

4.1 The architecture of the Rwandan Health Information Exchange 50

4.2 Context of architecture description [39] 57

4.3 Overview of components in the HIM architecture 59

4.4 The workflow of a transaction channel within the transaction mediation component 62

4.5 Scalability configurations of the HIM architecture 65

4.6 Extensibility of the HIM architecture 67

5.1 Systems architecture of the OpenHIM 72

5.2 OpenHIM persistence component data model 74

5.3 The structure of a sample mediation component 74

5.4 Query patients 78

5.5 Register a patient 78

5.6 Save an encounter 80

5.7 Query an encounter 81

6.1 Results of vertical scalability 91

6.2 Sample distributed HIM layout 92

6.3 Comparison of distributed vs. non-distributed configurations of the OpenHIM. 93
6.4 A simplified diagram of the MomConnect infrastructure (source: `http://www.jembi.org/project/national-pregnancy-registry/`) 99
6.5 HIM architecture utility tree 104

List of Tables

2.1 Interoperability Standards 21

6.2 Performance results of an m1.large instance using 5 concurrent threads . . . 90

6.3 Performance results of an m1.xlarge instance using 10 concurrent threads . 90

6.4 Performance results of a high specification machine using 20 concurrent threads 91

6.5 Performance results using two distributed m1.large instances 93

6.1 Estimated transactional load for the Rwandan HIE 117

Chapter 1

Introduction

Health Information Systems (HISs) are one of the six building blocks of a health system [66] and play a vital role in a country's health system. They store patient demographic and clinical information and provide health workers with views of this information. Often, a single health information system's scope extends only to the workflows associated with a single health facility, be it a clinic, hospital or other facility type. As these systems grow to accommodate new, more complex workflows, the data that they store becomes more valuable to both patients and health workers. Thus, sharing this information between health facilities becomes increasingly beneficial. Sharing of information in this context means that systems either provide information that other systems can consume; can themselves consume information shared by other systems; or both. Sharing of information gives health providers at separate facilities a view of the past medical history for a patient, as well as minimising the collection of duplicate information as the patient moves from site to site. This allows health providers access to additional information at the right time so that they may make decisions. This enables "continuity of care" for a patient no matter which health facility they attend. In addition, the time health providers spend collecting information can be reduced.

Health Information Systems in Low and Middle Income Countries (LMICs) have additional challenges not present in other environments [46, 1, 4]. There is often poor computing and network infrastructure due to limited funds or lack of technical expertise. There is also limited technical expertise to deal with the problems associated with sharing of clinical information. From a technological perspective one or more of the following are often true in these low resource settings: electrical power may be inconsistent and unpredictable; access to the Internet may be poor both in latency and speed, while also being inconsistent and unpredictable; users of systems are not sufficiently computer literate; and hardware is often older, poorly maintained and fails more frequently [42]. In addition, there are limited numbers of health workers to perform data capture and information management. This places a high burden on existing clinical staff to perform these administrative functions.

In addition, the current landscape of health information systems, in LMICs, is mostly characterised by fragmented, piecemeal applications deployed by multiple organisations [1, 8]. Applications are usually custom built to satisfy very specific needs, using heterogeneous architectures and technologies, with sharing of information low on the list of priorities. Recently the Ugandan Ministry of Health has placed a moratorium on mobile health systems because there were too many different projects run by too many different organisations for them to effectively co-ordinate[1]. While HISs may be useful in specific domains of health care they often contain information that is of high value if shared between systems external to that domain. It is beneficial to share certain information, such as patient demographics, between HISs to reduce replication among different HISs and thus reduce errors and improve efficiency of data capture.

One way to address the problem of sharing of information is through the development of a *Health Information Exchange*. A *Health Information Exchange* (HIE) is a system of software components that enable the sharing of clinical and administrative health care data among health care institutions, providers, and data repositories [17]. This is usually done for the mutual benefit of the systems involved in the exchange and to increase the effectiveness with which a patient can be treated. Types of information that could be shared include a patients medical history, patient administrative or demographic information, and information about health workers or health facilities. Allowing this information to be shared provides a fuller view of a patient's medical history and allows the clinician access to previously unavailable information.

1.1 Architectures for health information exchange

A number of architectures and paradigms have emerged to handle the complexities of enabling systems to interoperate. Service-oriented architecture (SOA) is a common approach that has been successfully applied in High Income Countries (HICs) to solve interoperability problems within an HIE [59, 49, 72]. SOA is also currently used as an approach to reduce

[1]See `http://www.ictworks.org/2012/02/22/ugandan-mhealth-moratorium-good-thing/`

complexity of large systems. HIEs are often large and complex due to the variety of systems that they connect.

Service-oriented computing is a paradigm where services form the foundational element of applications [54]. Services are self-contained, modular units that can execute a particular business function and are invoked through a published interface that is usually based on open standards [54, 33]. Service-Oriented Architecture (SOA) organises a set of services such that they can provide a cohesive set of business functions, while retaining loose coupling between the individual services [30, 54, 33]. Service-oriented architecture (SOA) has become the dominant architecture for facilitating interoperability between distributed software systems.

The Enterprise Service Bus (ESB) [10, 60] is an architectural model that eases the communication challenges between different services within a SOA. It is used to enable a SOA to be more easily developed by simplifying the connection of heterogeneous systems through a middleware component that enables communication between disparate services. This middleware component's common functions include message mediation, service orchestration and security [10]. These functions simplify the way in which service consumers interact with service providers and centralise much of the complexity associated with connecting to services.

The ESB approach has been previously applied to the problem of facilitating interoperability between heterogeneous information systems in the health domain in HICs [9, 59], but has not yet been proven in LMICs where conditions are significantly different.

1.2 Problem statement, aim and objectives

There are currently no accepted generic frameworks or methodologies for addressing interoperability challenges between HISs in LMICs. ESB-based HIE approaches have been used to address similar challenges in HICs. However, it is not presently known whether this approach will be effective in LMICs. This research examines whether an Enterprise Service Bus architecture will be an effective approach to addressing interoperability challenges between disparate HISs in LMICs and will also reduce the cost and effort of engineering interoperable HISs within LMICs.

This research explores the construction of a framework that would enable HIEs to be more easily deployed in sub-Saharan African LMICs. The first step towards this goal was to design, develop, analyse and validate a software architecture based on the Enterprise Service Bus architectural model that simplifies interoperability between HISs. This simplification should be in terms of complexity as well as the level of effort required to enable interoperability, specifically in LMIC contexts.

To achieve this aim, a case study approach was taken using the development of the Rwandan Health Information Exchange. The case study is used to better understand and identify the key challenges and concerns for health information exchange in LMICs and to serve as an environment for analysing and validating the architecture. The generalizability of the architecture is explored by studying its implementation in other LMIC environments.

The concrete objectives of this research are as follows:

1. Analyse existing approaches to addressing interoperability challenges and explore existing approaches used in HIEs for HICs.

2. Using the Rwandan HIE as a representative sub-Saharan African LMIC case study:

   (a) Analyse this specific environment and its requirements to determine a generic set of architectural concerns for an ESB-based architecture that might simplify the development and implementation of an HIE in LMICs.

   (b) Design a software architecture based on the established ESB architectural model that meets the architectural concerns identified from an analysis of the case study.

   (c) Validate the architecture by implementing and deploying a reference implementation in the representative country.

3. Analyse the effectiveness and suitability of the architecture against architectural concerns, design objectives and quality attributes.

4. Analyse the suitability and re-usability of the architecture in other LMIC environments.

1.3 Overview of the Research Design

It is a challenging task to develop systems as complex as HIEs that will have wide applicability in all LMICs. It was decided, instead, to initially focus on the development of the HIE in Rwanda as a representative case study. The ministry of health of Rwanda is embarking on the creation of an integrated National Health Information System (NHIS). Part of this plan involves the creation of a pilot HIE for a single district in Rwanda. This HIE enables medical records to be transferred between health facilities so that a patient's full record may be viewed at any health facility that they attend. The Rwandan HIE (RHIE) is used to explore the interoperability challenges experienced in LMICs and to provide an environment where a reference implementation could be deployed and analysed.

A number of key, general architectural concerns were extracted from RHIE requirements to drive the design of the architecture. Although the architecture was designed for the Rwandan use case, attempts were made to ensure that the architecture had certain desirable characteristics that would make it useful for solving similar problems in similar environments. To validate the architecture a reference implementation of the HIM architecture (the OpenHIM) was developed and deployed in Rwanda.

The architecture was evaluated along three dimensions. Firstly, an analysis of how well the architecture addresses the architectural concerns and solves the interoperability problems for the Rwandan HIE is given. The Rwandan implementation of the architecture serves to validate the architecture and demonstrates its applicability to a real world LMIC setting. Secondly, an analysis of other LMIC projects that have begun to adopt the architecture is given. Finally, a formal architecture analysis was performed. The architecture trade-off analysis method (ATAM) was used to determine the quality of the architecture by looking at the architecture's modifiability; scalability and performance; and it's security.

1.4 Contributions

The key contribution of this research is an ESB-based architecture (the Health Information Mediator) that is an efficacious and generalizable solution to interoperability problems within health information exchanges in sub-Saharan LMICs. This architecture's wide applicability in LMICs is demonstrated by deploying the reference implementation in two diverse low to middle income environments. The architecture is a step toward articulating a framework for constructing HIEs within LMICs.

The reference implementation of the architecture, a novel mediation component called the OpenHIM is also presented. The HIM architecture exhibits some novel features that contribute additional design principles to the area of health information mediation when compared to existing middleware architectures for constructing HIEs:

1. The HIM architecture is agnostic of health information standards used for messaging such that legacy or new cutting-edge standards may be used.

2. The HIM architecture enables health information messages to be mediated at both the inbound and outbound interface. This allows messages to be transformed such that any service provider or service consumer can be more easily connected even if they cannot produce standards conformant messages. This enables legacy HISs to be more easily integrated into an HIE.

3. The HIM architecture is agnostic of the type of transaction that needs to be enabled within an HIE. It provides generic mechanisms to enable the implementation of any type of transaction.

1.5 Thesis layout

In Chapter 2 the challenges involved in facilitating interoperability between systems are introduced and previous approaches to these problems are discussed. Chapter 3 explains and describes the methods used to achieve the research objectives. In Chapter 4 architectural concerns are extracted from the Rwandan HIE use case and those architectural concerns are used to drive the design of the architecture of a software component to facilitate interoperability: the HIM architecture. The architecture is validated with a reference implementation within the Rwandan HIE. The reference application, the OpenHIM, is described in Chapter

5. A comprehensive analysis of the HIM architecture is presented in Chapter 6. Finally, conclusions are drawn and we point to future work in Chapter 7.

Chapter 2

Literature review

In this chapter the challenges associated with interoperability between health information systems are identified. These problems are described and the key dimensions of interoperability are identified and explained. The extent to which existing health information standards solve the challenges of interoperability is also explored. In addition, a number of key architectures that attempt to solve the problems associated with enabling health systems interoperability are examined. Finally, outstanding issues are identified and discussed to frame the context for the remainder of the research.

2.1 Information systems interoperability

Heterogeneity of the component systems is the crucial characteristic that makes sharing of resources between systems (i.e. interoperability) difficult. Heterogeneity can be categorised as either information heterogeneity or system heterogeneity [52].

Information heterogeneity refers to the difference in both the structure and the content of the information used within a system. There are three dimensions to information heterogeneity [52]:

- **Syntactic heterogeneity** - describes the differences in how information is formatted and encoded in different systems.
- **Structural heterogeneity** - describes the differences in how information is structured in different systems. This includes the data structures that store the information and how that information is related.
- **Semantic heterogeneity** - describes the differences in the meaning of information in different systems. This includes understanding the concepts being communicated by the information and the context of the information.

System heterogeneity refers to the physical differences in component systems and includes [52]:

- **Information system heterogeneity** - refers to the differences in the design of systems. This includes differences in database management systems, data models and system capabilities.
- **Platform heterogeneity** - describes the differences between the underlying platforms on which the system runs. This includes differences in operating system, file systems, technology stacks and hardware.

Information and system heterogeneity are the main reasons why sharing information is a difficult task. In the following sections we explore how these differences have been addressed previously.

Interoperability attempts to mitigate the effects of heterogeneity by defining how systems can communicate in the presence of heterogeneity. Interoperability is a difficult problem to solve in software engineering due to the many facets that require consideration in order to make interoperability between different systems possible [62].

The IEEE glossary of 1990 [34] defines interoperability as:

> "the ability of two or more systems or components to exchange information and to use the information that has been exchanged."

This definition implies that for systems to interoperate they must not only be able to physically receive data from another system but also be able to make use of the received data in its internal business processes.

There are a number of different dimensions of interoperability that describe the levels at which systems can process and reason with data it receives. Sheth (1999) defines four classical dimensions of interoperability: semantic, structural, syntactic and system interoperability [62]. These dimensions address the different aspects of heterogeneity discussed in Section 2.1. Ouksel and Sheth later propose a newer framework for interoperability that includes two additional dimensions: pragmatics and social world [53]. In this framework they suggest that interoperability should include four key dimensions: syntactic, semantic, pragmatic and social world interoperability. Jinsoo Park [55] describes the classical dimensions of interoperability which include definitions of both syntactic and semantic interoperability that closely match those defined by Ouksel and Sheth.

A number of studies have been performed particularly for the health domain. The health level 7 (HL7) interoperability working group performed a comprehensive review of definitions of interoperability [28]. They identified many definitions from both the health domain as well as other, more established, information system domains such as the financial domain. They identify three major dimensions that are described by a number of different organisations, these are: technical interoperability, semantic interoperability and process interoperability. They also note that the concepts of social and process interoperability are currently emerging in interoperability definitions, especially in more established organisations.

These dimensions of interoperability are inter-related. They form a stack with each dimension depending on the one below it [28]. For example, semantic interoperability is not possible without having both syntactic and technical interoperability and you cannot have pragmatic interoperability without semantic interoperability and so on. The major dimensions of interoperability that commonly appear in literature are listed and described below. We use an example scenario to aid explanation. Figure 2.1 on page 11 shows the relationships between these dimensions of interoperability in the form of a pyramid diagram with each higher level dimension depending on lower level dimensions.

Example scenario

In this section an example scenario is described that illustrates the interoperability dimensions described above in practical terms.

Consider a scenario where a patient arrives at a rural clinic. This clinic has an electronic medical record (EMR) system that interoperates with a national server that stores patient medical records. The health workers at the rural clinic want to download the patient's medical history from the national server to their local EMR system. This will give them a more detailed history for that patient and save them from recapturing this information.

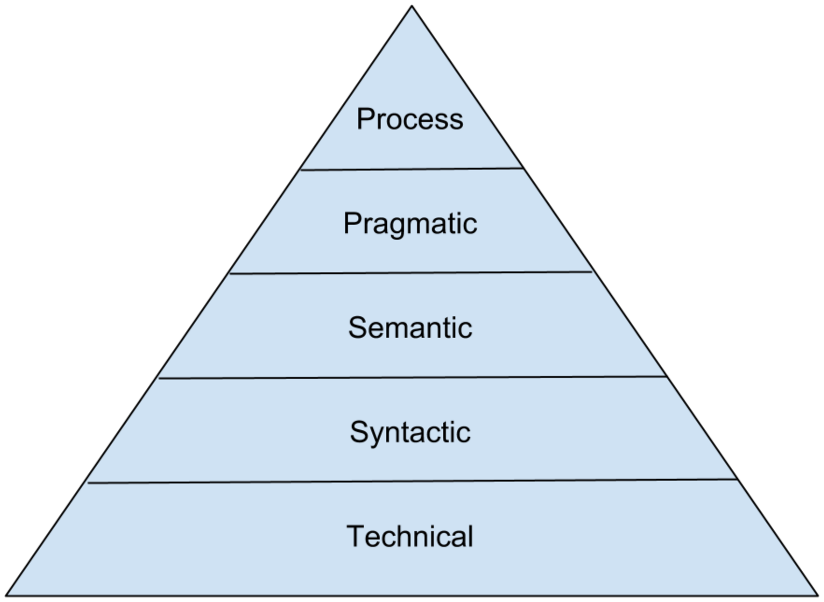


Figure 2.1: Dimensions of Interoperability

Below, examples are given to illustrate the meaning of each dimension of interoperability using the example scenario that we have defined:

- **System/technical interoperability** - The focus of this dimension is the transfer of data from one system to another. It is concerned with how the actual bytes get transferred between systems on a network. In our example, this dimension concerns the network packets and IT infrastructure that enable the electronic transfer of the data from the central server to the clinic's health information system. Currently a common mechanism is to make use of the TCP/IP [69] network stack and connect the communicating systems using specialised network equipment that supports this network stack. Higher level protocols such as HTTP [23] that build on TCP/IP are also prevalent and enable systems to transmit data in a standardised way.

- **Structural/syntactic interoperability** - Two systems are syntactically interoperable if they can effectively exchange data and internally process the exchanged data. In order for the exchanged data to be processed by a system, it must conform to a pre-defined structure or syntax that is known to both systems. Extensible Markup Language (XML) [67] is an example of a standard that assists in enabling syntactic interoperability. In our example, the EMR system at the rural clinic needs to know the structure of the previous history data so that it can internally process this information. If the EMR system was syntactically interoperable with the central sever then it is able to process, store and display the data to the user. However, this does not mean that the system understands the meaning of clinical concepts contained in the content of the message. In our example, if the message was of a particular XML structure then the application has syntactic interoperability as long as it knows the schema or structure that the XML message conforms to i.e. the structure of the XML tags and what each tag represents. However, it does not necessarily understand the meaning of the content between the tags. This XML message could be in the form of an XHTML[1] document. The system does not understand the content of the XHTML document, however, it can process the message and display the XHTML content to the user so that they can interpret the meaning of the message. This is much like how Internet browsers can display and process web pages for a user but the browser does not understand or interpret the meaning of the content of the web page. This is left to the user. Syntactic interoperability is a prerequisite for semantic interoperability.

- **Semantic interoperability** - Semantic interoperability concerns the interpretation of the meaning of data to enable the data to be used within the systems internal structures. It implies that both the sending and receiving system have an identical understanding of the information. Semantic interoperability can be accomplished by the participating systems sharing a controlled vocabulary with predefined meanings. In our example, this would allow the EMR system to understand the meaning of concepts in the patient's previous history. In our example, if the patient has an allergy to sulphur then the EMR system could interpret this and include it in its internal data model for additional processing and use within the application. This information could, for example, be used to provide an alert to a health worker if a sulphur containing drug was prescribed. To allow the receiving system to understand the meaning of this message, the message would have to include a reference to a controlled vocabulary to indicate the concept of an "allergy" and have a mechanism

[1]See the XHTML specification here: `http://www.w3.org/TR/2010/REC-xhtml11-20101123/`

to relate this to another reference to a controlled vocabulary to indicate that the allergy was to “sulphur”.

- **Pragmatic interoperability** - In order to process information that a system receives from another system, the receiving system also needs to understand the *intention* of the message it receives. Pragmatics has to do with understanding the intent of a message. In our example, this means that the EMR system understands that the information that it receives is a patient’s previous history and that it should be saved to that patient’s local record. In our example, if the previous patient history was transmitted using a HL7 version 2 [31] message format then the pragmatics of the message can be inferred from the *message type* and *trigger event* data elements that are contained within an HL7 message structure. Standard message formats often include a mechanism to identify the pragmatics of a message using a controlled vocabulary to describe the message intention, thus, pragmatic interoperability is related to semantic interoperability but sits at a higher level.

- **Process interoperability** - This form of interoperability has to do with integrating information from various systems into actual workflows and business processes performed by people. In our example, this would describe the workflow in which the patient’s previous encounters are fetched from the national-level server. It describes the expected interaction between the EMR system and the server as well as how the interaction between the systems integrates with the actual business process of the clinic staff treating the patient. Process interoperability describes how and when the data is received from the central server in the overall business process of the clinic and how that information is used by the health workers, within their workflow, to treat the patient.

In this research we focus on a subset of these dimensions of interoperability. Firstly, we consider semantic and pragmatic interoperability to be dealing with the same underlying concern of understanding the meaning of a particular message; the former has to do with the meaning of the content of the message and the latter has to do with the intention. These are two separate but related concepts. Due to this these two concepts have been collapsed into a single dimension that we will simply refer to as semantic interoperability. Secondly, process interoperability has to do with how information is used in actual workflows. In this research we restrict ourselves to consider only the flow of information rather than the context in which it is used. Thus, we do not consider process interoperability any further. Thirdly, Gibbons *et al.* [28], do not acknowledge syntactic interoperability as a dimension of interoperability between technical interoperability and semantic interoperability. However, we believe that this dimension is distinct and this view is supported in other research [52, 55].

The most relevant dimensions of interoperability that concern information sharing between heterogeneous systems that we chose to guide this research are thus: Technical interoperability, Syntactic interoperability and Semantic interoperability.

In the following sections we explore the varying degrees to which each dimension has been achieved in HISs.

2.2 Interoperability between health information systems

Health Information Systems (HISs) are information systems used in the health care domain. There are many different types of HISs that support different aspects of health care. These include Electronic Medical Records (EMR) systems, laboratory systems and pharmacy systems. These systems help health care workers manage the range of information needed to support a patient's health care.

Interoperability standards are a fundamental mechanism to address the problems faced with heterogeneous systems interoperability [2]. In this section standards that enable interoperability between health information systems at the different levels of interoperability are described along with studies that show how the use of interoperability standards produce interoperable health care systems. First, standards that address each level of interoperability are discussed.

2.2.1 Technical interoperability

Due to the global adoption of standards such as Transmission Control Protocol/Internet Protocol (TCP/IP) [69] and User Datagram Protocol (UDP) [57] the problem of technical interoperability, as we define it, has largely been solved. Higher level protocols such as Hypertext Transfer Protocol (HTTP) [23] have become popular due to the success of the Internet and this makes it even easier for disparate computer systems to share data. The advent of web-service technologies such as Simple Object Access Protocol (SOAP) [6] and Representational State Transfer (REST) [24] have made reliable data sharing between systems easier and more common. All major programming languages include HTTP support. It is either built in or supported via third party libraries. SOAP and REST support is also common. Even mobile devices support a variety of protocols, such as 3G, HSDPA and LTE, that enable Internet access from remote locations. Thus, technical interoperability is no longer a major concern. However, technical interoperability only allows the transmission of data from one computer to another across a network. It does not imply that the systems can process or understand received data [53, 28].

2.2.2 Syntactic interoperability

A message format describes the syntax and structure of a message and these are commonly used to enable syntactic interoperability. Message formats describe how data can be structured such that it can be transmitted and interpreted by another system. Current common messaging formats include XML (Extensible Markup Language), JSON (Javascript Object Notation) or other custom plain text mark-ups. Most modern interoperable systems rely on these formats. These message formats can be further constrained to represent data from a specific domain. In the health care domain there are a number of existing message formats. These include HL7 [31] (version 2 & 3) and OpenEHR Archetypes [11] among others [26, 20]. Each of these are message formats are used to describe different types of health data. HL7 and OpenEHR archetypes are designed to carry patient level clinical data, whereas others are designed to carry aggregate health indicators or other types of health information. HL7 version 2 defines a large number of message types with each having specific use cases and structures. OpenEHR archetypes define a method for describing the structure of clinical data as well as the metadata that describes the structure of this data.

A sample HL7 v2.5 of message of type ORU_R01 (observation result) is shown in Figure 2.2 on page 16.

```
MSH|^~\&|RapidSMS|F316|SHR|RwandaMOH|20111108065718||ORU^R01^ORU_R01|68080|D^C|2.5^RWA|||||||||BIR
PID|||1198270120343041^^^^NID
PV1|1|Community Health|316||||1197370056233083|||||||||||||||||||||||||||||||||||||20111108065718
OBR|1|||^Maternal Health Reporting
OBX|1|CE|^Risk Code||NP||||||F
OBX|2|TS|^Birth Date||20111108||||||F
OBX|3|NM|^Baby Weight||3.3|k|||||F
OBX|4|CE|^Birth Code||GI||||||F
OBX|5|NM|^Child Number||01||||||F
```

Figure 2.2: An HL7 v2 ORU_R01 message in ER7 format

This message contains a number of observations for a specific patient. These can be seen in the lines starting with OBX (OBX segments). A receiving system will know the structure and allowed data types for this specific message as this is defined in the HL7 version 2 specifications. Thus HL7 version 2 enables syntactic interoperability. However, as we can see some of the OBX segments contain specific codes such as “Risk Code” which is equal to “NP” in this case. This is part of the content of the message and the HL7 version 2 specification does not specify what this should contain. With syntactic interoperability the receiving system does not understand what “Risk Code = NP” means. To be able to understand this observation the systems need a shared understanding of the semantics of the message.

It is often difficult to enable syntactic interoperability with only the standards mentioned above. They contain generic constructs and data types used to construct messages. However for specific implementations one would have to restrict these further to define exactly what data can be placed in these messages and where it should be placed. These message formats often contain significant options as to what data elements may be included in a message. This allows them to be generally applicable to a number of use cases. However, this results in systems often choosing to implement these standards differently leading to varying support for optional data elements. The result is differences in the content of messages sent by different systems. This hinders interoperability. Some systems, both legacy and new, also implement their own custom or propriety message formats. It is common for modern day systems to implement RESTful web service interfaces. These interfaces are often an incarnation of that system’s data model and thus will be different between systems.

The use of these non-standardised message formats can make syntactic interoperability a challenge as each system will have its own format that needs to be supported.

Overall, a clearly specified message format that can be ubiquitously understood by heterogeneous systems can solve the syntactic interoperability problem. However, this is not always achievable with the standard message formats that currently exist.

2.2.3 Semantic interoperability

Medical data is voluminous, heterogeneous and has no canonical form [13]. This makes it challenging to codify and present in a uniformly understood way and also more challenging to work with than in domains where data is more discrete and uniformly represented and understood.

Degoulet et al [16] enumerates the following challenges with enabling semantic interoperability between health information systems:

- HISs often evolve from legacy systems where semantic meaning is hidden within the application.
- A greater variety of heterogeneous HISs than ever before are expected to interoperate, even between various domains of health care.
- Medical vocabulary is constantly changing and evolving.
- There is a great variety in medical terms between different levels of health providers and providers from different domains.

In order to enable a shared understanding of the concepts in the medical domain several structured, systematic code systems have been created. Some of these code systems are defined as ontologies containing medical concepts and others are lists or hierarchies of medical concepts mapped to codes with clearly specified meaning. An ontology is an explicit specification of a conceptualisation [29]. It is a formally represented body of knowledge about a particular domain in the form of concepts and the relationships between them. These code systems are complex to create and maintain especially in the health domain where health concepts are voluminous and always changing. Concepts are often coded in a standardised coding systems in order for systems to achieve some level of common understanding of what certain terms mean. In the health domain this is often done by using standard medical classifications such as ICD-10 (International Classification of Diseases) or LOINC (Logical Observation Identifiers Names and Codes), or coded ontologies such as SNOMED-CT (Systematized Nomenclature of Medicine - Clinical Terms) [12]. ICD-10 codifies diseases and health problems and LOINC codifies laboratory observations.

Code systems contain a large number of clinical terms in a structure where concepts can be referenced uniquely by a specific code. For example, if a system receives a message in a form that supports semantic interoperability the system will be able to interpret the meaning of the message content by examining the codes used to describe the message content. If the system has a mapping between the defined codes in a code system and its internal data model then it will be able to process and store the message content in its native, internal form as each code has a defined meaning. The fact that this clinical encounter was received from a separate system would be transparent to the user.

In order to communicate data in an understandable fashion, it needs to be mapped to concepts in these standard code systems. This mapping process is often difficult due to the complexity of data within the health domain. Multiple code systems exist that attempt to solve this, such as those mentioned above. There are often no exact mappings for concepts represented in a system to concepts represented in the code system [55]. These code systems may overlap and refer to concepts at a different layer of granularity. Code systems are also made to be generally applicable, thus, requiring local customisation for a specific use case for it to be useful. This is a key problem that makes interoperability in the health domain difficult as customised standards are no longer interoperable. Furthermore, legacy systems often do not support these standards and it is essential to integrate these systems into the overall HIS. Legacy systems semantics are often hidden within the application and it can be a tedious task to extract these [16]. It is also difficult for a country to choose between the different standards available and to map these to the data being collected. It requires experts in standards and standards development in order to perform such tasks. Code systems would also need to be customised to ensure a country's requirements can be met.

This can be a challenge for LMICs.

2.2.4 Achieving all levels of interoperability with HIS standards

A number of standards available to assist with solving the different dimensions of interoperability were described in the section above, such as HTTP [23], HL7 version 2 [31] and LOINC [25]. However, to achieve higher levels of interoperability multiple standards are needed [16, 58, 62, 18]. In this section a discussion on how subsets of these standards can be used together to achieve multiple levels of interoperability is given.

We categorise data interoperability standards into two broad categories: single concern standards that deal with a single aspect of interoperability and multi-concern standards that deal with multiple aspects of interoperability (usually by using multiple existing standards). Table 2.1a on page 21 shows the major single concern standards for health care interoperability separated by the dimensions of interoperability with which they are concerned.

In the health domain multi-concern standards are often referred to as “profiles”. Profiles are detailed technical specifications that show what standards can be used together to gain full technical, syntactic and semantic interoperability for a particular use case [63]. They do this by describing how the standards are to be used together and exactly what data values are allowed, as well as describing unambiguously what each data value means. This has the advantage of clearly specifying how the standard can be used to share specific information. However, this forces the profile to be specific to a certain use case, such as resolving a patient’s identity or storing and querying clinical documents. These specific use cases may not fit the needs of all environments and profiles do not exist for all use cases. For example, in the case of resolving a patient’s identity, a query may require parameters such as the patient’s first and last name and their date of birth. In some LMICs, especially in rural areas, a person’s date of birth is not always known.

Some standards are more appropriate in LMICs than others [2]. There are various facets that affect a standard’s suitability. These include: conciseness, understandability, tool availability, tool maturity and ease of implementation. Conciseness is important as LMICs often do not have the same level of bandwidth and network infrastructure as high income countries. LMICs also do not have the level of technical skill and in-depth knowledge to effectively use more complex standards. Tool maturity and availability is a key consideration in low resource settings as time and resources are spared if tools already exist and are freely available. Finally, a standard should be easy to implement. It should be easy to integrate into existing systems and not require large amounts of technical skill to implement, maintain and use within a LMIC.

The HL7 version 2 standard [31] is a commonly used message format for the health domain. This standard has a large number of freely available tools for developers. It consists of a large number of message types for many different use cases each with their own structure and defined syntax. HL7 version 2 alone cannot ensure semantic interoperability but it can be used to achieve syntactic interoperability. Other code systems such as SNOMED-CT or LOINC [12] are required to codify the data contained in a HL7 version 2 message so that it can be semantically understood. For technical interoperability HL7 version 2 employs the Minimal Lower Layer Protocol (MLLP) protocol. MLLP defines a protocol for sending HL7 version 2 messages over TCP sockets. MLLP makes use of a persistent TCP socket that uses particular bytes to delimit messages within the stream.

HL7 version 3 is a complete departure from HL7 version 2. HL7 version 3 is an XML-based standard and emphasises greater uniformity in the way messages are constructed by employing a generic reference information model (the RIM) that all HL7 version 3 messages must conform to. This, however, led to very large message sizes and made HL7 version 3 difficult to understand and implement due to the generality of the data model. HL7 version 3 requires that one restrict the base standard to a usable subset. For LMICs this is difficult due to the limited number of informatics experts available to do this and the amount of time it requires.

Integrating the Healthcare Enterprise (IHE) is an initiative that attempts to improve information sharing between health information systems[2]. IHE produce profiles for health information systems interoperability. These profiles describe what standards can be used and specify how they are used and what data elements must be included in order to satisfy an interoperability use case. Standardised IHE profiles such as Cross-Enterprise Document

[2]See `http://www.ihe.net/About/`

Sharing (XDS) or Patient Identifier Cross-Referencing (PIX) describe and restrict the use of the single concern standards in order to achieve full system, syntactic and semantic interoperability [35]. Table 2.1b shows some of the more common combinations of standards that are used to achieve full semantic interoperability between health information systems. IHE profiles are pragmatic in that they attempt to make use of existing standards that already have wide adoption so that integration of IHE profiles into existing systems can be simplified. IHE profiles are also very use case specific. This has the benefit of allowing them to clearly specify exactly how semantic interoperability can be achieved, however, it also means that different IHE profiles are needed for each use case. IHE covers many of the priority use cases but it is impossible for IHE to cover all conceivable use cases.

In Table 2.1b on page 21 we show four examples of how full semantic interoperability can be achieved by using a number of different base standards. Below we describe these examples and explore some of the strengths and weaknesses of each approach.

Technical Interoperability	SOAP
	REST
	TCP Sockets
Syntactic Interoperability	HL7 v2
	HL7 v3
	OpenEHR
Semantic Interoperability	SNOMED-CT
	ICD-10
	LOINC

(a) Single concern standards

Full Interoperability	System	Structure	Semantic
MLLP, HL7v2, SNOMED-CT	MLLP	HL7 v2	SNOMED-CT
IHE Profile: PIX v2	MLLP (TCP socket)	HL7 v2	constrained value sets
IHE Profile: PIX v3	SOAP	HL7 v3	constrained value sets
SOAP, HL7v3, SNOMED-CT	SOAP	HL7 v3	SNOMED-CT
IHE Profile: XDS	SOAP	ebXML	constrained value sets

(b) Full interoperability

Table 2.1: Interoperability Standards

In the first row of Table 2.1b on page 21 the use of HL7 version 2 over MLLP is shown along with supporting standards that enable semantic interoperability. The use of MLLP enables technical interoperability. HL7 version 2 messages are placed between the delimiter bytes in the ER7 format described by the HL7 version 2 standard. Each HL7 version 2 message has a known structure that is described in the standard to enable syntactic interoperability. In order to gain semantic interoperability, coded ontologies such as SNOMED-CT or standard code systems such as LOINC or ICD-10 are used. These code systems allow the semantics of certain clinical content within HL7 version 2 messages to be specified.

In the second row an example of the IHE PIX profile is shown [35, 36]. This profile specifies that HL7 version 2 message over the MLLP protocol are to be used for technical and syntactic interoperability. It also specifies some pre-defined values and content that should go into the HL7 version 2 messages. These values are specified in such a way that they are unambiguous to the receiving system thus enabling semantic interoperability.

The third row shows another version of the IHE PIX profile that uses HL7 version 3. In this case SOAP web services are used for technical interoperability. SOAP makes use of the HTTP protocol over a TCP/IP connection to transmit XML messages. In this case the content of that XML message is a constrained HL7 version 3 message with specific contents that represents a PIX message. Here semantic interoperability is provided by the PIX profile which clearly and unambiguously describes the values that go into the PIX message.

In the fourth row is an example where a number of co-operating standards may be chosen to achieve semantic interoperability as shown by Ryan *et al.* [58]. In this case SOAP was chosen as the protocol for technical interoperability, HL7 version 3 is used to provide syntactic interoperability and SNOMED-CT is used within the content of the HL7 version 3 messages to describe the meaning of the clinical content of that message.

The fifth row shows another IHE profile; the XDS.b profile [35, 37]. Again the SOAP protocol is used to gain technical interoperability. In this case the profile specifies the use of ebXML to enable syntactic interoperability. The profile lists all the possible options for the ebXML data elements and describes what content should go into the message. Doing so provides semantic interoperability between different systems.

As can be seen there are multiple methods available to achieve full semantic interoperability between systems. There is no right or wrong solution as different standards have different strengths and weaknesses. For example, IHE profiles provide an easy way to enable systems to interoperate for specific use cases. However, if they do not support a specific use case then a custom solution could be built from the existing standards such as HL7v3 combined with SNOMED-CT. This may take more time and resources but it provides a more flexible solution. This is an unresolved problem within health systems interoperability. Many different standards may be employed to provide an interoperable solution, however, it is not clear to the implementers what approach is best. Health information systems interoperability is still too immature and *de facto* standards are yet to emerge.

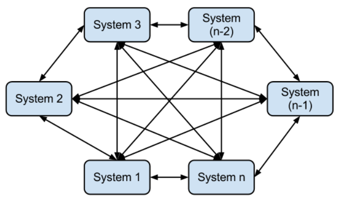
2.2.4.1 Health information standards in low resource settings

Recent research has begun to determine the suitability of various types of interoperability standards for LMICs. Abedesin *et al.* [2] surveyed the available standards and how they can be applied in LMICs. A new approach to creating a national health information system, called the middle-out approach, has also been proposed [14]. It has been found to be highly applicable to low resource settings [14, 47]. This approach describes how a standards framework can support both local autonomy and heterogeneity (of technology and process) of systems in the field while still maintaining the ability for these systems to cooperate as well as support national infrastructure and policy.

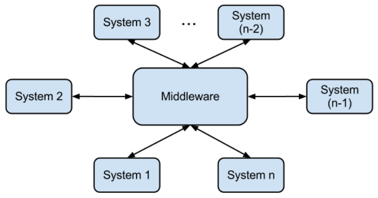
2.3 Architectural paradigms for distributed systems

This section explores the past and current architectural paradigms that describe how systems can be arranged as a collection of interoperating systems. In particular we focus on the enterprise service bus architectural model that helps instantiate the service-oriented architecture paradigm. Chappell describes the evolution of these architectural paradigms [10].

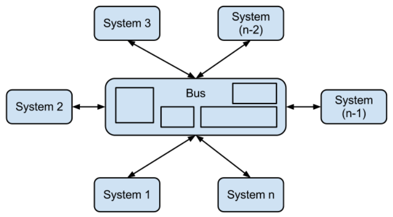
A software architecture is the gross organisation of a collection of interacting components of a software system that play an important role in: understanding, reuse, construction, evolution, analysis and management during the software development process [27].



(a) Point to point architecture



(b) Hub and spoke architecture



(c) Bus architecture

Figure 2.3: Distributed System Architectures

The first generation of architectures employed point to point communication (see Figure 2.3a on page 24). Heterogeneous systems that needed to interoperate were identified and a common way for the systems to communicate was developed. Each system would then be extended with adapters to allow it to communicate with the other system. This method worked well as it was easy to implement and well understood but it did not scale well. The adapters became increasingly difficult to maintain as new systems were added. To solve this, middleware systems were introduced. These are located between systems that want to exchange information. The middleware manages and facilitates all communication between these systems. This architecture is called the hub and spoke model. The middleware system often provides some more advanced functionality to facilitate interoperability, such as providing security, allowing messages to be transformed and validated, as well as exception handling. An example of such an architecture is given in Figure 2.3b on page 24. This solved the problem since only a single adapter is need for each new system. However, this introduces another problem; a single point of failure. If the middleware fails, all communication between systems are lost. However, this approach also maintains a single point of control.

This evolution of architectures shows the shift from the paradigm of system integration to the paradigm of system interoperability. Integration attempts to allow two specific systems to share information with each other as they exist at a specific point in time. Integration often makes use of system specific assumptions and results in rapid information sharing between systems. However, this sharing is brittle as it is closely coupled to the participating systems. Interoperability, on the other hand, enables systems to share information using a defined and open mechanism such that as systems are extended, added or reimplemented they are still able to interoperate [49]. Interoperability standards are often used to enable this. Interoperability allows systems to continue to share information even if the environment of these systems changes. System integration is often a first step to enabling systems interoperability. It is simpler and enables systems to start sharing information until more robust and advanced mechanisms can be put in place.

2.3.1 Service-oriented architecture

Service-oriented computing is a paradigm where services form the foundational elements for developing applications [54]. Services are self-contained, modular units that can execute a particular business function and are invoked through a published interface that is usually based on open standards [54, 33]. Service-Oriented Architecture (SOA) is a way of organising a set of services so that they can provide a cohesive set of business functions while retaining loose coupling between the individual services [30, 54, 33]. Loose coupling refers to the ability of services to be modular and independent so that they can be swapped out as needed. The service interface enables this by protecting the user of a particular service from the service's actual implementation.

A software system that makes use of a service is termed a service consumer and one that exposes a particular service is called a service provider. These concepts are abstract and thus a software system is able to act as both a service provider and a service consumer.

A basic SOA enables interoperability between software systems through the exchange of messages. SOA relies on the interaction between three major components: service providers, service consumers and a service registry. Service providers expose service implementations and publish service descriptions to the service registry. Service consumers may look up these service descriptions and use them to bind to a particular service implementation to enable them to make use of that service. This binding is dynamic and occurs at runtime [54]. See Figure 2.4 on page 26.

SOA does not specify a particular technology to enable the communication between service provider and service consumers. Any messaging technology could be applied to form a SOA implementation, however, the most common SOA implementation utilises web services to enable communication [54, 33, 30, 64]. Web services utilise web technologies that include

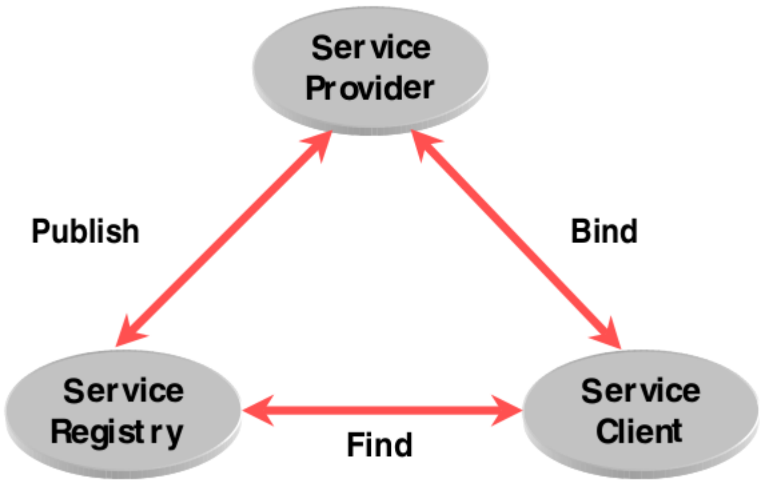


Figure 2.4: The basic Service Oriented Architecture [54]

higher level messaging protocols such as SOAP (simple object access protocol) [68] or the REST (representational state transfer) [24] architectural model on top of the HTTP protocol. They enable data to be easily passed between systems by utilising the protocol that powers the Internet. SOAP employs a particular XML envelope containing the message to be sent. The envelope contains all metadata about the message such as the action the message is intended to have and security details, among others. The envelope containing the message is then sent over HTTP as an HTTP body using the HTTP post verb. The HTTP protocol is just a mechanism to carry the message. All the message metadata and data are contained within the SOAP envelope. REST differs from SOAP as it does away with the need for an envelope. REST relies on using a particular constrained set of verbs to act on particular resources. When this model is applied to HTTP this means that data is sent and requested using the HTTP verbs (GET, POST, PUT, DELETE, etc.) and any data is conveyed using the HTTP body or if it is a query, in the URL as URL or query parameters. REST over HTTP differs from SOAP in that it does not specify an envelope in which the message is to be sent but rather makes use of the protocols constructs to convey message metadata. The functions of the underlying HTTP protocol are mapped to the RESTful model. For example, an HTTP POST request is mapped to identify the creation of a particular resource and the HTTP body of such a request is expected to contain the information of the resource to be created. The URL of the request identifies the resource to be acted upon.

SOA enables a number of key benefits including: loose coupling between services; location transparency; and composability of services [64]. A core benefit of SOA is its open nature. Each service is independent and provides an open interface to the other systems in the SOA. This increases maintainability as each service only deals with a specific concern and it also enables flexibility as new systems can easily consume the service provided by the SOA to add functionality. SOA is also scalable as services may be spread out over a number of different servers as required.

2.3.2 Enterprise Service Bus

The Enterprise Service Bus (ESB) is an architectural model used to instantiate the SOA paradigm [60]. The ESB approach dictates that a central bus (a middleware component) provides the communication mechanism and communication services between each of the software components [10, 60]. An ESB provides various functions including message mediation, service orchestration and security, among others. An ESB's internal components should be loosely coupled to prevent them from becoming a central point of failure. The ESB's main function is to deliver messages placed onto its bus to the intended recipient of the message and to deliver the message in a format that the recipient can process and extract meaning from (syntactic and semantic interoperability). In order to do this it performs mediation functions on messages. Figure 2.3c on page 24 shows an example of this architecture.

Orchestration refers to the execution of a business process that can interact with both internal and external services in order to complete that business process [56]. It can be made up of a series of steps containing business logic. Orchestration assists with interoperability by decomposing a transaction into manageable units that can be handed off to external services. This enables a separation of concerns and allows multiple, simpler services to exist. This orchestration is able to compose several external services together to provide higher level functionality. This enables the client systems to be simpler as they do not have to reimplement this composition logic or know how to contact the other services required for this orchestration. An example of orchestration within the health domain is as follows:

A message containing a patient's encounters is received by the ESB and it must be stored in a clinical data store. In this case the message needs to be validated and enriched with additional patient demographic information before it is sent to a service to be stored. An external service can be called to fetch this additional information and enrich the message before it is stored in the clinical data repository. Multiple services may be called depending on the implementation needs.

Mediation refers to the processing of data so that it can be communicated from one interface to another. "This term includes the processing needed to make the interfaces work, the knowledge structures that drive the transformations needed to transform data to information, and any intermediate storage that is needed" [65]. This enables numerous heterogeneous systems to communicate with each other using different message formats and protocols. For example, if a message arrives at the ESB in a standard form but the service to which the message needs to be sent only supports a custom XML message then mediation is the process of mapping the message content and transforming the message into the custom XML format.

The ESB approach provides a number of key benefits. An ESB can offer a central security layer to ensure that the applications interoperating with each other are authorised to do so. It also provides location transparency, central monitoring and administration functions. An ESB also provides mechanisms to do message transformation, message routing and accept messages using a number of different protocols. In addition it also provides intermediate processing of messages such as orchestration and meditation. The key benefit that an ESB adds to SOA is the ability to easily connect heterogeneous systems that may communicate using different message formats. In addition, service orchestration allows services to be composed together to provide higher level functionality and since this functionality is implemented in the central bus, many service consumers can make use of these higher level functions without having to reimplement the logic.

The challenge with the ESB approach is that it centralises a number of functions. While this is good for simplicity and re-use it can also hinder development and expansion of the system. This is because the ESB becomes a dependency knot that ties all services together. In order to change the functionality of the overall systems, the ESB must be altered to suit the new requirements. This can be difficult to manage especially if there are a number of partners involved.

2.4 Interoperability in health care systems

In this section previous work on architectures that facilitates interoperability between disparate health information systems is explored. Specific architectures and implementations are examined to identify how syntactic and semantic interoperability problems have been handled previously. Focus is placed on established ESB approaches as well as emerging approaches.

2.4.1 Canada Health Infoway: Mohawk reference implementation

Canada Health Infoway (CHI) has developed an Electronic Health Record Solution (EHRS) blueprint [9] for health information exchange (HIE) in Canada. This blueprint outlines the architecture of a regional health information system, including the services and transactions that are required as well as how they work together to provide a coherent HIE. Amongst others they describe some basic services commonly used in an SOA-based HIE. A subset of the most fundamental services are enumerated below:

- **Client Registry** - registers, stores and uniquely identifies patients.
- **Provider Registry** - registers, stores and uniquely identifies health providers.
- **Facility Registry** - registers, stores and uniquely identifies health facilities.
- **Shared Health Record** - stores and retrieves a patients clinical information.
- **Terminology Service** - stores common clinical and administrative terminology and maps between different code systems.

The blueprint also introduces a central component called the HIAL (Health Information Access Layer). This component manages the communication between the service providers and service consumers. Figure 2.5 on page 30 shows a conceptual view of the EHRS architecture. A reference implementation of the HIAL has been built using Microsoft BizTalk, a

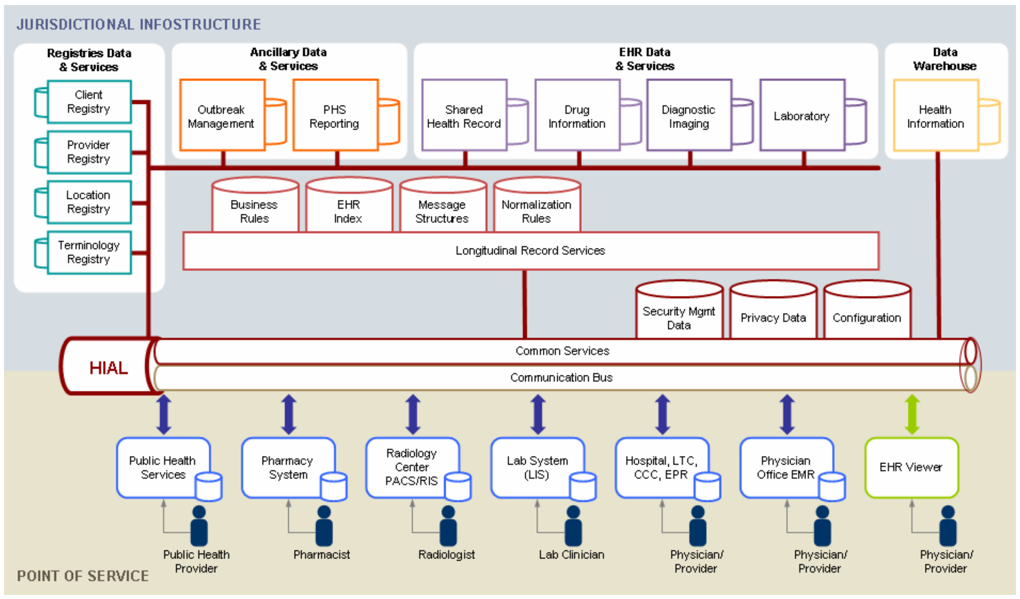


Figure 2.5: Canada Health Infoway: EHRS Conceptual View [9]

large proprietary application designed to provide integration and connectivity between applications. BizTalk consumes transactions received from client applications and transforms and orchestrates these transactions as required. SOAP web services are used to transmit messages. The messaging specification defined by the EHRS blueprint is HL7 version 3. HL7 version 3 [32] is the third version of the standard for transmitting clinical information. It makes use of structured XML that maps to a reference information model to format its messages.

The Health Information Access Layer (HIAL) utilises an Enterprise Service Bus (ESB) architectural model to facilitate interoperability. The architecture allows for disparate systems to provide services to clients though a well defined access layer. This layer is provided by the ESB software, BizTalk.

Messages that enter the HIAL are transformed into a canonical form that is based on the HL7 Reference Information Model (RIM) using on-ramp and off-ramp transformers. This enables semantic interoperability between each of the disparate informations systems. However, this has a restriction that all messages that needs to be exchanged must be modelled using the HL7 RIM. The HL7 RIM is a generic data model and can therefore express any information that needs to be stored. This common format enables the system to have a consistent canonical form for all data that flows through the system.

Once a message has been converted into the canonical form the HIAL is able to orchestrate that transaction by looking at flags set for that message. The HIAL handles communication with external services and when the transaction is complete it returns the result to the service consumer. This enables flexibility as the service providers are loosely coupled to the HIAL architecture and as such can be interchanged without affecting the rest of the architecture. The HIAL also handles authorisation, authentication, message logging and auditing services.

This architecture provides a high level of separation of concerns as each system is responsible for a particular function and this may be implemented in any appropriate technology. It also provides loose coupling between the components as all communication between components takes place using standardised messages communicated via the HIAL. This allows components to be easily modified or replaced if needed. Location transparency is also provided as client systems use a single unified interface (the HIAL) to access the components of the HIE. The HIAL also reduces the complexity of services by providing common services such as security management. These benefits come at the cost of additional architectural complexity and extensive centralisation of data.

A weakness of this architecture is that it only supports messages in the HL7 version 3 format and these messages are mapped to a canonical form based on the HL7 RIM for internal use. Due to this reliance, it is difficult to add support for additional message standards that may become prevalent in the future. It is desirable to support multiple messaging formats due to the ever changing nature of the health information systems environment, as well as the fact that no *de facto* messaging standards have emerged in the health domain. This architecture has been designed specifically for the Canadian use case. In our research, we attempt to create a more general architecture that may apply to multiple environments, particularly LMICs.

Overall, the architecture contains a number of foundational components that are highly desirable and that have become the foundation of a number of other architectures. In addition, a description of an ESB-based architecture for interoperability between health information systems has been provided. This architecture describes the major functions that a central component should perform within an HIE. These functions include routing, transformation, logging and auditing of messages as well as access control.

2.4.2 Health Service Bus by Ryan *et al.*

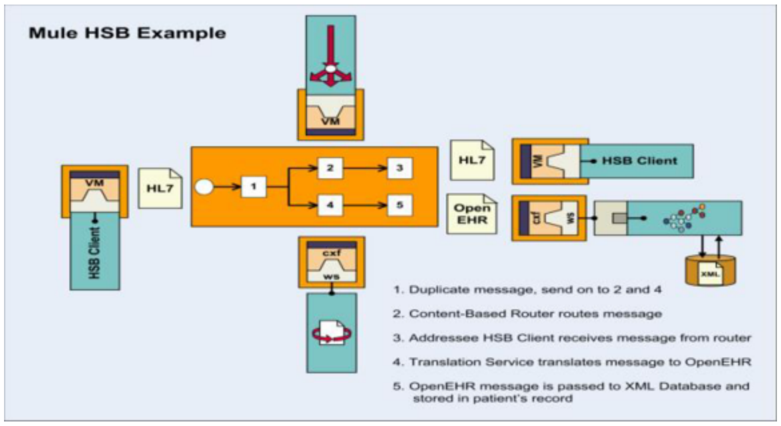

Mule HSB Example
VM
HL7
HSB Client
1
2
3
4
5
HL7
Open EHR
HSB Client
cxf
ws
XML
1. Duplicate message, send on to 2 and 4
2. Content-Based Router routes message
3. Addressee HSB Client receives message from router
4. Translation Service translates message to OpenEHR
5. OpenEHR message is passed to XML Database and stored in patient's record

Figure 2.6: The Health Service Bus by Ryan *et al.* [59]

Ryan *et al.* [58, 59] describe a service-oriented architecture for communication between disparate health information systems. They describe a component called the Health Service Bus (HSB) that implements an ESB to allow different systems to communicate with each other within an HIE. Figure 2.6 on page 32 shows an overview of this architecture. The HSB provides a link to three core services: a content-based router that is able to route messages from sending client systems to receiving client systems; a translation service that is able to translate messages between HL7 version 3, HL7 version 2 and OpenEHR archetypes; and an Electronic Health Record (EHR) that is able to store clinical information using OpenEHR archetypes in an XML Database. For data representation, HL7 version 3 is used to provide syntactic interoperability and the Systemized Nomenclature of Medicine - Clinical Terms (SNOMED-CT) ontology and code system provides semantic interoperability.

Ryan *et al.* show that the HSB can be extended to a number of different domains of health care by connecting additional types of client systems. These systems can make use of the HSB's core services in order to communicate more simply. The architecture delivers messages in a peer-to-peer approach between the different client systems. However, there is no explicit orchestration mechanism described in the HSB or descriptions of what sort of transactions are supported. The key contribution of this work is the concept of exposing some common key services on the bus so that communication between systems could occur more simply. Also, due to the use of HL7 version 3 combined with the SNOMED-CT code system semantic interoperability has been enabled within the exchange.

The core difference between this architecture and the CHI architecture, discussed above, is that the HSB clients interact in a peer-to-peer fashion via the HSB component whereas in the CHI architecture (described above) the HIAL makes information available to clients by storing information in central services. The CHI architecture requires that the clinical data that is sent from clients is stored in a central Shared Health Record (SHR) and if another client would like access to that information they would have to query the SHR. These messages are routed by the ESB component (the HIAL) to the SHR so that it may respond to the query.

The HSB architecture describes how interoperability can be achieved for a specific use case by using an ESB and HL7 version 3. We believe this could be extended to provide a more general approach, as well as provide support for multiple standard messaging formats. There are currently no *de facto* health information standards, so support of multiple standards is vital to support future needs and the ever changing nature of health information systems. Ryan *et al.* have made some progress to solve this by implementing a translation service, however the details of how this would allow multiple standards to be supported is missing. A clear understanding of how the internal components of the ESB are architected and how this architecture can be adapted for future needs is missing.

2.4.3 Xu *et al.* Mediator Services

Xu *et al.* define a multi-agent based coordinator and an architecture for service mediators that together provide a base architecture and implementation that enables interoperability between heterogeneous health information systems [70]. The coordinator called "Pilot" breaks down a service request from a service consumer into elementary steps that are then executed by a specialised agent that knows how to handle that specific task. Figure 2.7 on page 34 shows an overview of the architecture of the system. Each of these agents then utilizes a service mediator [71] that transforms the message into an intermediate (canonical) form; understands its semantics; and can transform the message into a form that the service provider can understand. The mediator also facilitates the communication between itself

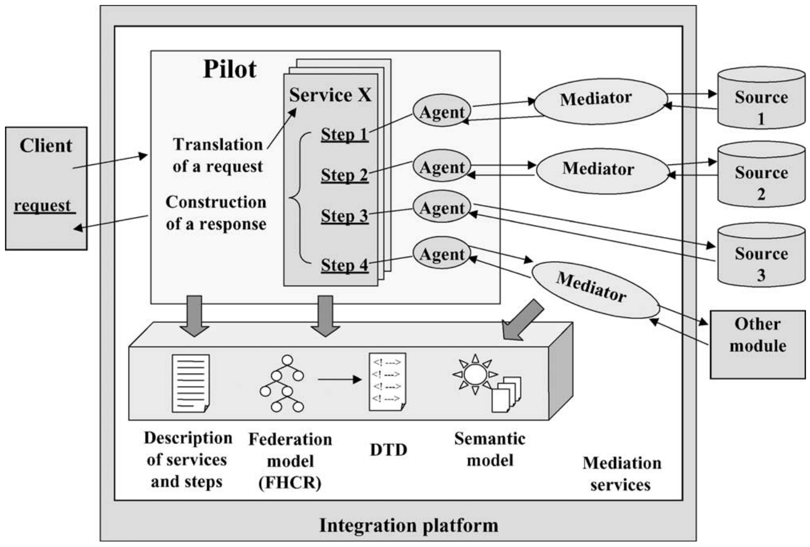


Figure 2.7: Architecture of the mediator service component [70]

and the service provider using a method that the service provider supports. This architecture does not specify a specific mechanism to achieve semantic interoperability. This is left up to the implementer of this architecture to decide what suits the problem best. The study identifies XML as a possible mechanism for syntactic interoperability, however, the use of XML alone does not lead to syntactic interoperability as we identified in Section 2.2.2. The nature of XML enables it to express any data in semi human readable and machine readable forms. A schema needs to be generated to restrict XML before data can be exchanged and be processed effectively by the receiving system.

The study also defines elementary requirements for service mediators: mediators should translate messages between different syntaxes; encapsulate APIs for communication with different service providers; and transform messages into a semantic equivalent between various reference systems [70]. We found this classification useful while identifying the practical problems faced when designing an architecture for health information mediation.

This implementation provides generic reusable components that allow the system to orchestrate services and communicate with service providers easily by providing mediators to handle the complexity of different communication protocols, message syntaxes and semantics. However, it fails to note that the same problems associated with interoperability between the coordinator and service providers are also true for interoperability with service consumers. Mediators are needed on both the client side as well as the service side. Clients can also communicate using varying communication protocols, message syntaxes and semantics as client systems could be legacy, propriety or have technical challenges that could prevent the full implementation of the format expected by the coordinator.

Xu *et al.* provide a generic architecture that allows messages to be orchestrated and mediated as needed for a specific transaction. The architecture is independent of the messaging format used which allows it to be used with a number of messaging formats. The architecture is also independent of the types of orchestration that need to occur allowing the architecture to be generally applicable for multiple use cases.

Xu *et al.* also note future work of logging, authentication and supplying audit trails for service calls. These issues are further discussed in the following chapters.

2.4.4 NEHTA

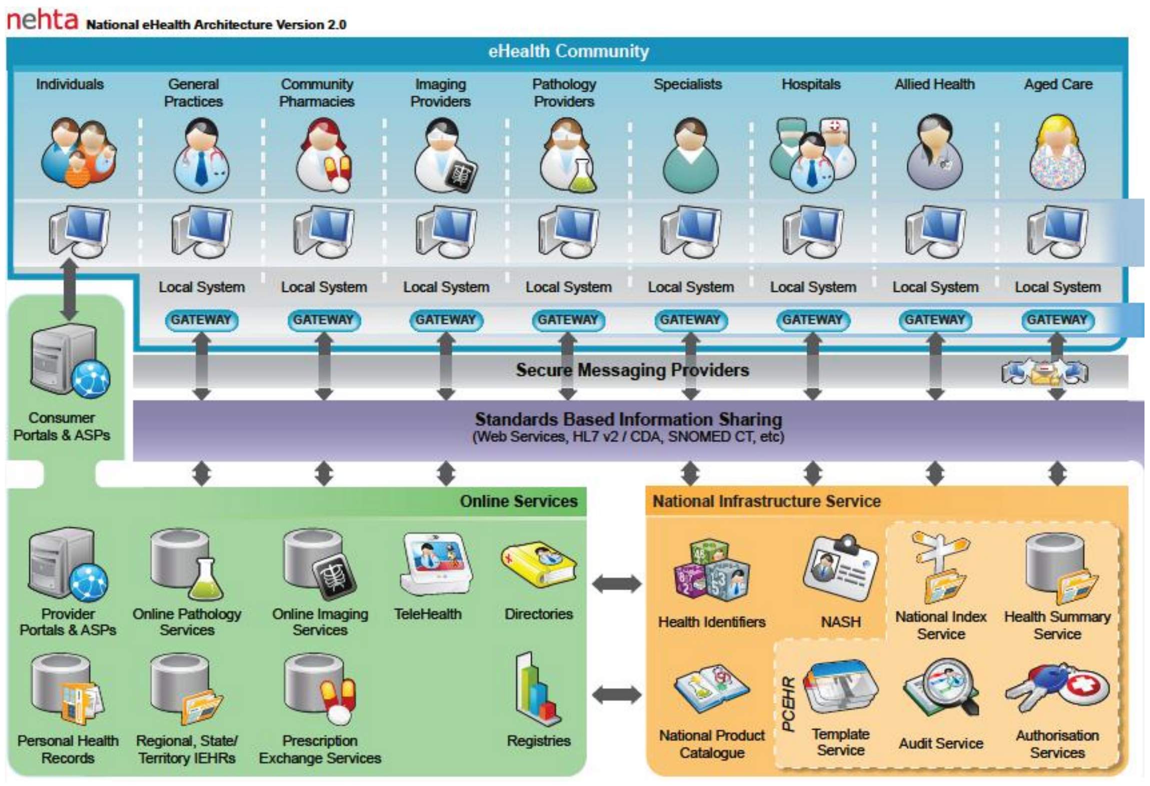


Figure 2.8: NEHTA National eHealth Architecture. [50]

The National E-Health Transition Authority (NEHTA) is an organisation that leads the development of Australia's national eHealth strategy. NEHTA is funded by the Australian government and all state and territory governments. Its role is to lead the uptake of eHealth systems of national significance and to coordinate the progression and adoption of eHealth by delivering urgently needed integration infrastructure and standards[3].

NETHA maintains a document, the NEHTA blueprint, which describes the architecture of the national eHealth system for Australia [50]. This architecture describes the foundational services, infrastructure services and standards for information sharing needed to support Australia's vision of a national eHealth architecture. A diagram of this is shown in Figure 2.8 on page 35. This architecture takes a standard service-oriented architecture approach. All of the required components of the architecture are exposed as services that the Australian eHealth community can invoke via standards-based interfaces. NEHTA does not describe any orchestration middleware to assist with adapting client systems to make use of the standards based interfaces or to coordinate the interaction with the multiple services available.

This approach would work well in an environment where eHealth applications are common in most health facilities and resources are available to improve these systems to enable them to participate in the national architecture. However, we question whether such an approach would work effectively in a LMIC environment where resources to change eHealth systems are scarce. In a low resource environment it is likely more efficient to reduce complexity at the point of care and manage complexity at a central point where resources can be shared among point of care systems. This however, in turn, requires additional coordination at the national level.

The NEHTA approach identifies and specifies many of the key foundation components of a national eHealth architecture similar to the work Canada Health Infoway has done with the Canadian EHRS blueprint. Its main limitation, when considered within low resource environments, is the complexity it places on the systems at the point of care.

2.5 Outstanding issues with ESB based Distributed Systems for HIS

We have identified that the key concerns for open distributed health information systems are those of interoperability, adaptability, scalability and fault tolerance. In light of these

[3]See NEHTA homepage: `http://www.nehta.gov.au/about-us`

various issues, a number of architectural paradigms have emerged. One of the most popular in the enterprise space is the service-oriented architecture. This paradigm focusses heavily on modularity and separation of concerns between services, as well as the use of a common set of protocols which systems within the architecture use to communicate. An implementation of this paradigm is the Enterprise Service Bus (ESB). This is a specialisation of the SOA where a central component is responsible for the routing of messages, as well as the orchestration of complex workflows between services as required by health information systems. These architectures can assist in creating distributed health information systems. An ESB is particularly useful in the health domain as it performs functions such as message orchestration and mediation. These functions are useful due to the fact that, as we have previously state, there are no *de facto* standards available in the health domain and due to the high complexity of health data that needs to be transmitted. This is particularly true for LMICs where technical resources are scarce and complexity should be kept to a minimum.

A fully interoperable system will exhibit technical interoperability, syntactic interoperability and semantic interoperability. Technical interoperability has been largely solved by the introduction of platform independent protocols such as TCP/IP, as well as higher level protocols that build upon this such as HTTP and web services. This leaves the majority of work to be done in the area of syntactic and semantic interoperability. Although many approaches and standards exist, none have emerged as *de facto*. Some standards have been implemented successfully such as HL7 version 2 for syntactic interoperability, and ICD-10 and SNOMED-CT for semantic interoperability. However, legacy systems often do not support these and there is much optionality and ambiguity in these standards that makes interoperability difficult. Often standards need to be specified as profiles to gain full semantic interoperability.

We have identified a number of architectures that enable the creation of an HIE using the ESB paradigm or functions thereof. Most of the existing approaches are built to solve a specific problem either by defining how specific dimensions of interoperability can be solved or by describing an architecture to enable a health information exchange to be more easily constructed for a particular environment and set of use cases. Also, these architectures have not been applied to LMIC previously. In addition, many of the architectures presented above rely on specific standards to achieve interoperability. Due to the variety of standards available in the health domain and given the ever changing environments that HIEs are deployed in, we believe that multiple standards should be supported.

The architectures that were studied provide a number of features that enable interoperability, however, none address all of the challenges that have been identified above. Particularly, the architectures were often designed for particular use cases and their applicability in LMICs has not been considered. The health domain is missing a generic, use case independent and standards independent architecture for enabling interoperability between health information systems within an HIE, particularly for LMICs.

Chapter 3

Research Design

The central concern of this work was to enable interoperability between disparate health information systems in sub-Saharan Low to Middle Income African countries (LMICs). Current thinking in enabling interoperability for health information systems and exchanging health information is the use of ESB based health information exchanges [59, 72]. This is a technology stack that connects diverse information systems and enables the sharing of patient level and aggregate data. However, this approach has been used only in HICs and its efficacy in LMICs has not been demonstrated. It is known that the nature of the health systems and the socio-political and economic environments of HICs and LMICs differ considerably and thus it cannot be assumed that solutions that worked in one setting will necessarily work in the other.

It is a challenging task to develop systems as complex as Health Information Exchanges that will have wide applicability in LMICs. The approach taken in this work was to develop an interoperability solution within a particular, representative LMIC namely Rwanda. The Rwandan case study was used to understand the requirements of such environments; learn from the experience of implementing interoperability solutions within low and middle income environments and to determine the technologies that work within such environments. Case study research is well suited to understanding innovation in information technology, particularly within a specific environment [15]. A careful study of the Rwandan environment and its requirements was made. Existing approaches from HICs were studied and an architecture was designed that addresses architectural concerns derived from the Rwandan requirements. Attempts were made to ensure that the architecture had certain desirable characteristics that would make it useful in solving similar problems in other low and middle income environments.

A reference implementation of the architecture (the OpenHIM) was developed and deployed in Rwanda to serve as a proof of concept. This experience was used to explore the limitations and impact of the architecture. Subsequently, the applicability of the architecture to other LMICs was demonstrated by studying its adoption in the MomConnect project in South Africa.

In addition, the quality characteristics of the architecture were further analysed using a standard architecture analysis method (ATAM). The main quality attributes that were analysed are modifiability; scalability and performance; and security. These attributes were identified as part of the ATAM process for their suitability to the concerns derived from the Rwandan case study.

3.1 The Rwandan Case Study

Between 2011 and 2012, the Ministry of Health of Rwanda along with a consortium of partner organisations developed a pilot HIE for the Rwamagana district of Rwanda. The RHIE stakeholder group consisted of Rwandan ministry of health officials, domain experts, research groups and software development and implementation organisations. This group included experts from Mohawk College in Canada, the Regenstrief Institute in the United States, Jembi Health Systems NPC in South Africa, IntraHealth International, Sysnet International, InSTEDD and the HeAL Laboratory in South Africa. This group collectively has extensive expertise in designing and implementing HIEs.

The Rwandan health care environment is made up of a number of small clinics and health posts spread throughout the districts of Rwanda. Most of the population receives primary care from these health posts and clinics. Each district has a larger district hospital to which patients can be referred to for further care. Generally, there is no automatic mechanism to share a patient's information between all these health facilities. The patient is asked to carry a referral note that gives a basic description of what they are being referred for. The receiving facility does not get any of the patient's past medical history and has to attempt to recapture this from the patient. This wastes valuable resources in facilities where there are already too few staff. Additionally, if patients move to different clinics they are often lost to follow up at the original clinic and it becomes difficult to track that patient's care over time.

Currently, the ministry is focussing its interoperability and data integration efforts on a single pilot district: the Rwamagana district. This district has fourteen health centres and one district hospital. The initial pilot is restricted to a single health service - collecting and sharing clinical information for Maternal and Child Health (MCH). The clinics within this district had already deployed an open source electronic medical record (EMR) system called OpenMRS [43]. Rwanda also utilises a short message service (SMS) based data collection tool called RapidSMS[1]. This tools empowers community health workers to visit pregnant women at their homes in their villages and allows them to submit information about these women via their mobile phone using SMS technology. The ministry is aiming to allow information to be shared between all of these point-of-care systems. The key outcomes are the ability to feed clinical information that the community health workers capture in the field into the clinic's local EMR system and to allow clinics to share information with other clinics and the district hospital.

The Rwandan Health Information Exchange (RHIE) enables the sharing of patient demographic information as well as their clinical information between different health facilities within the pilot district. It utilises a number of hosted infrastructure services to accomplish this. These services include client (patient), provider and facility registries as well as a shared health record and a terminology service. This research focusses on the architecture to facilitate interoperability between the point-of-care systems and the infrastructure services.

3.2 Deriving architectural concerns from the RHIE requirements

The RHIE stakeholder group took an iterative approach to eliciting the requirements for the RHIE project. This became our primary source of knowledge from which interoperability concerns and challenges were derived. The RHIE requirements were gathered through a number of in-country stakeholder meetings where the country's needs and challenges were identified. A number of domain experts were present in the stakeholder group to ensure industry best practise was being applied. Techniques such as user and stakeholder interviews, workshops and brainstorming sessions were used to help elicit the requirements.

[1]See `https://www.rapidsms.org/`

Due to the expertise of the groups that formed the stakeholder group, the requirements that were produced are considered to be current best practice from the combined learning and experience of the multiple groups involved. They are thus not only representative of the RHIE use case but are also representative of the lessons learnt by a variety of organisations.

Additionally, a prototype HIE was developed and demonstrated at a stand at the 2010 MedInfo conference[2] held in Cape Town, South Africa. This prototype was based on the Canadian HIE blueprint [9] and utilised some components developed by MOHAWK college in Canada. To facilitate interoperability within the HIE an ESB-based software component, called the health information access layer (HIAL), was used. The prototype proved to be a good mechanism to identify additional requirements and gave the project group a better understanding of the problem. It highlighted the problem that the messages between the point of care systems and the HIE should be kept as simple as possible to enable systems to connect more easily. It also highlighted the fact that multiple message formats should be supported as a country should be able to choose the format that fits their requirements and, importantly, would be implementable with the resources available within their environment.

Once the RHIE requirements were better refined the stakeholder group began constructing high level requirements documents and UML use case descriptions [51]. They were presented and discussed at the in-country meetings with the country stakeholders and discussed as part of the project team. Once the defined use cases were finalised, technical system use cases and sequence diagrams were developed for each of the components of the architecture.

The practical experience, knowledge, requirements and challenges that arose from the RHIE were studied and used as a departure point for this research. These inputs were generalised into high level architectural concerns for an ESB-based architecture for enabling interoperability within HIEs in LMICs. These extracted concerns are a subset of the most fundamental requirements gathered by the RHIE team and relate only to the ability of information to be shared among the various health informations systems.

[2]See `http://www.imia-medinfo.org/medinfo2010/`

3.3 Architecture design and implementation

Multiple previous health information mediation architectures have employed the ESB architectural model with success [59, 72]. In this research we explore the application of the ESB architectural model to HIEs within LMICs. The ESB approach also fits with the centralised, middle-out approach, that has been suggested for LMICs [14]. Thus, the ESB architectural model is used as a base on which an our architecture for health information mediation is designed.

The architectural concerns derived from the RHIE requirements were used to drive the design of the architecture, however, we endeavoured to ensure that the design decisions would apply to a wide variety of similar environments. The RHIE interoperability requirements were thought to be representative of the requirements encountered in other LIMCs in sub-Saharan Africa. An iterative approach was used to develop an architecture (the HIM architecture) that attempts to address these concerns. The design was captured using elements of UML such as component diagrams and activity diagrams and is described as an ISO/IEC FDIS 42010 architecture description [39, 21].

The Rwandan HIE was used to determine the suitability of the architecture in meeting the interoperability needs of a real world HIE. Care was taken to ensure the architecture was generic and modifiable such that it could be applied in a variety of different contexts. The goal was to design an architecture that was suitable for a large number of LMIC environments.

3.4 Analysis of suitability to LMIC environments

In order to determine the suitability of the architecture to LMIC environments, it was implemented in the case study environment and its adoption within other LMIC environments was studied.

To demonstrate that the architecture is able to function within the RHIE, a reference implementation, the OpenHIM, was developed and deployed along with a number of other infrastructure services developed by a number of the RHIE partners. The OpenHIM enabled point of care systems to more easily connect and exchange information with these infrastructure services. This provided a good opportunity to ensure that the architecture could be practically applied to a real world use case; that it addressed the identified interoperability concerns and that it was suitable for LMICs.

In order to determine if the architecture is not only applicable to the RHIE, where most of the requirements originated, it was also applied to another HIE problem in a different environment, i.e. South Africa. The MomConnect project was used to determine the suitability of the architecture for other environments. The MomConnect project enables pregnant mothers to be registered via their mobile phone. This registration data is sent to an HIE where it is processed and saved. This project is studied to show the re-usability of the architecture and to demonstrate how it could be applied to a different environment.

3.5 Analysis of architectural quality

A quantitative evaluation of the effectiveness of the architecture in the Rwandan HIE is difficult to perform due to the subjective nature of assessing the impact of the HIM architecture and the lack of concrete data to show the impact on health outcomes. Also, a direct link between health outcomes and a particular architecture of a single software component would be difficult to achieve. Instead a qualitative analysis of the architecture was performed based on the experience gained deploying the reference implementation in Rwanda.

Performing an effective software architecture analysis can be difficult due to the subjective nature of determining an architecture's quality. Qualitative methods such as the cost benefit analysis[3] perform well when the benefit can be easily quantified, for example an increase in revenue. However, this analysis method says little about the quality of the architecture itself. In addition, the benefit of an HIS is often improved health outcomes which can be much more difficult to quantify and relate back to a specific HIS implementation.

Qualitative analysis methods attempt to enable analysis even when an architecture is complex and when its quality is difficult to quantify. Such methods provide a formal method to qualitatively analyse architectures. A number of scenario-based qualitative analysis methods exist for analysing architectures. These methods include the following:

[3]See `http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm`

- SAAM, Software Architecture Analysis Method [40].
- ATAM, Architecture Trade-off Analysis Method [41].
- ALMA, Architecture Level Modifiability Analysis [5].
- FAAM, Family – Architecture Analysis Method [19].

Ionita et al provide an comparison of these analysis methods [38]. SAAM is a classical analysis method and many of these methods are extensions of SAAM. Most of the above methods, such as SAAM and ALMA, use modifiability as the core quality attribute against which an architecture is measured. ATAM is a successor to SAAM and extends it to provide a framework whereby a number of stakeholder defined quality attributes can be elicited and used to measure the quality of an architecture.

The ATAM was chosen as the method to use for architecture analysis as it provided the most comprehensive approach to architecture analysis. Multiple quality attributes are considered such as performance, security and modifiability. ATAM also helps to identify the trade-offs between these quality attributes as well as risk and sensitivity points of the architectures. This was found to provide a full and informative analysis of the architecture in question.

The ATAM is designed to give organisations embarking on a creating a new piece of software a method to evaluate and strengthen their architecture. For this use case an architecture evaluation group is formed to help take the project stakeholders through the process of performing the ATAM. This evaluation group is often external to the project to ensure objectivity. In this research the use of ATAM is different from this standard use. Here the ATAM was used to perform a retrospective analysis of the HIM architecture. Knowledge about the RHIE project was used to provide the stakeholder perspective and the knowledge of the HIM architecture and the ATAM was used to map those perspectives on to the architecture.

The ATAM outputs include: the **tradeoffs** between particular architectural decisions and the effect on multiple quality attributes; the **risks** associated with the architecture; as well as any **sensitivity points** where an architectural decision directly influences a particular quality attribute. Armed with these outputs the strength and suitability of the architecture to address the architectural concerns is determined.

The ATAM process helped to identify the following quality attributes that were used in the analysis. These are shown here in order of importance to the suitability of the HIM architecture:

1. Modifiability
2. Scalability and performance
3. Security
4. Re-usability
5. Availability

The architecture is analysed against each of these quality attributes by following the ATAM process. The ATAM process specifies that architectural approaches for the architecture in question, as well as a number of desired scenarios that the architecture enables, be enumerated. Then, the scenarios are categorised under the identified quality attributes. To illustrate this clearly a utility tree is drawn. This digram shows how particular scenarios relate to the quality attributes identified for the architecture. The architectural approaches are then mapped onto the scenarios that they enable. In doing so, trade-offs between the quality attributes, attribute sensitivity points and architectural risks can be determined. These outcomes are utilised to reason about the architecture's suitability and helps identify points of weakness.

3.6 Limitations and summary

The methodology described above is considered to be suitable to determine the extent to which the research objectives were achieved, however, there are some limitations to the above methodology that should be pointed out. Software architectures can be difficult to analyse objectively and quantitatively. Thus, a qualitative analysis was performed on the architecture. However, these views of the architectures may be subjective to the views and context of the author. An attempt has been made to keep the analysis as objective as possible, however, some subjectivity may still exist.

In this research the RHIE is used to extract and generalise architectural concerns for LMICs. However, these generalised concerns may not fully capture the needs of all other environments or situations. The challenges faced in Rwanda may not map to those faced in another environment. However, they are believed to be representative of common problems faced with interoperability architectures. The literature reviewed in Chapter 2 help to show that the challenges faced in Rwanda are not unique and will likely apply to other environments.

Chapter 4

Design of the HIM Architecture

The primary concern of this work was to address the interoperability challenges between health information systems within an Health Information Exchange (HIE) in low and middle income countries. To this end, we designed an architecture to facilitate and simplify interoperability within HIEs. We took, as a point of departure, the requirements of the Rwandan Health Information Exchange (RHIE) as elicited by the stakeholder group of the RHIE project. These requirements were then analysed and distilled into a set of architectural concerns to inform the creation of an architecture to facilitate interoperability between HISs within an HIE. These concerns address the foundational challenges associated with interoperability as reported in the literature. This chapter first motivates these concerns and then describes this architecture in detail.

4.1 A national health information system for Rwanda

The Rwanda Ministry of Health (MoH) has already made significant progress in developing a National Health Information System (NHIS), that includes, among others, community health systems, health management information systems and the national roll-out of an electronic medical record application [45]. The Rwanda Health Information Exchange (RHIE) project, led by the Rwandan Ministry of Health and supported by a consortium of partners and donors has developed an Health Information Exchange (HIE) to facilitate interoperability between individual health information systems and applications.

Implementation of the Rwandan HIE was achieved in several phases. The first phase was to implement the foundational components, including client, professional and facility registries, a terminology service and a shared health record. These services assist in enabling interoperability between point of care information systems supporting maternal health in the fifteen health facilities in the Rwamagana district. There are two types of point of care systems in use in this district in Rwanda. These are implementations of OpenMRS [43, 3, 61], an Electronic Medial Record (EMR) system and RapidSMS, an SMS based data collection tool. RapidSMS allows community health workers (CHWs) in Rwanda to submit maternal and child health information to a central server using SMS messages from mobile phones. There are many CHWs within Rwanda and this information plays an important role in monitoring the progress of pregnant women and the health of children where frequent visits to clinics are not possible. In subsequent phases, the HIE will need to accommodate other domains of care and have the ability to scale nationally.

The HIE's main function is to enable the point of care systems currently implemented in Rwanda to connect and interoperate more easily. Using the HIE, the MoH plans to promote data re-use between the connected systems and to facilitate information sharing. It also aims to provide patients with a continuity of care record [22] to enable access to a patient's clinical information from different health facilities, thus improving the tracking of patients and reducing the number of patients lost-to-follow-up.

The first phase involves deploying a set of foundational infrastructure services that provide services to the point of care applications. The HIE will allow the systems to share clinical information and ensure that shared information uniquely identifies the patient, provider and facility associated with that clinical information within the information exchange (See Figure 4.1 on page 50).

The foundational infrastructure services are:

- Shared Health Record: this service responds to queries for an appropriate subset of the patient's longitudinal, patient-centric medical record.
- Client Registry: this service responds to queries for a patient's demographic and identifying information used to uniquely identify patients.
- Facility Registry: this service responds to queries for data on the facilities participating in the information exchange. This is primarily used to maintain current and valid facility codes required in transactions.
- Professional/Provider Registry: this service responds to queries for information about

health care professionals who work at participating health care facilities in the information exchange. This is primarily used to uniquely identify health care professionals within the HIE.

- Terminology Service: this service responds to queries about the validity of codes within code systems used by the implementation. The system stores all the clinical code systems (e.g. LOINC, ICD10 and country specific code systems) that will be used within the HIE and facilitates verification and mapping between codes.

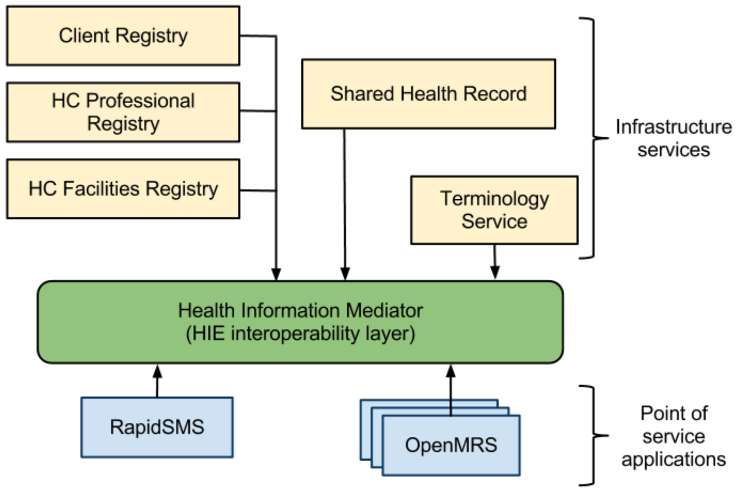


Figure 4.1: The architecture of the Rwandan Health Information Exchange

4.2 The architectural concerns for HIS interoperability

In this section, we explore each of the identified problems and challenges with interoperability between health information systems in order to draw out a set of architectural concerns that address the challenges associated with interoperability between disparate HISs in low resource settings. The Rwandan HIE use case and its requirements are used to elicit and identify these concerns.

The identified architectural concerns should not define how an HIE should be configured for a particular implementation but rather analyse only those concerns that relate to the common interoperability problems faced when constructing an HIE. The key architectural concerns that we identified are enumerated below. These are explained and justified in the following sections using the RHIE case study.

The architectural concerns are as follows: *An architecture to enable interoperability between disparate HIS for LMICs must...*

1. Facilitate interoperability between disparate and heterogeneous systems, both existing and future.
2. Balance central governance with local autonomy.
3. Adapt and scale within a changing environment.
4. Prevent local changes from propagating through the system.
5. Provide a low barrier to entry to connect new and legacy systems.
6. Be secure and auditable.
7. Be reusable across a multitude of environments.

Each of these concerns are described in detail and justified against the Rwandan HIE in the following sections.

4.2.1 Concern #1: Facilitate interoperability between disparate and heterogeneous systems, both existing and future

In the context of the Rwandan NHIS, the HIE initially allows the OpenMRS and RapidSMS systems to interoperate with the infrastructure services (client registry, provider registry, facility registry and the shared health record) in order to share information. Each system embodies different technologies and designs and the architecture for health information mediation should enable these systems to interact effectively and provide service providers and service consumers a mechanism through which they can communicate.

The architecture must provide mechanisms to allow existing disparate and heterogeneous systems to be incorporated into the HIE with minimal changes to the systems themselves while still allowing for local autonomy. The systems need to be able to grow and develop independently of the overall HIE and the other systems participating in the HIE. For example, in the RHIE the facility registry infrastructure service may want to implement new features to track new facilities. This may require updating or changing its API and data model. In addition, the architecture must be technology agnostic, with minimal restrictions on the technologies used within participating systems in order to incorporate as many system as possible. Challenges to be solved by the architecture include the full suite of technical, syntactic and semantic heterogeneity between systems.

4.2.2 Concern #2: Balance central governance with local autonomy

Central governance is an important issue to be considered for the construction of regional or national level HIEs. The ability to have central control reside with central authorities such as ministries of health can be beneficial for the coordination of an HIE as a whole but this central control should not be too prescriptive of policy and process at a facility level. This is due to the fact that health facilities often have different systems, workflows or processes in place for their particular environment and they resist central control of their workflows. However, a balance between central control and local autonomy must be struck to ensure coordination and interoperability between health facilities.

For example, in the RHIE, central governance is important for defining the transactions required for sharing information between facilities; for defining how information can be secured and securely accessed; and how information can be audited within an HIE. These are issues that are best defined centrally in order to provide a unified approach to creating an HIE [14].

A middle out approach to building a national health information system has been recommended for LMICs [14, 47]. In such an approach it is the responsibility of the central authority to specify the suite of standards to be used and the framework under which point of care systems should operate.

4.2.3 Concern #3: Adapt and scale within a changing environment

The focus of the current RHIE project is to enable the sharing of maternal health information between point of service applications in a single district. However, the RHIE architecture will also need to adapt to include new functionality as the project progresses. The HIE must be able to expand in such a way that its services may be readily expanded to other districts in Rwanda; to incorporate additional domains of health care (for example HIV and TB programmes); and to allow other systems to be incorporated as part of the growth of the HIE. Additional health information systems such as aggregate reporting systems, pharmacy systems, decision support systems and others may need to be added in time, and the HIE will be required to support these new use cases. Therefore, the HIE must exist in an environment where requirements are always changing and a health information mediation architecture should support this.

The architecture must support incremental development and evolution of a country's HIE as it expands over time. This is especially true in low-resource environments where many organisations implement disparate information systems for a variety of purposes and domains of health care [7]. An essential feature of an HIE is its ability to cope with change. The architecture must be flexible enough to deal with changing and evolving HIE requirements.

The system must also be able to scale, in terms of transaction volume, geographical locations and increased functionality to handle the demand of widespread use and to cope with the load that additional functionality places on the system. This is needed as a country's use of the HIE may increase over time. For example, the RHIE is expected to grow from connecting a few clinics to connecting all clinics in a district. Potentially, this will expand to a national roll-out. In addition, new workflows or infrastructure services may be added and the HIE must cope with this increased transactional load.

4.2.4 Concern #4: Prevent local changes from propagating through the system

In Rwanda, development teams in different organisations design and maintain participating systems such as OpenMRS, RapidSMS and the infrastructure services. Currently, there are fourteen partners working on the RHIE project with seven different development teams working on the various participating systems that must be able to develop independently without affecting other systems.

Participating systems will need to balance local requirements and NHIS requirements, but from a practical perspective, development teams will often prioritise local needs and requirements. Changes to participating systems should have minimal effect on other systems within an HIE and systems must also be protected as much as possible from changes to infrastructure services. All systems must still maintain a large degree of local autonomy, especially since these systems are implemented and maintained by a variety of disparate organisations. An architecture that facilitates interoperability between multiple disparate systems must enable this to occur.

4.2.5 Concern #5: Provide a low barrier to entry to connect new and legacy systems

Implementing partners have development teams distributed around the world with varying degrees of expertise and technical skills. Interoperating with the infrastructure services in order to share information must be simple and require minimal effort for both current and future technical teams. A number of existing health information systems including the OpenMRS implementations and the central RapidSMS server implementation existed before the Rwandan HIE was conceived. These systems should not require major changes to be incorporated into the HIE.

The architecture should reduce the burden of connecting new and legacy systems within the HIE. The approach toward integration of legacy systems should be to 'embrace and extend' rather than to 'rip and replace'. The architecture must provide a minimal barrier to entry to incorporate a system into the HIE and reduce the overhead required to modify a particular system to participate in the HIE. This feature will maximise the existing investment in legacy applications and help prevent useful and functioning legacy applications from being abandoned unnecessarily.

4.2.6 Concern #6: Be secure and auditable

An HIE should be able to protect a patient's clinical and demographic information. Measures should be put in place to make sure that information cannot be intercepted during transmission between systems. The architecture should ensure that only the health providers that have the patient's consent can view and/or add to a patient record and that these providers are properly authenticated.

It is also vital that each transaction in a health information exchange be logged and stored with audit information so that the source of any piece of data can be traced back to a responsible party.

4.2.7 Concern #7: Be reusable across a multitude of environments

The architecture should be able to be reapplied in multiple other sub-Saharan African LMICs. Rwanda is not alone in its need for interoperable HISs. Many other environments experience similar problems, especially in other sub-Saharan African LMICs. These environments may benefit from such an architecture. This concern drives us to create an architecture that does not make assumptions about the environment in which it will be deployed. This allows the potential for lessons, skills, resources and cost to be shared among multiple environments which is highly beneficial in resource constrained LMICs.

4.3 The Health Information Mediator

In the following sections the architecture of a new software component, the **Health Information Mediator (HIM),** that enables HIEs to be more easily constructed is described. This architecture addresses the key architectural concerns described above to produce an architecture that is able to facilitate interoperability in LMICs. The design and implementation of the **HIM** is loosely based on the reference implementation of the HIAL that was developed as part of the reference application for the Canada Health Infoway (CHI) EHR Blueprint [9, 72]. In this work we attempt to expand and generalise the HIAL architecture so that it will work in multiple environments and more closely address the concerns for LMICs that have been distilled from the RHIE requirements.

4.3.1 Overall HIM architectural paradigm: The ESB

The ESB architectural model forms the basis of the HIM architecture. The ESB approach was chosen due to the fact it is a current and widely used approach for implementing SOAs that provides a central component that enables us to perform access control, prescribe data exchange formats and perform message transformation and orchestration. The central component allows us to implement the middle-out approach in HIEs. The ESB architectural model has also been applied with success previously within other health information exchanges [59, 72] and, importantly, it also addresses architectural concern #2.

An ESB provides a central bus that connects each of the infrastructure services while allowing the services to remain loosely coupled and to retain local autonomy [10, 60]. An ESB's transformation functionality will enable service consumers to be connected to service providers easily without requiring substantial re-engineering of legacy systems. The orchestration function of an ESB would also provide the ability for the ESB to simplify the interaction with service providers and therefore reduce the difficulty in connecting service consumers to the HIE.

4.3.2 Architectural viewpoints

ISO/IEC FDIS 42010 [39, 21] was used to structure the architecture description and to provide a formal language and metamodel for the architecture description. Using this language an architecture is described by presenting a number of architectural views with each viewpoint framing a number of concerns of different groups of stakeholders with an interest in the system. A viewpoint is a way of looking at systems from a particular perspective whereas a view is the result of applying a viewpoint to a particular system-of-interest [39]. Together, these views make up an architecture description. Figure 4.2 on page 57 shows the relationships between the concepts that make up an architecture description.

Based on the concerns identified in Section 4.2, three major viewpoints of the HIM architecture are described below.

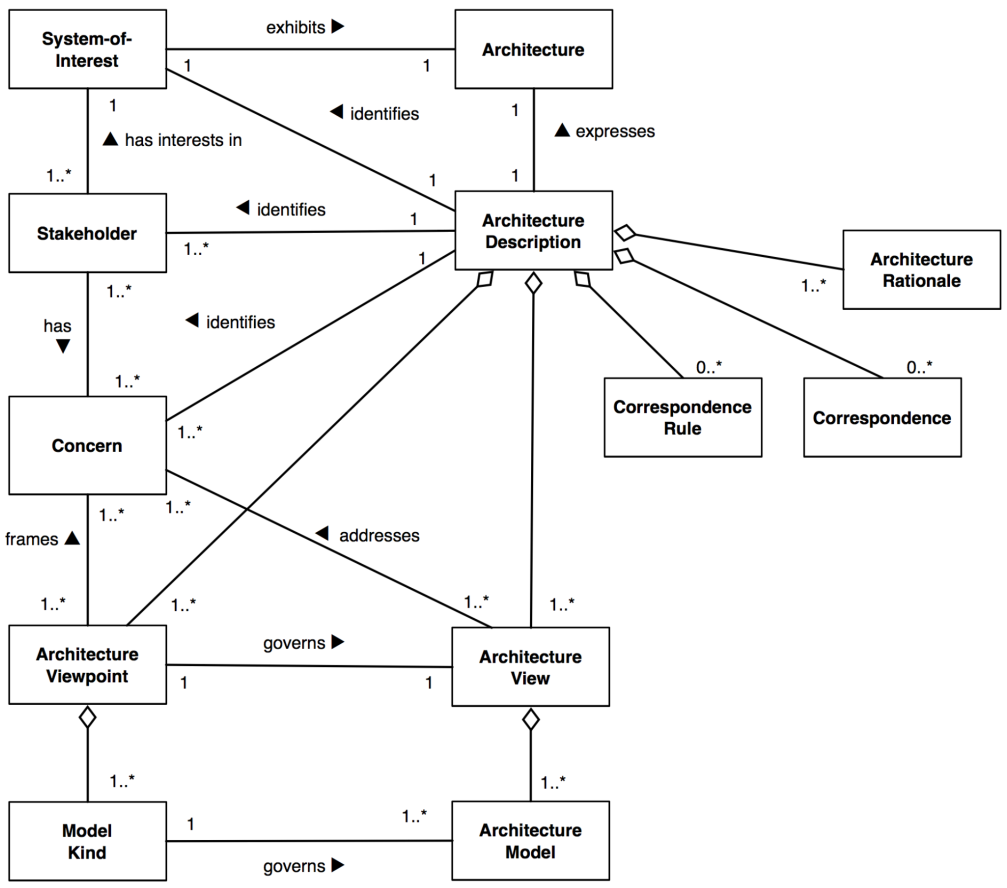


Figure 4.2: Context of architecture description [39]

Interoperability viewpoint

The interoperability viewpoint describes the overall functionality of the system in facilitating interoperability between systems. It shows the interactions between health information systems and describes the components of the system that allow interoperability to occur. The flow of messages through the system is also expressed. The models include custom diagrams showing transaction flow as well as component diagrams. This viewpoint frames the following concerns:

- Concern #1: Facilitate interoperability between disparate and heterogeneous systems, both existing and future.
- Concern #2: Balance central governance and with local autonomy.
- Concern #4: Prevent local changes from propagating through the system.

- Concern #5: Provide a low barrier to entry to connect new and legacy systems.
- Concern #6: Be secure and auditable.
- Concern #7: Be reusable across a multitude of environments.

Scalability viewpoint

The scalability viewpoint describes the scalability of the architecture. In this viewpoint, models identify how the components of the systems can handle varying loads as well as describe the deployment architectures. These describe how the components can be arranged in order to sustain high transaction loads. This viewpoint frames the following concerns:

- Concern #3: Adapt and scale within a changing environment.

Modifiability viewpoint

This viewpoint shows the architecture's ability to grow with a country's NHIS and how new services can be added or changed within the architecture. It shows how additional systems or transactions can be added to the system over time and expresses the strategies that the architecture employs to allow this to occur. This viewpoint frames the following concerns:

- Concern #3: Adapt and scale within a changing environment.
- Concern #7: Be reusable across a multitude of environments.

4.3.3 Architecture of the HIM

In this section the three viewpoints described in Section 4.3.2 are used to describe the different aspects of the HIM architecture. An architectural view is described for each of the viewpoints to show how the HIM architecture is able to address the concerns described in Section 4.2.

4.3.3.1 Interoperability View

The HIM has been designed as a middleware system to enable interoperability between participating systems and infrastructure services. It is based on the Enterprise Service Bus (ESB) architectural model.

In the HIM architecture all participating systems in the HIE are represented as services. Systems that provide services to other systems are termed service providers, while systems that make requests of other systems are termed service consumers. All service requests are made via the HIM. The HIM provides mediation and orchestration functions within the HIE.

Our approach contains four major components described by the following 4-tuple:

$HIM = \{I, A, P, M\}$

where HIM is the Health Information Mediator architecture, I is the Interface component, A is the Access Control component P is the Persistence component and M is the Mediation component.

Figure 4.3 on page 59 shows the order in which transactions flow through each of the components.

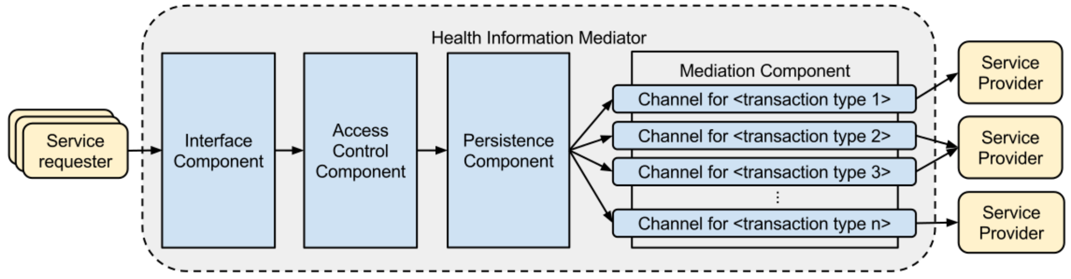

Health Information Mediator
Service requester
Interface Component
Access Control Component
Persistence Component
Mediation Component
Channel for <transaction type 1>
Channel for <transaction type 2>
Channel for <transaction type 3>
Channel for <transaction type n>
Service Provider
Service Provider
Service Provider

Figure 4.3: Overview of components in the HIM architecture

The details of each component are described below:

I - Interface Component All interactions with the service providers are carried out via the HIM. The interface component exposes an application programming interface (API) that allows systems or applications to make service requests through the HIE. It is responsible for defining and handling all incoming service requests. Service requests are received using a standard protocol (e.g. HTTP) and translated into a common internal format that is accessible by the other components in the layer (e.g. Java Objects). The request is then passed to the access control component for further processing. This component addresses concern #1 by providing a mechanism by which HISs at the point-of-care (service consumers) can communicate with service providers.

The interface component not only provides a single and consistent entry point for all service requests. A single point of access simplifies interactions with the HIE as the systems can make service requests without needing to know the location of the service providers. A single endpoint is exposed by the HIM and messages are routed to the correct service provider. A client of the HIM implementation need only know of the single endpoint exposed by the HIM in order to access services of the HIE.

The functions provided by the API are defined according to the requirements of the HIE implementation, thus they are implementation specific. In the Rwandan use case this includes functions to save and query a patient's clinical record within the shared health record and to query and update records in the client, provider and facility registries.

A - Access Control Component The access control component prevent unauthorised access to patient information. This component is responsible for dealing with two major concerns: identifying if the client accessing information is who they say they are (Authentication) and that they have the appropriate privileges and authority to retrieve or submit patient information (Authorisation). This component aims to address concern #6 and #2 by providing a mechanism by which access to patient information can be secured in a central place.

The concrete implementation of this component will depend on the transport protocol used with the HIM implementation as well as the security policies defined by the implementing jurisdiction. Thus, the details of the implementation of this component will depend on the needs of the jurisdiction implementing it.

This component also provides a central place for defining and applying advanced security policies. In this component, access to the API and access to specific functions within the API must be strictly controlled. The component also allows data-level security policies to be applied, if needed. In this research, we have not addressed the complexities of defining how these security policies could be applied in order to focus on the architectural significance of security and not the implementation details. The security requirements would differ between implementations.

P - Persistence Component This component receives authorised service requests from the access control component and starts and monitors a transaction required to fulfil the request to completion. This component addresses concern #6 and #2 by enabling requests and responses to be audited and stored in a central location for accountability purposes.

The persistence stores a copy of each transaction received by the HIM and maintains a persistent data store for the request data, the response data and metadata for each transaction. This data is stored for logging and audit purposes and can also be used to identify, handle and reprocess exceptions. This allows the administrators of the system to identify and solve recurring problems or failures. The storing of transactions should be done as an asynchronous process so that the the I/O penalty doesn't affect the transactional performance of the HIM implementation. In addition, the data store should only be accessible to the HIM implementation itself and it should block external access. The data should also be stored in an encrypted form. This ensures that any patient information contained in the stored transaction is adequately protected.

Transaction metadata allows administrators of the system to monitor transactions and gauge the health of the system. This is useful for discovering performance bottlenecks.

M - Mediation Component The mediation component executes transactions destined for the service providers within the HIE. Its main functions are orchestration and message translation. This component addresses a number of the architectural concerns. Firstly, it addresses concern #4 and #5 by enabling messages to be transformed at both inbound and outbound steps in a transaction channel. Secondly, it addresses concern #7 by allowing a multitude of transactions to be implemented and by enabling multiple standard messaging formats to be supported. Thirdly, it assists in addressing concern #2 by enabling a key set of supported transactions to be managed centrally as transaction channels.

The mediation component is made up of a number of transaction channels. A channel is provided for each transaction type. For example, a transaction type to save a patient's encounter. Messages received from the persistence component are routed to the appropriate transaction channel that can handle the transaction type of the current transaction. Each channel contains the necessary logic to normalise, orchestrate and de-normalise that transaction. Each function exposed by the API in the interface component maps to a transaction type and therefore to a transaction channel.

Below we describe the process that occurs within a single transaction channel contained within the mediation component.

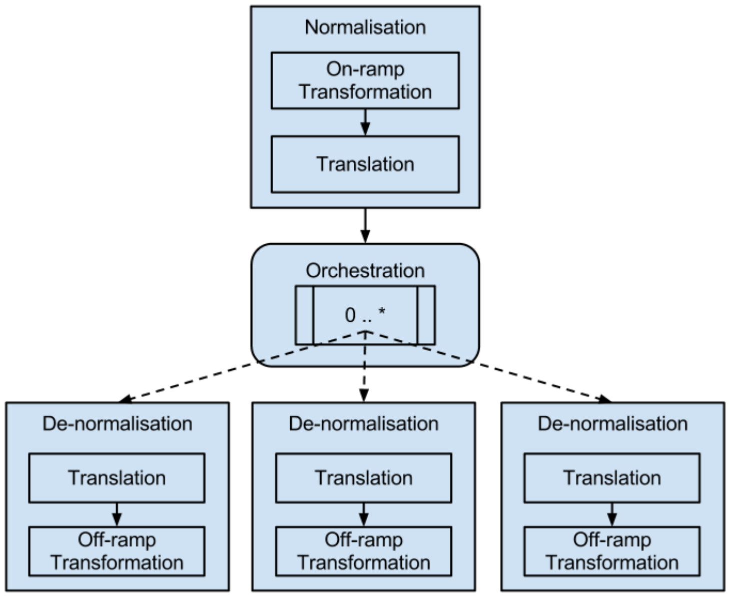


Figure 4.4: The workflow of a transaction channel within the transaction mediation component

Figure 4.4 on page 62 shows the inner workings of the transaction mediation component described earlier. Each transaction type has its own transaction channel. The diagram represents the workflow within a single transaction channel.

A transaction channel always begins with a normalisation sub-component. This sub-component transforms the request message contained within a transaction to a normalised state. This normalised state is called the canonical form for that transaction. After this process the transaction data must be in a consistent and predictable format to allow components following this step to process it in a predictable fashion, no matter what format it arrived in. This process consists of two operations. Firstly, an on-ramp transformation is applied. This ensures that the message is transformed into a form that the HIM can process, thus enabling syntactic interoperability for the transaction. For example, if the transaction arrives from a legacy application that only supported exporting data in a custom XML format, this process would ensure that the XML is transformed into the canonical form that the HIM can understand, such as an HL7 version 2 message. Secondly, a translation operation is invoked. This operation is responsible for ensuring the codes and code systems used within the transaction are translated to a standard set of vocabulary or clinical terms, called reference terms, that have a common interpretation by other components of the HIM. This could involve a call to a terminology service to translate and verify that the codes used within the transaction are represented in, or are translated to, known reference terms. In this way semantic interoperability between service requesters and providers is achieved.

Following this, the transaction is sent to the orchestration sub-component. This sub-component is responsible for performing implementation-specific orchestration for the current transaction. The process of orchestration is described in Peltz *et al.* [56]. The aim of the orchestration component is to execute the received transaction and perform any consequent action(s) required for this transaction. This could include zero or more calls to external services as well as the execution of business logic. This component compiles the response for the executed transaction and returns this to the persistence component which forwards the response to the service requester via the interface component. The calls to external systems should be done in parallel where possible to ensure that the orchestration is done quickly and efficiently as possible.

A de-normalisation sub-component is provided for each external service call. This sub-component is responsible for transforming or constructing a service request in a format that is understandable to the service provider. This operates in a similar way to the normalisation component except the operations occur in the reverse order. This approach serves to decouple service providers from the orchestration component, which allows for service providers to be easily modified or replaced with minimal impact on the mediation component.

The transaction channel for a specific transaction can be as complex or as simple as a transaction requires. For example, consider a simple transaction: the message arrives in a known form and no orchestration is required, the message just needs to be sent on to a service provider system in the same format as it arrived. In this case the transaction channel would be very simple; no normalisation would need to occur, no orchestration is required and no de-normalisation needs to occur. Thus, the transaction channel for this transaction just passes the message on to the service provider. Now, consider a more complex example: a message arrives in a form that requires normalisation, we need to enrich the message with some additional information before it can be sent to the service provider and the service provider uses a custom JSON message that the original message must be denormalised into. In this case, as the message arrives the normalisation sub-component executes. This converts the message into a syntactic structure that the other HIM components can understand, as well as translates any codes that are used in the message to reference terms that are understood by the orchestration component. Now the message is in a standard form so it is passed onto the orchestration sub-component. This sub-component reads the message and executes logic in order to enrich the message. In this example, this logic requires the use of an external service so a de-normalisation sub-component is executed to generate a message in a form that the service provider can understand. When the response is received the orchestration component is able to enrich the original message with the information required. Then, the enriched message is passed to a de-normalisation sub-component so it can be sent to the intended recipient. This de-normalisation component executes a translation function to convert reference terms into codes that the service provider can understand and then executes a function to transform the message into the JSON format. This message is then sent to the service provider. As can be seen, these transaction channels can be as complex or simple as is required for the transaction. This allows the HIM to contain complex processing logic such that the service consumer and service provider systems may remain as simple as possible.

The transaction channels provide a generic structure so that any needed transactions may be implemented as required. Both the messaging format and the logic of each transaction channel is left up to the implementing party. This allows the HIM to remain agnostic of message format and transaction type so that it is applicable in a variety of settings and does not impose specific message formats and transactions on implementers of the architecture. These transaction channels are also designed to be modular so that the implementation of these may be shared and re-used. Each transaction channel is independent and should aim to perform a single task efficiently. This allows different transaction channels to be added or removed as required for a specific implementation. The use of normalisation and denormalisation components enable syntactic and semantic interoperability as syntactic and semantic heterogeneity between systems can be mitigated.

4.3.3.2 Scalability view

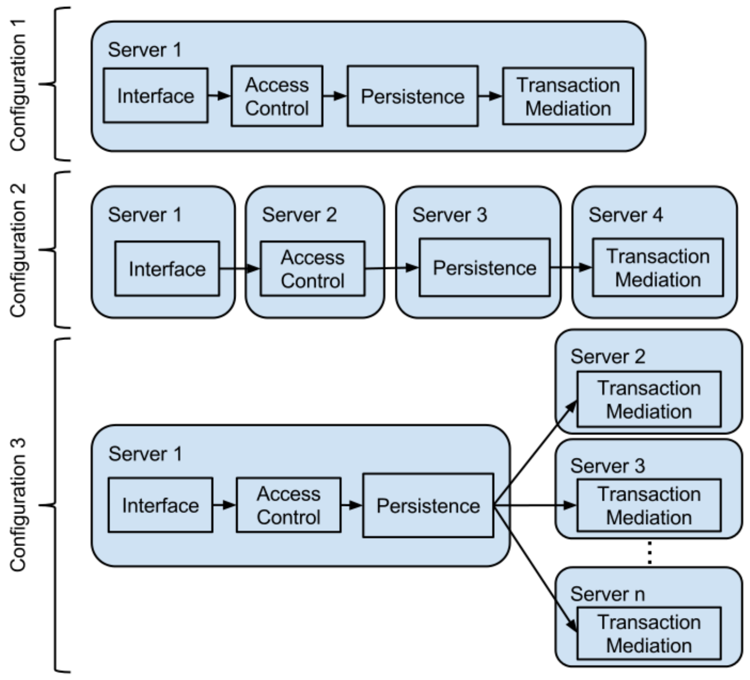


Figure 4.5: Scalability configurations of the HIM architecture

Figure 4.5 on page 65 shows the scalability of the architecture. This view described how architectural concerns #3 is addressed. There are four major components: the interface API, the access control component, the persistence component and the mediation component. Each of these components are loosely coupled so that they can be deployed across different servers. This is shown in “Configuration 2” of Figure 4.5 on page 65. The four components are responsible for separate, independent units of work. This loose coupling allows the components to be spread over different hardware as long as they are able to communicate over a network. The ESB architectural model used for this architecture ensures that the components are loosely coupled and can be deployed in a distributed fashion. Each of the components of the HIM architecture must also be implemented in a stateless manner. There is no need to share state or session between the components as they are each responsible for an independent unit of work. This enables each of the components or even the entire HIM server to be deployed redundantly if required. Thus, enabling greater options for scalability.

It is also feasible to further separate the persistence component and the transaction mediation component through clustering. The persistence component performs the static function of persisting any transaction that passes through it. As this function is not dynamic it could easily be replicated over multiple servers in a cluster with the provision that the data stores are synchronised. Many database implementations support clustering to improve performance and this may be used in order to improve performance. This component could also be invoked in an asynchronous fashion as the mediation component subsequent to it does not require this process to be completed in order to continue its processing.

The transaction mediation component can be scaled horizontally. This component holds a set of channels, one for each transaction type that is supported by the implementation. Each of these channels encapsulates information about how each transaction should be transformed and orchestrated and each of these channels are modular. This allows each transaction channel to run independently which enables them to be deployed across multiple, separate servers. This is shown in “Configuration 3” in Figure 4.5 on page 65.

These configurations show two important aspects of the architecture. Firstly, the HIM’s performance in terms of the volume of transactions that it can handle, i.e. splitting the load between different servers increases the capability of the system to handle and process a higher volume of transactions timeously. Additional servers can be introduced as transaction volumes grow to stabilise performance. Secondly, it demonstrates the HIM’s robustness. Since each of the three components are responsible for separate units of work and individual components can be replicated over different physical machines, this architecture is able to provide redundancy. The number of instances of each component can vary depending on the performance required of the defined transaction types and the reliability requirements for a particular implementation.

4.3.3.3 Modifiability view

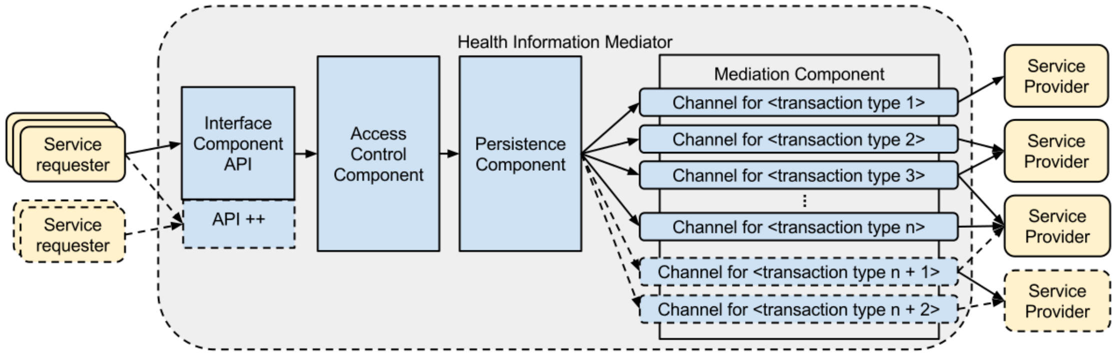

Health Information Mediator
Service requester
Service requester
Interface Component API
API ++
Access Control Component
Persistence Component
Mediation Component
Channel for <transaction type 1>
Channel for <transaction type 2>
Channel for <transaction type 3>
Channel for <transaction type n>
Channel for <transaction type n + 1>
Channel for <transaction type n + 2>
Service Provider
Service Provider
Service Provider
Service Provider

Figure 4.6: Extensibility of the HIM architecture

Modifiability is an important consideration for this architecture. Figure 4.6 on page 67 shows how additional services could be added to the architecture. This view described how concerns #3 and #7 are addressed by enabling new transactions channels to be added over time.

To add additional services the interface component's API needs to be extended by adding new API endpoints for each new transaction that needs to be supported. The persistence component and the access control component are generic enough that they do not require any change to process new types of service requests. The transaction's mediation component is where most of the changes are required. This component is designed to encapsulate transaction mediation logic for each transaction type. A new transaction channel can easily be added to support a new type of service request. The new channel will encapsulate all the logic for normalising the transaction, executing the necessary orchestration steps and de-normalising the transaction when an external service orchestration call is made. This encapsulation simplifies the addition of new service request types as functionality increases and the HIE expands. This enables the HIM architecture to adapt to the new requirements of an HIE environment.

In this chapter, the key concerns of an architecture that facilitates interoperability have been extracted for LMICs. A combination of previous research and a real world use case drove the identification of these concerns. This architecture description describes three distinct architectural views. Each of these views are described in detail to produce a concrete description of the HIM architecture.

Chapter 5

Implementation of the HIM Architecture in Rwanda

The architecture was validated by a reference implementation, the OpenHIM, within the Rwanda Health Information Exchange (RHIE). In this chapter we describe the design and implementation of the OpenHIM for the RHIE. The larger design decisions are discussed first, then the implementation technologies and details are described in the sections that follow.

5.1 The point of care interface

The interface component of the HIM architecture is responsible for exposing an API for the point of care systems to use in order to interact with the HIE. An important design decision is thus how this API should be exposed. An appropriate technical interoperability standard must be chosen. The API needs to be simple to interact with and be platform independent. Web services are a suitable choice for this API as they are platform and technology independent and are widely supported. Web services utilise the HTTP protocol in order to transmit information. There are two competing paradigms for web services. namely, Simple Object Access Protocol (SOAP) [6] and RESTful web services [24]. These two paradigms are explained in more details in Section 2.3.1.

A RESTful web service approach was chosen as the technical interoperability standard for this implementation as it is the simplest approach for point of care systems to implement. REST does not require an XML envelope to transport the message, contrary to SOAP. REST maps actions and message data directly to constructs of the protocol on which it is being implemented (e.g. HTTP). Using HTTP these services become very easy to call as most languages provide functionality to communicate using HTTP. Thus, we opted to make use of REST over HTTP for technical interoperability. SOAP has a good array of extensions to handle security and other messaging concerns. However, while these are comprehensive solutions they make SOAP complex to process and thus additional software libraries are required to work with SOAP messages.

5.2 Messaging format

As REST does not prescribe a format that the data should be in (it is merely an architectural style) any data format can be used. Using REST over HTTP allows us to use the HTTP concept of content types. HTTP content types allow us to specify the format of the information that we are sending.

Not all information can be encoded in a single data format. Information should be expressed in the data format that makes the most sense. For example, a patient's medical record information would most likely be represented differently from a health facility's monthly report of aggregate indicators.

By defining different content types, information can be supplied in a variety of different formats. Multiple data formats for the representation of the same information could even be offered. For example, allowing patient information to be returned in either XML or JSON. This would allow the client to request information in a format that is easiest for them to process. Internally, the HIM architecture could handle these varying formats using its normalisation and de-normalisation functions.

This being said we still had to choose a canonical format to use within the RHIE. There are three main contenders for messaging formats for clinical information. OpenEHR Archetypes, HL7 version 2 and HL7 version 3. HL7 version 2 was chosen as it is an established message format within the health domain, enabling us to make use of the large variety of tools that support HL7 version 2. HL7 version 3 is much more complicated to handle and process than HL7 version 2 and it does not have as much tool support. The size of HL7 version 3 messages have also grown significantly and this makes it unsuitable for low resource settings where connectivity is poor and there are few developers who understand the complexities of HL7 version 3. Thus, HL7 version 2 was deemed to be the appropriate choice for the RHIE.

The use of HL7 version 2 gives us syntactic interoperability by defining the specific message formats for different types of transactions, such as saving a patient's record or communicating a patient's encounter data. However, it does not enable semantic interoperability. For semantic interoperability the project team elected to use a combination of LOINC codes, ICD-10 codes and custom Rwanda specific codes for information for which mappings could not be found.

IHE profiles could have also been used to provide semantic interoperability for some of the RHIE transaction, however, the project team was not familiar with IHE profiles and the benefits that they provide. Thus, other standards that the team was more familiar with were chosen. In future work, the RHIE team is looking to migrate some of the supported transactions to make use of IHE profiles.

5.3 Mule ESB

Mule ESB is a widely used platform for enabling application integration and interoperability. Mule ESB is a lightweight Java-based enterprise service bus (ESB) and integration platform[1]. It is an open source product with optional commercial offerings available. It is designed to be highly scalable and lightweight. The main functions that Mule ESB provides are the ability to host and create connections to services and to provide service mediation, message routing and data transformation functions. Mule ESB as a platform runs as a server and allows the user to host Mule applications that implement the desired functionality for their implementation.

Mule ESB was chosen as the base platform on which to build the HIM implementation due to the fact that it provides many of the functions that we require. It also has a large user base and support community. Additionally, the open source edition of Mule ESB enables it to be used freely in low resource settings with no licensing requirements.

Mule ESB is built around the notion of Mule flows. A Mule flow consists of a series of message processors that are executed in order as the message passes through the flow. The flow contains logic and flow control operators that are able to process a message, as well as communicate that message to other external endpoints. Mule flows are the base structure

[1]See `http://www.mulesoft.org/what-mule-esb`

that Mule uses to enable users to implement an ESB. A flow typically has a single inbound endpoint that receives messages and one or more message processors that process messages and outbound endpoints that send messages to an external service.

5.4 System Architecture of the OpenHIM

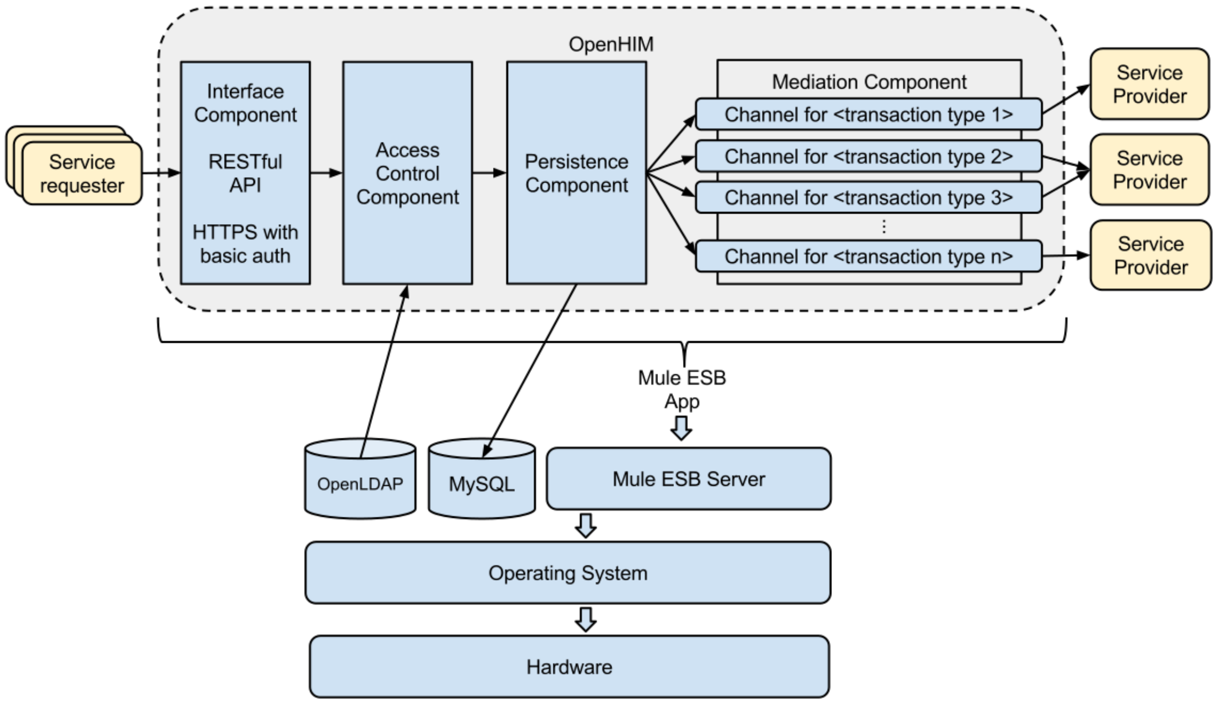


Figure 5.1: Systems architecture of the OpenHIM

The Open Health Information Mediator (OpenHIM) was implemented as a reference application of the HIM architecture. It was developed as an open source project in collaboration with Jembi Health Systems NPC. The project code is hosted on Github[2] and is currently still maintained and used by Jembi Health Systems NPC. It also now forms part of the reference applications for the OpenHIE project[3].

The OpenHIM is implemented as a single Mule ESB application with a number of different components that represent the components described in the HIM architecture. These components are invoked in order as shown in Figure 5.1 on page 72 and as described in Section 4.3.3.1. Figure 5.1 on page 72 also shows the technical architecture of the OpenHIM

[2]The source code can be found at `https://github.com/jembi/openhim`
[3]For more information about this project see `http://ohie.org/`

application. In the sections that follow the implementation of each of these components is described in more detail.

5.4.1 Implementation of the point of care interface and access control components

The interface and the access control components are implemented as a single Mule flow that exposes an HTTP endpoint. This endpoint is the entry point for any service calls into the HIE's RESTful API. This endpoint is secured using HTTPS with a server side certificate and basic authentication. Within this flow the user accounts for the basic authentication are handled by a separate LDAP (Lightweight Directory Access Protocol) server, specifically OpenLDAP. The Mule flow is setup with an HTTPS endpoint that authenticates requests using the OpenLDAP database. This can be seen in Figure 5.1 on page 72.

Each application that needs to interact with the HIE requires a separate username and password in order to authenticate itself. These usernames and passwords are stored in the LDAP database. This makes it easier to manage which applications are allowed to access HIE services. This model allows applications to be authorised to call only certain services in accordance with a specific role assigned to that application's user account. The structure for this role-based restriction functionality exists but has not yet been implemented.

Service calls enter the interface component, get authenticated and authorised and are then passed to a queue in order to be processed by the persistence component.

5.4.2 Implementation of the persistence component

The persistence component is also implemented as a single Mule flow. This flow accepts messages from an input queue and writes the raw message, as well as metadata (such as when the message was received and which service consumer sent the message), to a MySQL database. When a message response is returned to the service consumer this component also stores the response messages, as well as metadata about the response including the time the response was received and any errors that may have occurred. These are all stored in the MySQL database to create an audit trail of messages that have been processed by the HIE as well as to log errors that may occur within the HIE. The data model of the OpenHIM persistence component can be see in Figure 5.2 on page 74.

Each message is assigned a status to track if that message was processed correctly. There are three processing states: processing, completed and error. This allows transactions to be tracked and monitored within the HIE.

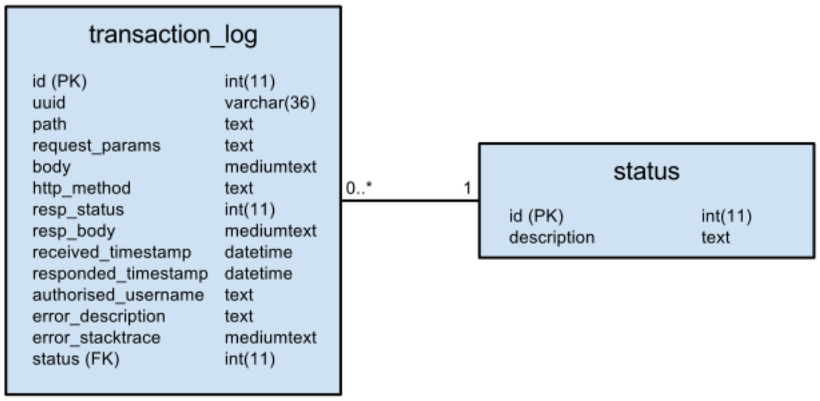


Figure 5.2: OpenHIM persistence component data model

5.4.3 Implementation of the mediation component

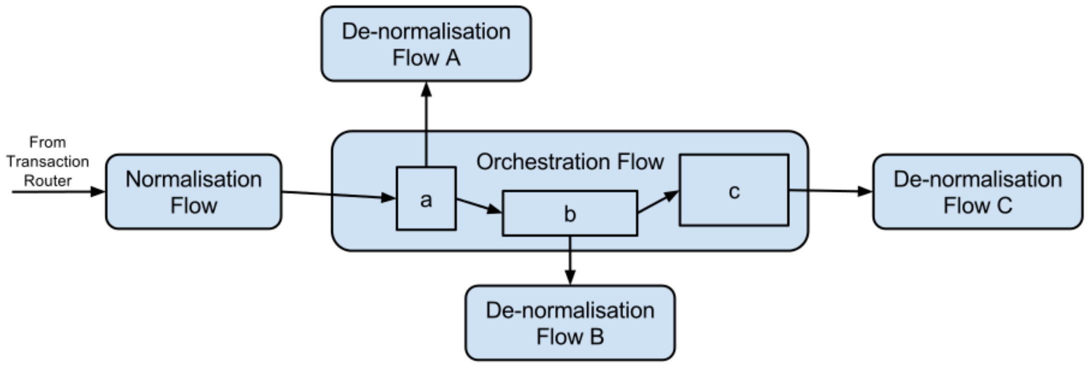


Figure 5.3: The structure of a sample mediation component

In order for messages to be sent to the correct transaction channel they must be sent to a transaction router. The transaction router is a Mule flow that fits between the persistence component and the mediation component. It accepts all messages and determines what transaction type is being invoked by the message and sends that message to the corresponding transaction channel to be processed. This is done by inspecting the URL path and HTTP method of the message and looking up what transaction channel can handle the request.

The mediation component consists of a number of Mule flows that each have different responsibilities. Flows are designed to either normalise, de-normalise or orchestrate a specific transaction. For each transaction channel within the mediation component there is exactly one Mule flow for each normalisation and orchestration. For de-normalisation there may be zero to many Mule flows in order to handle communication with external services needed by the orchestration flow. Each transaction channel maps to exactly one service that is exposed by the interface component. Figure 5.3 on page 74 shows a sample of a mediation component depicted as a component diagram. Each block is a separate Mule flow. The different types of flows shown in this diagram are explained below.

Normalisation flow The normalisation flow for each mediation component contains logic that can transform a transaction into a canonical form that is understood by the succeeding components in the pipeline. It is always the first flow that a message is passed to in a transaction channel. It contains two functions in order to perform transaction transformations. Firstly, an on-ramp transformation function that transforms the message structure into a standardised canonical form and, secondly, a translation function that translates the message semantics into a standardised form. The implementation of each of these functions is implementation specific and dependent on the transaction type and the formats that a specific transaction may be received in. These can be implemented as Java code or using a Mule XSLT[4] transformer depending on the needs of the implementation, as well as the chosen canonical message format for that implementation.

Orchestration flow The orchestration flow for each mediation component contains the orchestration logic to execute transactions. This logic is dependent on the transaction type that is being executed and is implemented using a suitable construct depending on the specific implementation. These flows expect that the messages that they receive are already in a standardised canonical form. The logic in this flow is expected to be able

[4]XSTL (Extensible Stylesheet Language Transformations) is a language that allows an XML document to be transformed into a new XML document with a different structure.

to make external service calls to other service providers within the HIE. For example, to allow identifiers to be checked for validity or to save an encounter in the shared health record. Many of these external calls can be made and de-normalisation flows are invoked to perform the actual sending of a message to an external service provider. Mule ESB's provided enterprise integration pattern implementations are used to implement the required orchestration logic.

De-normalisation flows Each of the de-normalisation flows map to an external service call that is referenced from the orchestration component. Each of these flows contain the logic in order to convert the message from the standardised canonical form into a form that the infrastructure service can understand. This logic is dependent on the external service that is being contacted. Mule outbound endpoints are used within these flows in order to physically send the message to the infrastructure service's endpoint.

5.5 Implementation of the Rwandan HIE

The OpenHIM was successfully deployed with the other HIE components in Rwanda during September 2012. The current system connects fifteen health facilities in the Rwamagana district to the HIE deployed in the national data centre in Kigali [5].

The infrastructure services that form the rest of the Rwandan HIE were implemented by different parties utilising a wide variety of open source projects, which are listed below:

- Shared Health Record: OpenMRS (OpenMRS Foundation, Regenstrief Institute and Partners in Health)[6] [43] with modules developed by Jembi Health Systems NPC [7].
- Client Registry: OpenEMPI (SYSNET International)[8].
- Provider Registry: a custom open source webapp built on OpenLDAP (Intrahealth)[9].
- Facility Registry: ResourceMapper (InSTEDD)[10].

[5]See the implementation blog at `http://rwandahie.blogspot.com/2012/09/click.html`
[6]`http://openmrs.org/`
[7]`https://github.com/jembi/rhea-shr-adapter`
[8]`http://www.openempi.org/`
[9]`http://www.ihris.org/wiki/Provider_Registry`
[10]`http://resourcemap.instedd.org/`

- Terminology service: Apelon DTS (Apelon Inc.) and a custom webapp front end (Jembi Health Systems NPC)[11].

In the next section a description of the workflow that allows clinics in Rwanda to share information using the OpenHIM is described in detail.

5.5.1 The RHIE workflow

The OpenHIM exposes the services required for sharing clinical information between the clinics in Rwanda. It provides a variety of services that enable the sharing of clinical information between clinics. These include services to:

- Register a patient's demographic information.
- Query for a specific patient or for a list of patients.
- Update a patient's demographic information.
- Save a patient's encounters to a central clinical data repository.
- Query for a list of a patient's previous encounters from the central clinical data repository.

When a patient arrives for maternal care at the clinic the registration clerk first looks them up on the local OpenMRS system. If they cannot be found on the local system a query can be made to the client registry if the clinic has connectivity. This workflow can be seen in Figure 5.4 on page 78. The numbers in brackets in the following descriptions refer to specific interaction in the sequence diagrams.

The OpenMRS system makes a web service call to the OpenHIM to query for a patient matching the criteria that the registration clerk entered (1). The OpenHIM accepts this message if the request is correctly authorised and authenticated. It then stores the message and passes it on for mediation. In this case the message is de-normalised to a form that the client registry can understand (2) and no normalisation or orchestration functions are

[11]`http://www.apelondts.org/` and `https://github.com/jembi/ts-browser`

required. The OpenHIM makes the call to the Client Registry (3), normalises the response into a form that it can understand (5) and returns a list of one or more patients to the OpenMRS system (6). OpenMRS displays the choices to the user to verify which record matches the patient. If the patient cannot be found in the Client Registry, the registration clerk enters the patient demographic information and this is then sent as a message to the Client Registry through the OpenHIM to register that patient. This workflow is shown in Figure 5.5 on page 78.

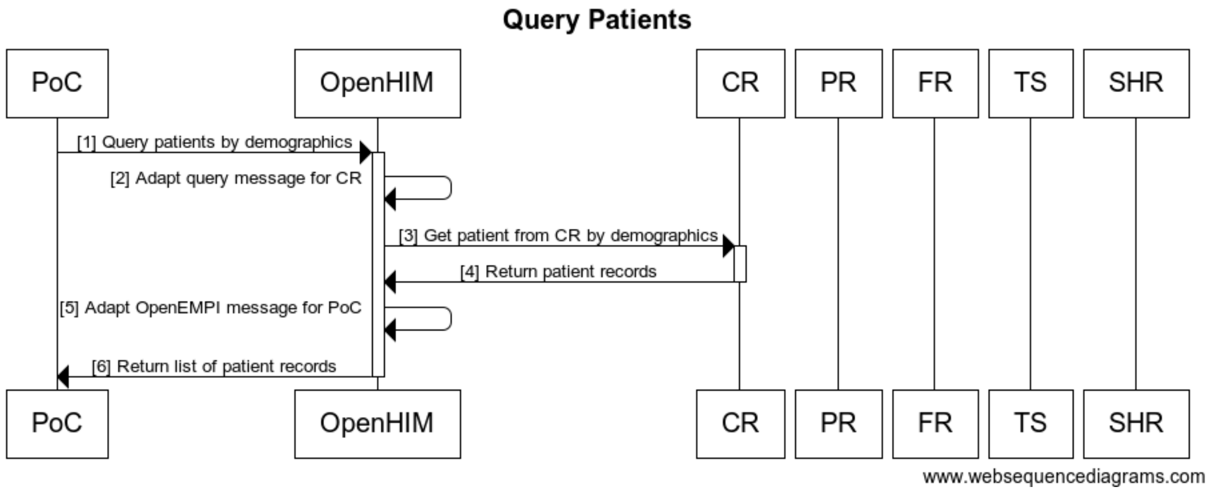

Query Patients
PoC
OpenHIM
CR
PR
FR
TS
SHR
[1] Query patients by demographics
[2] Adapt query message for CR
[3] Get patient from CR by demographics
[4] Return patient records
[5] Adapt OpenEMPI message for PoC
[6] Return list of patient records
www.websequencediagrams.com

Figure 5.4: Query patients

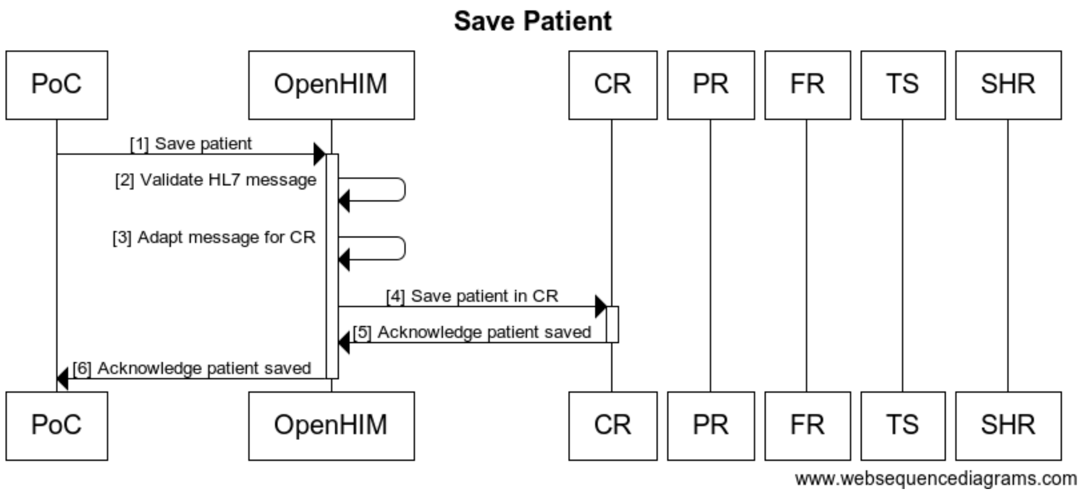


Figure 5.5: Register a patient

Once the patient is registered on the local system they are ready to be seen by a clinician. Figure 5.6 on page 80 show this workflow. The clinician looks them up in OpenMRS and is able to fill in forms capturing observations about the patient's visit. Each form that is saved triggers the OpenMRS system to invoke the save encounter transaction. This makes a call to the OpenHIM with a message containing the clinical encounter and all the observations that the clinician entered in HL7 v2 format (1). The OpenHIM receives this message and stores it. It is then passed on to the mediation function. This save encounter transaction has a number of orchestration steps that the OpenHIM must perform. The message is validated against the client registry, provider registry and facility registry to ensure that the identifiers used for the patient, provider and location are known in each of those registries (2 - 10). The message is also enriched with the patient's enterprise identifier that the client registry uses to uniquely identify patients as well as the provider's enterprise identifier that the provider registry uses to uniquely identify a provider within the HIE. Once that is completed, each of the codes (from code systems such as LOINC or ICD-10) used within the message are checked against the terminology server to ensure that they are known and valid (11 - 13). If this orchestration occurs successfully, only then is the message sent on to the shared health record system where it is stored in the patient's shared health record or in a new record if one does not already exist (13). A response is sent to OpenMRS so that it can determine if the transaction was successful or not (15).

When the patient arrives at another clinic their record can be retrieved and stored in the OpenMRS system at that clinic. Figure 5.7 on page 81 shows this workflow. The patient must already be registered in the system as described above. The OpenMRS system makes a call to the OpenHIM to query for previous encounters for that patient (1). The OpenHIM performs some orchestration steps to resolve the patient's identifier within the HIE (2 - 4). The OpenHIM fetches the patient's health record from the Shared Health Record service provider (5, 6). It then performs orchestration on the response to enrich the message with the patient's identifier and provider's identifier that are known by the local OpenMRS system (7 - 10). These are obtained from the Client Registry and the Provider Registry respectively. The enriched message is returned to the OpenMRS system at the clinic where it is stored locally in the system so that it may be viewed by that clinic's clinicians (11).

Using these workflows, clinics within Rwandan are able to share patient information effectively. The OpenHIM is used primarily to simplify these interactions by authenticating and

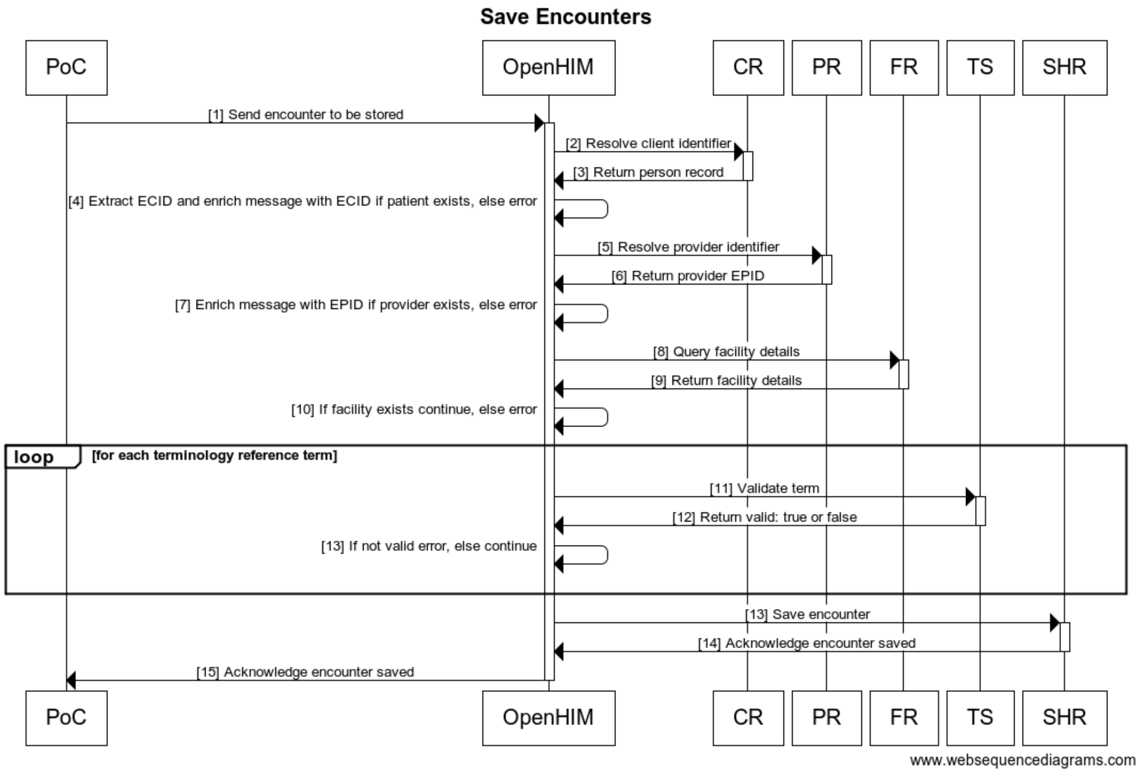

Save Encounters
PoC
OpenHIM
CR
PR
FR
TS
SHR
[1] Send encounter to be stored
[2] Resolve client identifier
[3] Return person record
[4] Extract ECID and enrich message with ECID if patient exists, else error
[5] Resolve provider identifier
[6] Return provider EPID
[7] Enrich message with EPID if provider exists, else error
[8] Query facility details
[9] Return facility details
[10] If facility exists continue, else error
loop
[for each terminology reference term]
[11] Validate term
[12] Return valid: true or false
[13] If not valid error, else continue
[13] Save encounter
[14] Acknowledge encounter saved
[15] Acknowledge encounter saved
www.websequencediagrams.com

Figure 5.6: Save an encounter

authorising requests and by performing the various orchestration and message transformation tasks that are required.

The RHIE implementation has now been operational for more than two years and has enabled clinical data to be shared between fourteen health facilities and the district hospital in the Rwamagana district. This implementation has enabled us to analyse the correctness of the architectural concerns and the extent to which they have been met. It also provides the first significant validation that the HIM architecture, and by extension an ESB-based architecture, can facilitate interoperability between HISs in LMICs.

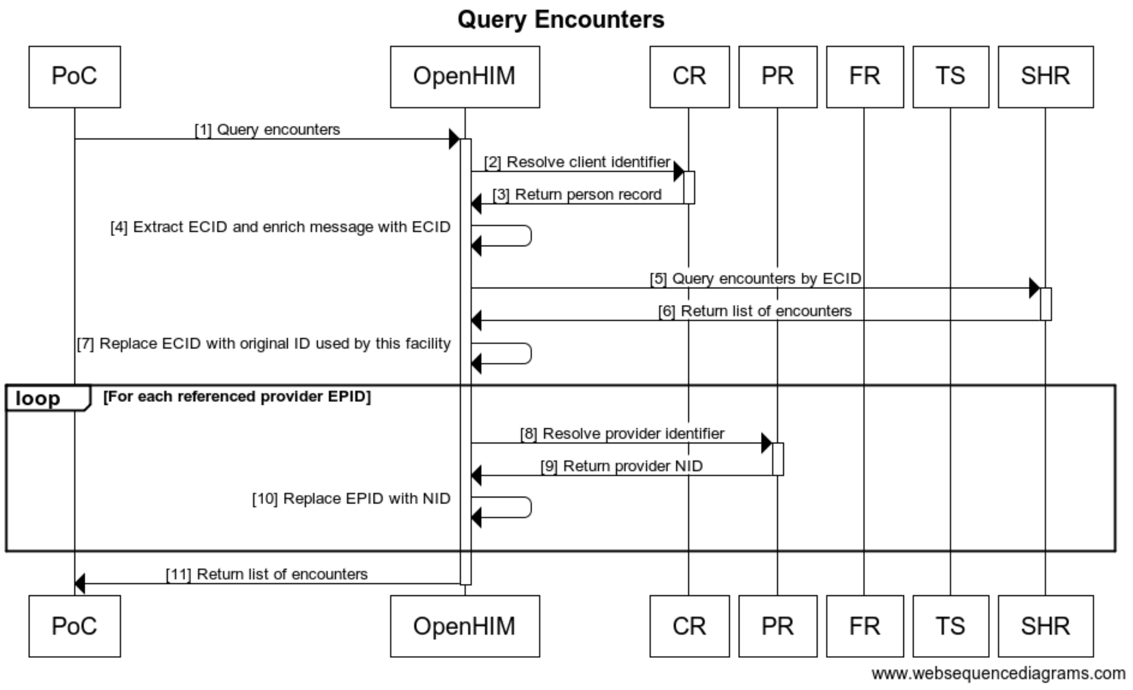

Query Encounters
PoC
OpenHIM
CR
PR
FR
TS
SHR
[1] Query encounters
[2] Resolve client identifier
[3] Return person record
[4] Extract ECID and enrich message with ECID
[5] Query encounters by ECID
[6] Return list of encounters
[7] Replace ECID with original ID used by this facility
loop
[For each referenced provider EPID]
[8] Resolve provider identifier
[9] Return provider NID
[10] Replace EPID with NID
[11] Return list of encounters
www.websequencediagrams.com

Figure 5.7: Query an encounter

Chapter 6

Analysis and discussion

The analysis and evaluation of software architectures is a challenging task [41]. In this chapter, an analysis of the extent to which the architecture satisfies the identified concerns is given first. The Rwandan Health Information Exchange implementation is used to provide concrete illustrations via the implementation of the OpenHIM. Secondly, the re-usability of the architecture is demonstrated by describing its implementation in another low resource setting, namely the the MomConnect implementation in South Africa. Finally, the Architecture Trade-off Analysis Method (ATAM) [41] is performed on the architecture. This analysis identifies the quality of the architecture in terms of quality attributes that are derived from the architectural concerns. The chapter concludes with a comparison between the HIM architecture and other architectures identified in the literature.

6.1 Analysis of the HIM architecture

In this section we analyse the extent to which the architectural concerns (identified in Section 4.2) are satisfied by the HIM architecture. For each concern a concrete scenario is described that shows how the architecture caters for a particular concern. Further, the RHIE deployment, which has been operational for two years, is used to demonstrate and validate that the HIM satisfies these concerns in a real world environment.

6.1.1 Concern #1: Facilitate interoperability

The architecture facilitates interoperability by providing a mechanism through which disparate HISs communicate. There are two aspects to this communication:

- The architecture provides a physical communication channel through which the disparate systems may communicate

- As the system are heterogeneous, syntactic message format and semantic message content can be modified to enable the systems to interoperate.

Consider two point of care systems (X1 and X2) that wish to communicate patient demographic information to a central server (Y). The physical part of the communication is provided by supplying an endpoint for system X1 and X2 to send data to (the interface component). The requests are authenticated and authorized by the access control component and are then routed to a mediator that is able to send the message to system Y. The architecture enables syntactic and semantic interoperability through mediator components. These components perform message transformation or complex orchestration using the normalisation and de-normalisation sub-components or the orchestration sub-components respectively. This allows for the systems to exchange messages even if they do not use the same message format.

In the RHIE OpenHIM reference implementation, HTTP endpoint are used to provide physical communication between HIS. The OpenHIM routes transactions between the point of care systems and the infrastructure services. A number of normalisation and de-normalisation components are implemented within mediators to enables heterogeneous systems to communicate more easily. The de-normalisation component transforms standard HL7v2 ADT messages (that the point-of-care systems are required to submit) to a custom XML format that is used by the client registries API. There are several transformations of this kind within the RHIE implementation.

In practise the physical communication provided by the OpenHIM allows systems to communicate without them having to know the details or location of the servers they communicate with. Syntactic and semantic interoperability is more tricky. The HIM architecture provides a framework to allow different syntactic and semantic message formats to be handled. However, it is left to the implementer to develop these normalisation and de-normalisation steps. It is currently not possible to do this automatically, however, this is an open problem and there is work under way to attempt to solve this problem [55].

6.1.1.1 Supporting syntactic interoperability

Syntactic interoperability standards allow information to be exchanged in a format and structure that both systems can interpret. There are many messaging standards available in the health domain for syntactic interoperability, each with different structures for representing data and servicing various messaging needs; for example, exchanging clinical information (HL7 v2, HL7 v3, OpenEHR Archetypes [11, 26, 20]) or aggregate health information for reporting (SDMX-HD [7]).

The HIM architecture allows the implementer to specify the use of different syntactic standards per transaction channel. The chosen syntactic standard will be used as the internal canonical form for processing messages within that transaction channel. Each transaction channel will have a canonical form that is specified for that particular implementation. The canonical form ensures that there is a single well defined format that is used within a transaction channel. In the future, this canonical form may be changed as new standards are adopted.

Support is provided for incorporating legacy systems that may already use standards that are different to the canonical format. It may not be easy or desirable to modify these systems. Thus, the HIM architecture prescribes normalisation and de-normalisation functions that enable messages to be converted into and out of the canonical message format and that may be as simple or complex as an implementation requires. They may just enhance a message to be more standardised or they may completely change the format and structure of the message to convert it to a different syntactic standard.

For example, the OpenHIM for the RHIE implementation was built to support HL7 version 2.6 for syntactic interoperability. All messages are converted into, and all orchestration occurs, using this canonical form. In certain cases, de-normalisation components transform these messages into a syntactic message format that other service providers can understand if they do not support HL7 version 2.6 messages. For example, the client registry (OpenEMPI) has its own custom RESTful API and a de-normalisation component of the OpenHIM converts the HL7 version 2.6 message into the format of this custom API whenever messages are exchanged with the client registry.

The use of a canonical form allows the internal orchestration steps to be simpler as they only have to support the canonical message format. The service consumers and service providers do not even have to support that particular format. This minimises the repetition of orchestration logic. However, the transformations from unsupported formats into the canonical form and vice versa has to be developed when implementing the HIM architecture. These transformations can be complex to develop and this can be a time consuming task [55]. Hosting these transformations centrally in the HIM implementation makes it easier to reuse this logic for point of care systems that require syntactic transformation of message formats.

6.1.1.2 Supporting semantic interoperability

Semantic interoperability standards allow information exchanged between heterogeneous systems to be interpreted and processed internally within a system's data structures. Semantic interoperability can be supported through the use of terminology services and particular mediation steps that allow code systems used in a message to be mapped to other code systems. A terminology service holds mappings between different code systems and is able to rapidly lookup these mappings in real time. Use of a terminology service allows messages to be modified so that they retain their original message semantics but the codes and code systems that are used to convey those semantics can be changed to those expected by other systems.

In the HIM architecture a terminology service can be called as a part of the mediation of a message. The HIM architecture specifies particular steps in the normalisation and denormalisation components that specify where message translation should take place. In this way semantic differences can be mitigated. However, it is up to the implementer of the HIM architecture to decide how this translation takes place. The terminology service could be as simple as a lookup table or it may be an external application dedicated to mapping and responding to queries about code systems. The HIM normalisation and de-normalisation components are responsible for resolving the message semantics and modifying the message so that it can be understood by intended recipients. In the case of the normalisation component, the message semantics must be resolved to the code systems expected in the canonical form so that orchestration can take place using well understood semantics. In the case of the de-normalisation components the message semantics must be resolved to the code systems that the service provider expects.

The OpenHIM reference implementation uses a terminology service to do basic validation of the terminology used in messages to ensure that semantic interoperability is achieved. The HIM architecture's normalisation components are used to perform validation of codes within the message to ensure that the messages contain terminology that is understood by the rest of the system.

New point of care systems that want to connect to the HIE have two options involving different levels of effort. If a legacy point of care system is difficult to extend but is already able to produce information in a legacy format it could simply be adapted to send this information to the HIM implementation and logic could be developed in the HIM to normalise this information into its canonical format so that it may be further processed. Alternatively, if the point of care system can be easily modified it may take less effort to extend the existing system to produce messages in the canonical format supported by the HIM implementation. Different data models can be mitigated through the use of the HIM architecture's orchestration functions. Information can be retrieved from other sources to fill in the gaps between what was expected and what the legacy systems were able to send.

The ability to embrace existing systems rather than ripping them out and replacing them is beneficial [14] especially in low resource settings [47]. The middle out approach to producing a national health information system involves defining interoperability frameworks nationally. Legacy systems are encouraged to extend their functionality to make use of the national infrastructure [14]. The HIM architecture supports this middle out approach by enabling legacy point of care systems to connect to national infrastructure even if these systems are unable to adapt to use the prescribed standard.

The support for multiple standards that embrace legacy systems and nuances in existing systems is of particular importance for LMICs, it ensures that existing investment in HIS is not abandoned. The architecture is standards agnostic and does not restrict an implementer to specific standards.

6.1.2 Concern #2: Balance central governance with local autonomy

The centralised nature of an interoperability layer allows access, configuration and monitoring of an HIE to be simplified by providing a single central facility where management occurs. This is ideal in a low resource setting as regional or facility level management of the HIE infrastructure can be challenging due to a lack of technical sophistication and infrastructure. Thus, it is better handled in a central location where dedicated resources are available to manage the HIE. The competing approach is a peer-to-peer model where HISs communicate directly with each other. This model works well in an environment where HISs at each health facility are sophisticated and can themselves organise health data from multiple sources, apply security policies and execute workflows to fetch and validate data from multiple other HISs. Thus, this approach is more applicable to HICs. Our goal is to simplify the interaction between HISs within the HIE. Thus, a centralised, middle-out approach as suggested in previous work is deemed appropriate.

In the RHIE, this approach was beneficial for a number of reasons. Firstly, access to health information could be controlled and managed centrally. This enabled the project co-ordinators and the Rwandan ministry of health to easily manage access and ensure that only designated systems, that are safe and secure, could have access to patient health information. Secondly, the transactions that the HIE supports could be managed centrally so that each point-of-care system could access a unified and appropriate set of transactions. Thirdly, this central approach enabled transactions to be audited for accountability and for transactions to be logged such that success and failure rates could be determined.

6.1.3 Concern #3: Adapt and scale

The HIM architecture was designed to adapt to new environments and modified for new functionality to be added over time. During the implementation of the OpenHIM transactions needed to be updated and new transactions needed to be created to support new interactions. For example, a new transaction was added to allow the client registry to notify the shared health record (SHR) that a patient record had either been linked or unlinked from another patient record. This would allow the SHR to link or split its own records in order to keep them synchronised with the client registry. The HIM architecture allowed these transactions to be added by enabling additional mediators to be deployed within the OpenHIM reference application. These were added dynamically with no effect on the existing transactions due to the separation and encapsulation of the mediator components within the architecture.

A limitation discovered was that a modification to a transaction within the HIE necessitated an update to the OpenHIM reference application and its mediators. This is because all communication between the point of care systems and the HIE takes place through the OpenHIM. These modifications require domain knowledge and experience with the OpenHIM software and requires that skilled developer resources are available to make these modifications. This could potentially cause delays in making additional functionality available.

The OpenHIM reference application deployed within Rwanda was able to scale within a single district of Rwanda. The deployment began by connecting two health facilities and scaled, over the course of two years, to cover fourteen health facilities as well as the district hospital. The transactional load over that time increased from a few hundred transactions per month to around 20,000 transactions per month. The OpenHIM was able to handle this load with ease.

To further test the scalability features, a preliminary empirical analysis of performance and scalability has been performed using the OpenHIM. There are two aspects of scalability that were considered in this analysis. Namely, vertical scalability (scaling up) and horizontal scalability (scaling out) [44]. Vertical scalability is the ability of the system to scale when additional system resources are allocated to the machine running the system. This is the simplest form of scalability. Horizontal scalability, on the other hand, is the ability of a system to scale across multiple computers in order to gain additional performance and reliability. Thus, the application must be able to be distributed across multiple servers to benefit from this scalability technique.

The following performance bounds were identified to evaluate scalability:

- Response times through the OpenHIM (ignoring other infrastructure services response time) should be < 500ms on average for each transaction type.

- Maximum response times should not exceed two seconds for any request.

To identify the load we expect from a national HIE in a low resource setting, we obtained visit statistics from the health facilities in the Rwamagana district of Rwanda. From this information we were able to extrapolate the load that would be placed on a nationally deployed OpenHIM implementation. Table 6.1 on page 117 shows these estimates.

The load estimates have been split into the relevant transactions that are executed through the OpenHIM. Each of the transactions has an estimate of the number of times it will be invoked per visit. Combining this with the total number of visits per month we get an average number of transactions per second (TPS) for each transaction type for an eight hour work day. This figure gives us a baseline to evaluate the performance of the HIM for different hardware specifications and gives us a baseline on which to generate test transactions using a realistic distribution of transaction types. The results of the performance analysis are described below.

6.1.3.1 HIM vertical scalability

To evaluate vertical scalability, three different hardware specifications were chosen and a performance analysis was performed on each of these systems. The first two hardware specifications were virtual machines running on Amazon's elastic cloud compute infrastructure. We made use of a m1.large and a m1.xlarge instance for these tests[1]. The third specification was a high powered laptop.

For each test case the number of concurrent users was increased until the results exceeded the performance indicators that are set out in Section 6.1.3. The points where maximum performance was achieved while staying within the performance bounds were recorded. The recorded results are shown in the following sections.

EC2 m1.large instance The m1.large EC2 instance could handle a maximum of five concurrent threads issuing continuous requests.

[1]See Amazon web services EC2 instance types: `http://aws.amazon.com/ec2/instance-types/`

Transaction	# Samples	Ave.	Min.	Max.	Std. Dev.	TPS
Save patient	63	137ms	100ms	297ms	43.22	0.73
Get patients	59	155ms	115ms	262ms	41.49	0.65
Update patient	48	164ms	122ms	315ms	50.18	0.56
Save encounters	137	560ms	391ms	1234ms	150.86	1.5
Get encounters	50	284ms	215ms	463ms	68.11	0.55
TOTAL	357	326ms	100ms	1234ms	214.95	3.8

Table 6.2: Performance results of an m1.large instance using 5 concurrent threads

EC2 m1.xlarge instance The m1.xlarge EC2 instance could handle a maximum of 10 concurrent threads issuing continuous requests.

Transaction	# Samples	Ave.	Min.	Max.	Std. Dev.	TPS
Save patient	82	102ms	74ms	250ms	28.95	0.88
Get patients	76	117ms	84ms	517ms	57.11	0.85
Update patient	68	138ms	92ms	384ms	59.43	0.72
Save encounters	259	464ms	340ms	1114ms	134.55	2.7
Get encounters	93	218ms	172ms	630ms	55.76	1
TOTAL	578	289ms	74ms	1114ms	188.97	5.9

Table 6.3: Performance results of an m1.xlarge instance using 10 concurrent threads

High specification machine

CPU:	Intel Core i7-2760QM @ 2.40GHz
Memory:	8GB
Hard-Drive:	250GB Solid State Drive
Platform:	Ubuntu Desktop 12.10 x86_64

The high specification machine could handle a maximum of 20 concurrent threads issuing continuous requests.

Transaction	# Samples	Ave.	Min.	Max.	Std. Dev.	TPS
Save patient	276	195ms	60ms	1209ms	156.19	2.9
Get patients	242	201ms	64ms	1236ms	151.41	2.5
Update patient	254	240ms	63ms	1311ms	198.94	2.7
Save encounters	764	564ms	178ms	1549ms	248.06	7.9
Get encounters	266	317ms	106ms	1352ms	185.17	2.8
TOTAL	1802	377ms	60ms	1549ms	266.15	18.3

Table 6.4: Performance results of a high specification machine using 20 concurrent threads

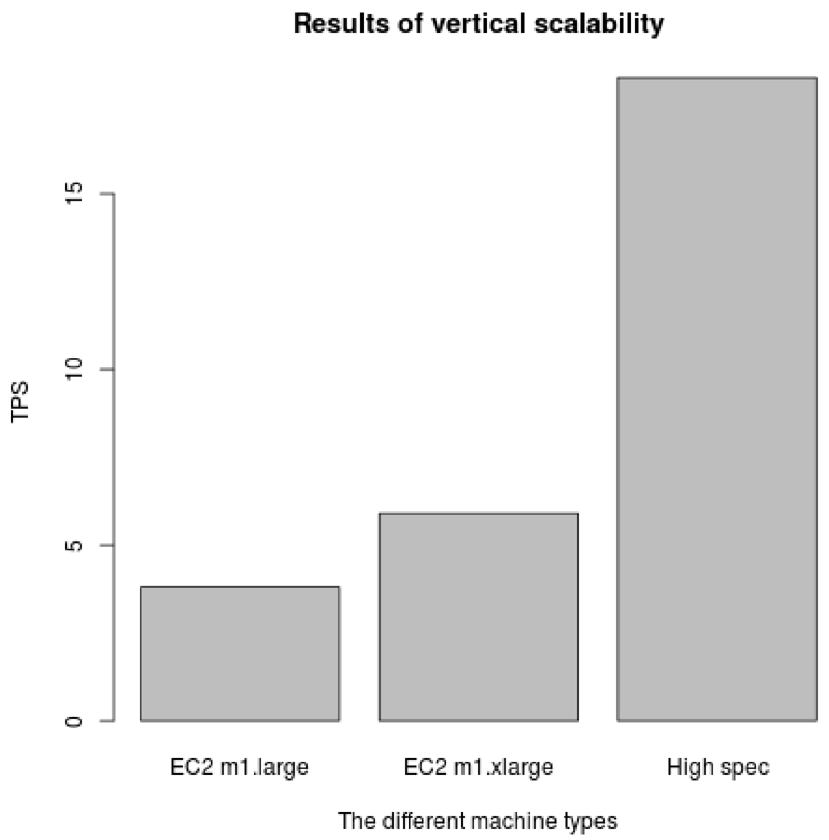


Figure 6.1: Results of vertical scalability

As can be seen, the HIM responded favourably to the increase in processor speed. On the two virtual environments we were able to almost double the number of concurrent threads by moving to a machine with double the processing power. On the high power physical machine we noticed that performance was drastically better. This is likely due to the powerful processor and the addition of a solid state drive for disk write performance.

The OpenHIM easily handles the estimated load for a single district on all of the tested hardware specifications. However, it falls short of the estimated peak national load of 36.61 TPS. The highest TPS achieved was 18.3 TPS. More powerful hardware could likely reach the target, however the cost to performance ratio increases greatly as our performance needs increase. A better approach is to spread the OpenHIM out over a number of smaller servers. This is discussed in the next section.

6.1.3.2 HIM horizontal scalability

The components of the HIM architecture are designed to be loosely coupled so that they can be spread over a number of servers. To demonstrate this, the reference application was distributed by running the most expensive/time consuming transaction mediation on a separate server. This layout can be seen in Figure 6.2 on page 92. In this case, when a save encounter transaction is encountered it is routed to a mediation component that is hosted on an independent server. All other mediation components are still run locally on the original server. The save encounter transaction is responsible for 47% of the estimated transaction load. The performance test suite was executed against the reference application to show how it reacts to being deployed in a distributed fashion. The results are shown in the next section.

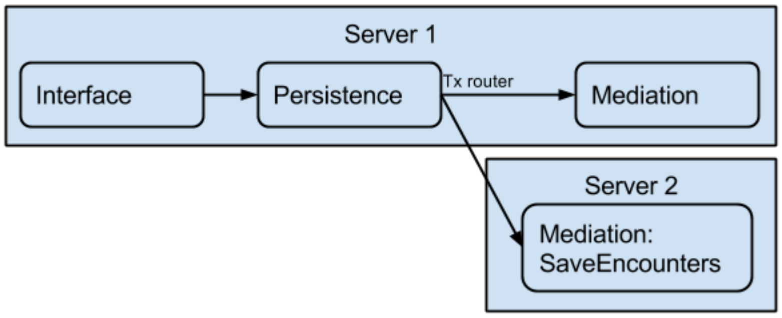


Figure 6.2: Sample distributed HIM layout

Transaction	# Samples	Ave.	Min.	Max.	Std. Dev.	TPS
Save patient	81	223ms	117ms	686ms	102.89	0.99
Get patients	68	243ms	130ms	852ms	115.24	0.85
Update patient	91	237ms	139ms	735ms	114.79	0.98
Save encounters	213	574ms	383ms	1304ms	169.63	2.2
Get encounters	61	364ms	233ms	650ms	112.52	0.74
TOTAL	514	390ms	117ms	1304ms	211.67	5.3

Table 6.5: Performance results using two distributed m1.large instances

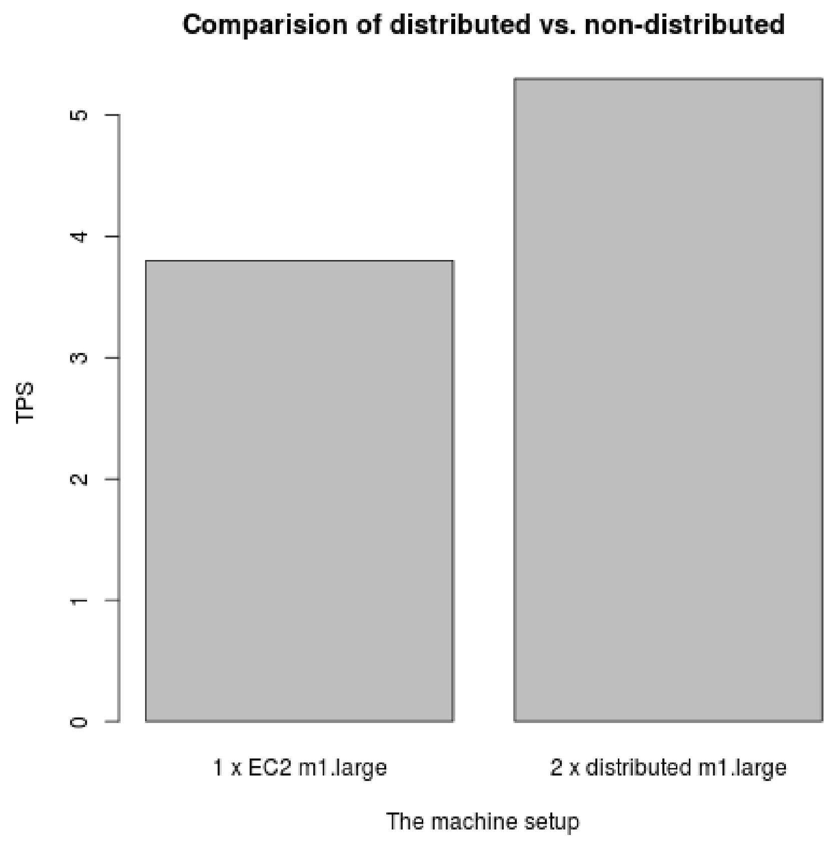


Figure 6.3: Comparison of distributed vs. non-distributed configurations of the OpenHIM.

2x m1.large instance As can be seen in Table 6.5 on page 93 distributing the application had a major impact on performance. The HIM was able to handle the same load as a single m1.xlarge instance from our previous tests. The distributed application was able to handle a load of ten concurrent users satisfactorily. It achieved a throughput of 5.3 TPS using two m1.large instances compared to the 5.9 TPS of the much more powerful m1.xlarge instance. This throughput is a 39.47% increase over the throughput of a single (non-distributed) m1.large instance from distributing the save encounter mediation.

The loose coupling of major components of the HIM allows us to identify bottlenecks in individual components and distribute them as needed. This is not restricted to just the mediation components but can apply to the interface and persistence components, as well as the sub-components of each transaction channel. Many different combination of components could be deployed in a distributed fashion depending on an implementation's needs. This is the ideal case for horizontal scalability. Many more servers can be added in order to reach the performance goals. This is possible due to the independence and encapsulation of each of the components. The only requirement being that the HIM implementation would need to make use of a suitable platform independent messaging standard for inter-component communication between components.

6.1.4 Concern #4: Prevent propagation of local changes

The configuration of an HIE will always require modifications as a country's needs change over time. This concern ensures that such changes are isolated and manageable. HISs that participate in the HIE should be able to adapt and evolve to suit their changing requirements and their connection to an HIE should not hinder them in this regard.

The HIM architecture allows for this by providing an abstraction layer through which all communication between the systems within an HIE takes place. Consider an infrastructure service that stores shared patient records in a central repository. This central data store exposes a standards based interface using the XDS.b profile specified by Integrating the Healthcare Enterprise (IHE). A number of point of care systems in hospitals and clinics use the XDS.b standard to feed data to the central repository via an implementation of the HIM architecture. After some time, some clinic systems want to upgrade their point of care systems to newer version that uses a profile of the newer FHIR based standard to transmit clinical information. Previously they would have to wait until the upgrade to using FHIR was coordinated between all the systems participating in the HIE as well as the central data store. However, the HIM architecture enables both standards to be used concurrently. A normalisation component can be added to the mediator that can still accept the older XDS.b standard and also transform the incoming FHIR based messages to the canonical form used by the mediator. The mediator will then still be able to communicate these messages to the central data store as per normal via its XDS.b interface. This allows the different systems operating within the HIE to localise their changes and thus be more responsive to user needs.

In the Rwandan OpenHIM implementation, infrastructure services within the RHIE were swapped out and updated over time. In particular a newer version of the client registry application, OpenEMPI, was deployed that changed aspects of its API. The OpenHIM mediator that performed transformation of messages for the client registry was all that needed to change to handle these API changes. The point of care systems that submitted patient demographic data could continue to submit data as before. The HIM architecture provided a mechanism to protect these system from this change.

However, there are some issues with this approach. It is not always easy to map one standards based messaging format to another unless they share an information model. This will not always be the case. The HIM architecture only provides the abstractions to make this flexibility possible but this flexibility depends on the particular standard being used. If any changes need to be made to the transactions that the HIE supports, the central component also needs to be changed. This mean that all other systems are not able to make use of these new transactions until the changes have been implemented within the HIM implementation. The HIM would likely be controlled by a government entity and the client systems are often controlled by a wide variety of organisations that can move faster than a government entity. Thus, problems could be encountered if the government entity is not responsive enough to change requests and innovation could be hampered.

6.1.5 Concern #5: Provide a low barrier to entry

A key requirement for an HIE is to enable many different types of HISs to be able to connect and share information with each other. For LMICs this is further complicated as the HISs systems can be much simpler and often do not have the budget of larger established HISs found in many developed countries. Thus, connection to the HIE must be as simple as possible.

The HIM architecture enables this by providing two mechanisms:

1. The HIM architecture is able to adapt legacy messages formats that older systems may use to those that the HIE expects. This is done using the normalisation component of mediators.

2. The HIM architecture centralises complex orchestration so that point off care systems do not have to reimplement these business processes. They may leverage the existing logic provided in a central location.

For example, consider a set of existing legacy point of care systems that need to connect to an existing HIE that utilised the HIM architecture. These systems are already able to exchange information using a custom legacy data exchange format. The HIM implementation could be used to connect these systems more easily by constructing a normalisation component within a mediator to transform the legacy exchange format into the current format that the HIM expects. In this case only one system, the HIM implementation, needs to be altered. For new point of care systems that are being developed, the expected data exchange format can be utilised and those systems can make use of the existing orchestration logic within the HIM implementation. Thus, their connection to the HIE is simplified.

Due to the orchestration logic for saving and querying encounters being encapsulated within an HIM mediator, the point of care systems are able to communicate with the HIE in a simple way. This allows multiple point of service applications to be integrated with the HIE easily using simpler service calls. In addition, if the point of care application couldn't capture the required scope of information to send in their messages to the HIE, the messages can be enriched within the encounter mediator. This allowed even the most basic point of care systems to connect to the HIE. An example of this is the RapidSMS system which was connected to the RHIE. RapidSMS allows community health workers to send coded SMS messages indicating the status of, or risks to a woman's pregnancy. These message carry little information due to the restriction on the size of an SMS message. The save encounter mediator within the RHIE is able to enrich the message with additional patient demographic information from the client registry before it gets saved in the clinical data store. In this way the interaction and the infrastructure services are significantly simplified.

6.1.6 Concern #6: Secure and auditable

The HIM architecture ensures that transaction are audited and that communications are secured. Audit logs are stored via the persistence components and the access control components controls authentication and authorisation within the HIE.

Within the RHIE, the security aspect was handled with HTTPS communication using "basic authentication" as the mechanism by which clients were authenticated and authorised. The OpenHIM reference application's implementation of the access control component reads client credentials from an LDAP database to ensure that only authorised clients may communicate with the HIE infrastructure.

6.2 Re-usability of the HIM architecture in other LMICs

The re-usability of the HIM architecture (concern #7) was tested by implementing it as part of a maternal mHealth initiative within South Africa[2]. The HIM architecture has also seen adoption in a few other HIE projects in South Africa.

6.2.1 MomConnect OpenHIM implementation

The South Africa National Department of Health along with a number of partner organisations (independent of this research) has recently implemented a program for mobile maternal health, called MomConnect. This program includes an HIE and national pregnancy registry (NPR) that forms part of a system that enables pregnant mothers to be registered and tracked throughout their pregnancy. The system is based on the Health Normative Standards Framework for Interoperability in eHealth in South Africa (HNSF), version 2.0 [48]. In addition, pregnant mothers are sent health promotion information about their pregnancy via SMS. The goal of the MomConnect program, including the NPR and HIE is to better inform mothers-to-be and to track their pregnancies, even in rural areas. The OpenHIM reference application was deployed in the implementation of this HIE. Figure 6.4 on page 99 shows the high level logical architecture of MomConnect which includes the following layers:

[2]See `http://www.jembi.org/project/national-pregnancy-registry/`

- Edge devices that include mobile phones and other computing devices that are responsible for collecting, collating and transmitting data to the consumer applications.
- Consumer applications that include mHealth applications and services providing value-added services to end-users. These could include electronic medical record (EMR) services.
- An HIE as a centralized platform and technical implementation of the Health Normative Standards Framework for Interoperability in eHealth in South Africa (HNSF) that is responsible for providing a single interoperability layer to receive and send messages in a well-specified, standard format between consumer applications and demographic and clinical repositories. The HIM architecture fits into this layer as it provides the platform on which interoperability can be achieved.
- Demographic and clinical repositories are centralized repositories of information and functionality, including the national pregnancy registry (NPR) that stores demographic details of pregnant women as well as client, provider and facility registries.
- Security/audit services include basic certificates and encryption to ensure the security of the messages being passed through the system.

The mobile health application, Vumi, is used in the Service Layer to capture the registration data from mobile phone handsets using USSD technology. Vumi then sends the registration data to the HIM architecture reference application, the OpenHIM, in a standardised format. This format consists of a CDA document, specifically defined for this use case and is transmitted using the Mobile Health Documents (MHD) profile from IHE. Once received by the OpenHIM, the data is orchestrated and sent to a number of infrastructure services for storage. The required orchestration was implemented using the mediator components as described in the HIM architecture. The orchestration includes validation and registration of client demographics in a client registry (OpenEMPI) and storage of clinical information in a shared health record (DHIS2 Tracker and OpenMRS), which acts as the NPR. Storage and reporting of registration data is done by an aggregate data collection tool, DHIS2. The HIE is also expected to allow multiple other mobile application vendors to connect. The HIM architecture allows this to be done simply as it enforces a single standardised interface

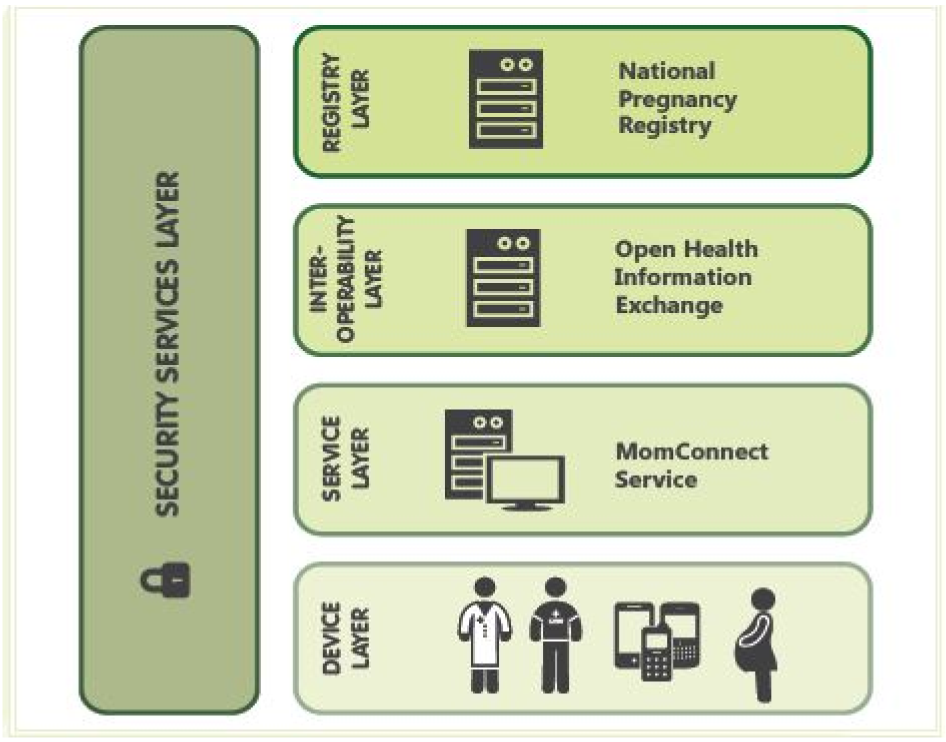


Figure 6.4: A simplified diagram of the MomConnect infrastructure (source: `http://www.jembi.org/project/national-pregnancy-registry/`)

for each point of care system to connect to and has configurable access control mechanisms to allow other point of care application access to the infrastructure as they come online.

The changes required of the OpenHIM reference application from the Rwandan HIE use case were minimal. The OpenHIM interface, access control and persistence components were used unchanged except for a small amount of re-configuration for the new environment. The only components that needed to be modified for this implementation were the implementation-specific mediation channels in the mediation component. These channels are designed to be easily swapped in and out as per the HIM architecture. It took a team of two software developers three months to design and develop the required mediators. These plugged directly into the OpenHIM's core component that housed the interface, access control and persistence components. In comparison, the original RHIE development and implementation (including the development of the OpenHIM tool) took numerous developers over a year to develop. Re-purposing the existing tool was much more efficient.

The requirements and challenges of interoperability between the disparate health information systems within this HIE were found to be very similar to those experienced in the Rwandan HIE. The robustness of the HIM architecture was clearly demonstrated by the fact that it could be readily applied to another environments with minimal effort. The major effort required is in developing the new mediation components and these are implementation specific so this will be required of any new implementation. This implementation serves as a validation of the HIM architecture and demonstrates the generalizability of this approach.

6.2.2 Other uses of the HIM architecture

In addition to MomConnect, several additional implementations are further demonstrating the usefulness of this architecture. The HIM architecture has been incorporated within the OpenHIE initiative to provide an architecture on which the components of an HIE and point of care systems can interoperate[3]. As part of this work a new, more modern, application is being developed based on the HIM architecture. This project is open source and is being developed by Jembi Health Systems NPC. The source code of this new tool can be found at: `https://github.com/jembi/openhim-core-js`. OpenHIE is expected to be applied in multiple LMICs in the future and thus far the HIM architecture has been found to be suitable as an architecture to facilitate interoperability for these environments.

In addition, the OpenHIM has been implemented by independent parties for a few smaller projects within South Africa and in other LMICs. Examples of this include the exchange of aggregate data between district health informations systems in the Western Cape, South Africa as well as a deployment in Liberia to enable health provider data to be integrated to enable better communications with health providers. The Liberia project, mHero[4], was deployed as a part of the current Ebola response effort.

The experience thus far in term of the level of effort required for re-purposing and the fit for purpose of the HIM architecture has been similar to that of the MomConnect project. This demonstrates the potential for the HIM architecture to be applicable in a wide variety

[3]See `http://ohie.org/architecture/`

[4]See `http://mhero.org/mHero/`

of LMIC environments.

6.3 Architecture quality analysis

The ATAM is used to analyse the HIM architecture to determine its quality with regard to certain quality attributes. These quality attributes are derived from the architectural concerns that we have identified. ATAM is a scenario-based method for evaluating software architectures [41] and is an evolution of the Software Architecture Analysis Method (SAAM). ATAM describes a process whereby a stakeholder group is convened to evaluate the suitability of a software architecture to the architectural concerns. ATAM follows a rigorous process where aspects that affect certain architectural qualities (such as performance, security, modifiability and availability) are identified, prioritised and the architectural decisions that affect these qualities are then determined. Special focus is placed on finding trade-off points where more than one quality is affected by particular architectural decisions. The ATAM analysis does not cover all functional aspect of the architecture but provides some useful insight into the extent to which certain concerns are solved.

ATAM consists of a number of steps that the analysis team leads various stakeholder through in order to elicit information about the architecture and to determine which scenarios and quality attributes are most important for the architecture to succeed. This is done by studying architectural concerns of the architecture. The ATAM steps are listed below:

1. Present the ATAM
2. Present the business drivers
3. Present the architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyse architectural approaches
7. Brainstorm and Prioritize Scenarios

8. Analyse Architecture Approaches (second round)
9. Present Results

In this research the ATAM was performed solely by the author utilising knowledge gained from the Rwandan case study. Thus, certain information sharing steps were not required. Steps 1-3 are left out as they are related to information sharing and step 5 and 7 are combined as both steps involve the identification of scenarios. Due to this step 8 (a repetition of step 6) is no longer needed.

In the sections that follow the outcomes from each of the remaining steps of the ATAM analysis of the HIM architecture are presented.

6.3.1 Identification of business drivers

As part of the ATAM process, business drivers were identified from the requirements from Rwandan health information exchange (RHIE) use case and the role of the HIM architecture in realising that use case.

The core business drivers are listed below and are mapped to the architectural concerns (see Section 4.2 for the enumeration of architectural concerns) from which they were derived.

- Modifiability is important as messaging standards may change over time and new transactions, orchestrations and HISs may be added over time. (Concern #1 and #4)
- Scalability and performance are important as such a system may be deployed at a national level and it should remain functional at scale. (Concern #2)
- Security is important as a patient's health information is highly confidential and should not be tampered with or viewed by unauthorised parties. (Concern #5)
- Availability is important as the HIEs interoperability infrastructure needs to always be available so that vital health data can be captured and retrieved around the clock. (Concern #2)
- Re-usability is important as the architecture should be applicable to a wide variety of LMIC environments. (Concern #6)

Thus, five major quality attributes were chosen for the HIM architecture based on the business drivers:

1. Modifiability
2. Scalability and performance
3. Security
4. Availability
5. Re-usability

Scalability and performance were combined into a single attribute as the HIM architecture handles both of these distinct attributes in a similar way. Scenarios that were identified for either of these attributes were found to affect the other as well.

6.3.2 Utility Tree

Following steps 4 and 5 of the ATAM process a utility tree was drawn up. The utility tree maps scenarios derived from the architectural concerns of the architecture to the quality attributes that the scenarios fall under. To construct the utility tree a number of scenarios that the architecture needs to address must be elicited. The high level requirements that were derived from the Rwandan HIE use case were used to formulate these scenarios along with known scenarios gathered from the Rwandan case study. The utility tree (see 6.5) contains a list of these scenarios that have been prioritised and mapped to the particular quality attribute that it addresses.

Following the ATAM process, the evaluation scenarios that appeared in the utility tree were prioritised along two dimensions. The first dimension is the importance of the scenarios to the success of the architecture and the second dimension is the anticipated difficulty in achieving this scenario. These rating can be seen in brackets on the utility tree scenarios, rated on a scale of high (H), medium (M) and low (L).

Using these priorities the highest rated quality attributes were identified. These attributes are the most important to the success of the architecture and, thus, define its quality. The

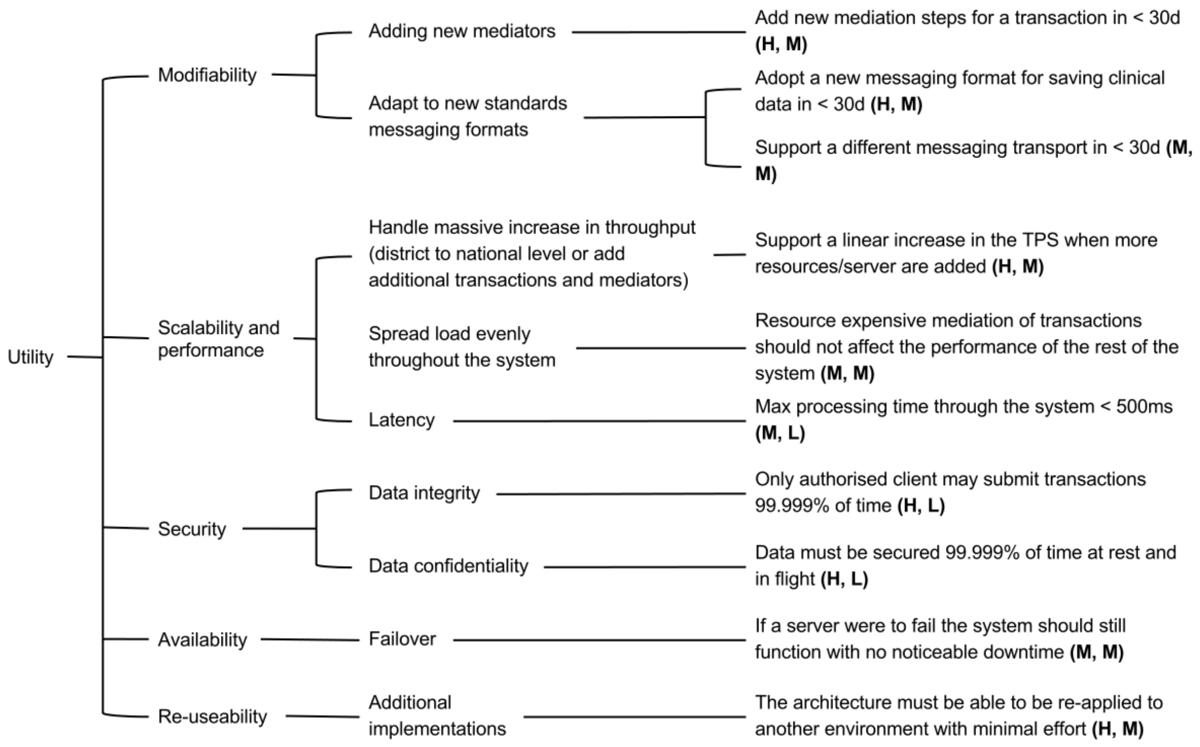

Utility
Modifiability
Adding new mediators
Add new mediation steps for a transaction in < 30d (H, M)
Adapt to new standards messaging formats
Adopt a new messaging format for saving clinical data in < 30d (H, M)
Support a different messaging transport in < 30d (M, M)
Scalability and performance
Handle massive increase in throughput (district to national level or add additional transactions and mediators)
Support a linear increase in the TPS when more resources/server are added (H, M)
Spread load evenly throughout the system
Resource expensive mediation of transactions should not affect the performance of the rest of the system (M, M)
Latency
Max processing time through the system < 500ms (M, L)
Security
Data integrity
Only authorised client may submit transactions 99.999% of time (H, L)
Data confidentiality
Data must be secured 99.999% of time at rest and in flight (H, L)
Availability
Failover
If a server were to fail the system should still function with no noticeable downtime (M, M)
Re-useability
Additional implementations
The architecture must be able to be re-applied to another environment with minimal effort (H, M)

Figure 6.5: HIM architecture utility tree

attributes that had scenarios of high importance to the project were chosen as the attributes that will be evaluated to determine the overall quality of the architecture. Availability was left out of this analysis as the other qualities were determined to be of greater importance. Thus, the overall HIM architecture quality can be defined as:

$$Q_{Ach} = f(Q_{Mod}, Q_{Sca}, Q_{per}, Q_{Sec}, Q_{reuse})$$

The total architecture quality is a function of modifiability, scalability and performance, and security.

6.3.3 Architectural analysis

Subsequent to the generation of the utility tree, each of the evaluation scenarios identified in the utility tree were analysed to identify their mapping to architectural approaches present in the HIM architecture. Risks, sensitivity points and trade-offs associated with the scenarios were also identified by following the ATAM process. Trade-offs are architectural decisions that effect multiple quality attributes, risks describe any architectural risks associated with the architecture and sensitivity points describe an architectural decision that directly influences a particular quality attribute [41].

Below, tables linking each scenario identified in the utility tree to architectural approaches are shown. In addition, any trade-offs, risks or sensitivity points are linked to each architectural decision where applicable. The risks, sensitivity points and trade-offs are shown as symbols (R#, S#, T#) and these are enumerated and described in detail below the scenario tables. Each mapping is grouped under the quality attribute to which it relates.

6.3.3.1 Modifiability scenarios

Scenario 1.1

Scenario: Add new mediation steps for a transaction in < 30d
Attribute: Modifiability
Environment: HIE use case growth/expansion
Stimulus: A new mediator is needed to adapt and/or orchestrate a new transaction
Response: The new mediator is incorporated into the system

Architectural decisions mapping:

Architecture Decisions	Risk	Sensitivity	Trade-off
Mediators are encapsulated and self contained components that can be easily added to the system.	R1		
A router component allows the dynamic configuration of new routes to different mediators.			T1

Scenario 1.2

Scenario: Adopt a new messaging format for saving clinical data in < 30d

Attribute: Modifiability

Environment: HIE update/modernisation

Stimulus: A new messaging format is adopted for an HIE

Response: The new messaging format is implemented in the HIE

Architecture Decisions	Risk	Sensitivity	Trade-off
The normalisation and denormalisation functions are encapsulated so that they may easily be replaced to allow for new messaging formats.			
A canonical model is prescribed for use in mediators such that the orchestration steps are reusable.		S1	

Scenario 1.3

Scenario: Support a different messaging transport in < 30d

Attribute: Modifiability

Environment: HIE update/modernisation

Stimulus: A new transport needs to be supported to support a new message exchange format

Response: Support for a new transport mechanism

Architecture Decisions	Risk	Sensitivity	Trade-off
The encapsulation of the interface component enables a new protocol to be supported through the implementation of a new interface component.			T4

6.3.3.2 Scalability and performance scenarios

Scenario 2.1

Scenario: Support a linear increase in the TPS when more resources/server are added

Attribute: Scalability and performance

Environment: HIE growth

Stimulus: The HIE needs to expand to support more transactions (either increase in number of sending systems or increase in number of transaction types - added mediators)

Response: The system retains acceptable performance even increased throughput

Architecture Decisions	Risk	Sensitivity	Trade-off
A stateless design allows for horizontal scaling of components.			T2

Scenario 2.2

Scenario: Resource expensive mediators should not affect the performance of the rest of the system

Attribute: Scalability and performance

Environment: General HIE operation

Stimulus: A processing expensive mediator is discovered that is affecting the performance of the system

Response: The system deployment is adjusted to solve the performance issue

Architecture Decisions	Risk	Sensitivity	Trade-off
Mediators are encapsulated and separated so that they may be deployed on separate servers if needed.			
Mediators are stateless so they can be deployed redundantly.			Similar to T2

Scenario 2.3

Scenario: Max processing time through the system < 500ms

Attribute: Scalability and performance

Environment: General HIE operation

Stimulus: Incoming transaction

Response: Transactions processing time through the system (excluding the response time of external systems) is < 500ms

Architecture Decisions	Risk	Sensitivity	Trade-off
Data persistence, in the persistence component, occurs asynchronously.			
Mediators should perform external system interactions in parallel where possible.			

6.3.3.3 Security scenarios

Scenario 3.1

Scenario: Only authorised clients may submit transactions 99.999% of time

Attribute: Security

Environment: General HIE operation

Stimulus: Incoming transaction

Response: An authorised clients systems transactions are processed others are rejected

Architecture Decisions	Risk	Sensitivity	Trade-off
A dedicated access control component authenticates and authorises transactions.			

Scenario 3.2

Scenario: Data must be secured 99.999% of time at rest and in flight

Attribute: Security

Environment: General HIE operation

Stimulus: Incoming transaction

Response: Data is protected when being transmitted and when stored

Architecture Decisions	Risk	Sensitivity	Trade-off
Best practise encryption mechanism are enforced by the a dedicated access control component.			T3
The persistence component's data store is only accessible by the core system and public access is blocked.	R2		

6.3.3.4 Availability scenarios

Scenario 4.1

Scenario: If a server were to fail the system should still function with no noticeable downtime

Attribute: Availability

Environment: General HIE operation

Stimulus: A server fails due to hardware or software failure

Response: The system still operates

Architecture Decisions	Risk	Sensitivity	Trade-off
A stateless design allows redundant servers to be deployed, such that if one were to go down the system can remain operational.			Similar to T2

6.3.3.5 Re-usability scenarios

Scenario 5.1

Scenario: The architecture must be able to be re-applied to another environment with minimal effort

Attribute: Re-usability

Environment: New implementation in a new environment

Stimulus: The architecture needs to be adapted to be applied to a new environment

Response: The architecture is flexible enough to be applied to the new environment

Architecture Decisions	Risk	Sensitivity	Trade-off
Mediators are generic and new mediators can be implemented for any new or different use cases			
Architecture is agnostic of messaging standards or protocol			

6.3.3.6 Risks, sensitivity points and trade-offs

The various risks, sensitivity points and trade-offs identified in the architectural mappings above are described below. Each of these are referenced by one of more of the architectural mappings presented above.

Risks:

- R1 - Encapsulating mediators allows the new mediators or modification of existing mediators to be made easily, however, it implies that the mediators be independent and self contained, this affects reusability among the mediator components. There is no defined mechanism by which mediators can reuse code and there will likely be situations where the mediators perform related or identical functions, such as, contacting a particular external service.

- R2 - Storing transactions in the persistence component open up the possibility of the persistence component being compromised and PI being exposed. Care should be taken to protect this data store as much as possible.

Sensitivity:

- S1 - Canonical models often are a reflection of the syntactic and semantic messaging standard used to transmit data, it can be difficult to define a purely abstract independent canonical model that is standards independent. The more generic and abstract the canonical model the greater the potential for modifiability.

Trade-offs:

- T1 - All transaction travel through a the message router. The router component enables easy modifiability as it allows new or updated mediators to be introduced. However, it also implies that every transaction has to pass through the router component. This can affect the performance of the system under load. Care should be taken to ensure this component does not become a bottleneck.

- T2 - The system should be stateless so that the system can be scaled out horizontally. However, working in a stateless mode implied that data (such as client account or transaction logs) be shared among all server in a cluster. This is more expensive than just accessing a systems primary memory. Thus, there is a trade-off of performance to gain scalability.
- T3 - Every transaction received by the system needs to be decrypted. The more expensive the decryption process is often related to how difficult the encryption is to break. Thus, there is a trade-off between security and performance.
- T4 - Down the line processing of a transaction could depend on the transport on which the transaction is received. Fully abstracting away the transport can be challenging and affects the implementability of the design.

6.3.4 Analysis of quality attributes

In the following sections the finding for each of the quality attributes of the HIM architecture are summarised and discussed.

6.3.4.1 Modifiability (Q_{Mod})

From our analysis and prioritization of the various scenarios it was determined that modifiability is the most desired quality attribute for the architecture. This is due to the fact that a country's HIE will always have to respond to change as the need for the exchange of health information grows. New standards will emerge over time and new transactions will be identified and will need to be incorporated into the HIE. Also, new messaging protocols may be developed or existing protocols extended.

The major mechanisms employed by the HIM architecture to deal with this need for change are separation of concerns and encapsulation of components. In particular mediators are designed to be independent and encapsulated such that new mediators, or changes to existing mediators, are simple and do not affect other parts of the system. The architecture also specifies a configurable router component that allows new mediators to be 'plugged' into the system with ease. This allows additional functionality to be added over time and allows new standards to be supported via the development of new mediators or the refactoring of existing mediators. In addition, the interface component is also encapsulated. This allows multiple interface components to exist or a new interface to be developed if need be.

A risk discovered during the evaluation is that due to the mediators being highly independent and encapsulated there is no defined mechanism for the mediators to share or reuse code. This could be a hindrance to both modifiability and to implementability. An implementation of this architecture should take this into consideration and extract common code into libraries for use within the mediators. A trade-off point was also discovered during this evaluation and during the authors experience implementing this architecture. This is that subsequent processing can often depend on the protocol used to receive the message that can make it difficult to separate out the interface details from the processing logic. Building more advanced abstractions improves the modifiability of the interface component but it makes the design more difficult to implement. A good example of this is implementing a simple RESTful web service. RESTful web services use constructs within the HTTP protocol (the HTTP method) to convey semantic meaning for the action of the request. This makes separating the protocol logic from the message processing logic difficult. Another identified risk is the fact that the router component may become of a bottleneck that could affect the performance of the HIM. This is because all messages that flow through an HIM implementation must pass through the router. This is an integral part of the HIM architecture and care should be taken to ensure that an implementation of the router component can support the desired transaction load for an particular implementation.

6.3.4.2 Scalability and Performance (Q_{Sca} and Q_{per})

Scalability and performance also feature highly as a desired quality attributes in our evaluation. This is because a country's HIE infrastructure could expand from a pilot at district level to a national deployment. Performance is also key as every message that is sent to the HIE infrastructure must pass through the HIM. The performance overhead must be kept to a minimum so that requests may be responsive. In specific terms we specified that the total processing time through a HIM implementation should not exceed 500ms per request. Many requests must also be handled simultaneously as any point of care system (at clinics, hospitals, laboratories etc.) will be sending in requests sporadically. An HIM implementation should support a linear increase in transactions per second (TPS) as more servers are added. Mediators will also be of differing complexity and some will be more expensive (in terms of processing requirements) than others. The architecture must enable additional resources to be dedicated to those expensive mediators while still allowing other mediators and other parts of the system to respond quickly.

The architecture employs two major approaches to allow for these scenarios. The interface component, the access control component, the persistence component and each of the mediators are stateless. This allows an entire HIM implementation to be deployed redundantly over a number of servers. This allows for scalable performance and handling of a high transaction load. It enables a relatively linear reaction to an increase in TPS as more servers can be added dynamically to handle increased transaction load. However, additional overhead may be incurred as distribution of the system increases. The stateless design along with the fact that mediators are independent and encapsulated allow mediators to be split onto servers that are separate from the rest of the HIM implementation. This also allows mediators to be scaled horizontally if they need additional processing power. Simple load balancers can be placed in front of the HIM architecture or individual mediators to enable this horizontal scalability. To improve performance HIM implementations are also encouraged to perform any I/O operations in an asynchronous manner and in parallel where possible. This mainly includes mediators that are communicating with external systems and the persistence component.

During this evaluation we discovered a trade-off relating to the stateless design. While the stateless design allows us to easily expand to multiple servers and service a variety of different transaction loads it also affects the outright system performance. A stateless design implies that information cannot be cached or stored in a server's primary memory because other servers need to access to this information. All data and metadata must be persisted to a shared data store which can be more expensive than the in-memory alternative. Caching software could be used to partly alleviate this problem, however, it is unlikely that the raw speed of in-memory data access could be achieved.

6.3.4.3 Security (Q_{Sec})

Security also featured highly during the prioritisation of scenarios. The security of patient information is of vital importance for an HIE. Data should be protected both when it is transmitted between systems as well as when it is at rest in a permanent data store. Additionally only those who are allowed to view and update a patient's information should be allowed access to it.

The HIM architecture does not specify how endpoints may be secured as this will depend on the communication protocol chosen by the HIM implementation. For most protocols there are best practise methods to achieving adequate security. However, the HIM architecture does define an access control component that enables the HIM to ensure that only authorized persons may access a particular patient's information. The access control component determines both identity (authentication) and the authority (authorisation) of the person or system making a request and is able to allow or deny the request at a central location. The HIM architecture also specifies that the data store used for storing transactions only be accessible by the core HIM application and access be blocked from any other sources.

As with most secure systems there is a trade-off between performance of the system and the level of security achieved. This trade-off is left to the implementing party to determine based on their requirements.

6.3.4.4 Re-usability (Q_{Reuse})

The HIM architecture was designed to be generic so that it may be reapplied in other environments. This is accomplished firstly with mediators that provide a framework for implementing message transformation or orchestration logic. Secondly, the HIM architecture is agnostic of messaging standards that are used (through the normalisation and de-normalisation feature of mediators) as well as the communication protocol that is used (through the encapsulation of the interface component). This allows for greater flexibility and reuse.

6.4 Comparison with existing approaches

The HIM architecture provides a set of components that enable interoperability between disparate health information systems. It describes a central component that is able to perform many of the common tasks that are required for interoperability to safely occur; such as security, message logging, transaction auditing and message mediation. The Enterprise Service Bus (ESB) architectural model provides a set of features that allows simplified enterprise application integration. These features include message orchestration, transformation and routing. The HIM architecture provides a component-based architecture that further specialises the ESB architecture.

The HIM architecture is influenced by previous approaches such as Mohawk's HIAL [9, 72] reference implementation and HSB by Ryan *et al.* [59] to help define the functions and structure of the HIM components. While previous approaches were implemented in high income countries, this research investigates its applicability in LMICs.

The HIAL architecture and the HSB make use of an ESB for communication between the various components of the HIE. Similarly, the HIM uses an ESB to simplify the communication between different components. Also, many of the functions that are described by the HIAL and the HSB are also provided by the more general HIM architecture. These functions include logging, access control, message transformation and message orchestration. The HIAL provides some on-ramp and off-ramp functions to normalise messages into a canonical form. The HIM architecture extends this idea in its normalisation and denormalisation components to not only consider message structure but to also include an explicit step for message translation to allow for semantic differences.

Furthermore, this work makes use of the ideas presented in Xu *et al.* [70, 71] to provide some formalism for the way in which messages are mediated. This mechanism allows us to make the architecture agnostic of message format, allowing implementers to implement whatever standards stack they wish in order to achieve semantic interoperability. This separation allows the architecture to be applicable even as different health information standards gain or lose adoption. Xu *et al.* describes an architecture that enables message mediation but we believe there are additional considerations. Their architecture has mediators between the central component and the service providers. We, however, extend this concept to the client side as well. Both incoming and outgoing messages should be mediated into a format that the central component can process. This allows clients to send messages that are not in the canonical form and allows them to be transformed into the canonical form by the HIM implementation. This addition also simplifies client message generation which, as stated above, is a desirable feature for low resource settings.

The HIM architecture differs from previous approaches in that it is not designed for specific environments and use cases. Both the HIAL and the HSB are designed around specific messaging standards which make them use case specific. We believe that while many of the challenges faced are similar, there are nuances to working in low resource settings that are unique to our study. In this research we have created a generalised architecture and abstracted away the actual transaction implementations to allow it to apply to a variety of use cases and environments. All implementation specific logic and choice of standards are encapsulated into transaction mediation channels. We have defined a structure for these mediation channels that an implementation may follow to gain semantic interoperability.

	District	National	Tx/visit	District TPS	National TPS
Total visits per month	36 072.22	1 217 666.67			
SHR: Save encounter	108 216.67	3 653 000	3	0.17	5.77
SHR: Get encounter	36 072.22	1 217 666.67	1	0.06	1.92
SHR: Total	**144 288.89**	**4 870 666.67**	**4**	**0.23**	**7.69**
SHR: Peak load (total x3)				**0.68**	**23.06**
CR: Save patient	36 072.22	1 217 666.67	1	0.06	1.92
CR: Edit patient	3 607.22	121 766.67	0.1	0.01	0.19
CR: Query patient	45 090.28	1 522 083.33	1.25	0.07	2.40
CR: Total	**84 769.72**	**2 861 516.67**	**2.35**	**0.13**	**4.52**
CR: Peak load (total x3)				**0.40**	**13.55**
Overall TOTAL	**229 058.61**	**7 732 183.33**	**6.35**	**0.36**	**12.20**
Overall Peak Load				**1.08**	**36.61**

Table 6.1: Estimated transactional load for the Rwandan HIE

Chapter 7

Conclusion

This research has demonstrated that an ESB-based architecture can successfully enhance interoperability between disparate HISs connected through a health information exchange in LMICs. A set of concerns for an architecture that facilitates interoperability were derived from the the Rwandan HIE case study. These concerns were used to drive the design of the Health Information Mediator architecture (HIM), an ESB-based architecture of a software component for use within health information exchanges that aims to facilitate and simplify interoperability. A reference implementation of the architecture, the OpenHIM, was implemented within the Rwandan HIE. The architecture was analysed against the architectural concerns and the OpenHIM implementation in Rwanda was used to validate the architecture. In addition, the Architecture Trade-off Analysis Method (ATAM) analysis was used to formally analyse the quality of the architecture. The architecture was also compared and contrasted to existing architectures described in the literature.

Seven architectural concerns were derived and enumerated. These concerns specify that an architecture that enables interoperability between HISs in LMICs must: facilitate interoperability between disparate and heterogeneous systems, both existing and future; balance central governance with local autonomy; adapt and scale within a changing environment; prevent local changes from propagating through the system; provide a low barrier to entry to connect new and legacy systems; be secure and auditable; and be reusable across a multitude of environments.

The HIM architecture was introduced to address the concerns and was described using ISO 42010 [39, 21] architecture descriptions. Three different views of the architecture were described, each framing a different set of concerns. The three viewpoints used were scalability, modifiability and interoperability. The HIM architecture description presents a proposed solution for simplifying interoperability in low resource countries like Rwanda and formalises the description of such an architecture so that it can be reused in other settings.

A reference application, the Open Health Information Mediator (OpenHIM) was created and implemented in Rwanda as part of the RHIE project. It allowed a number of point-of-care systems to connect to infrastructure services to form an HIE for the pilot district of Rwamagana. The implementation of the HIM architecture in the RHIE has been successful and has provided the first validation of the HIM architecture in LMICs.

The architecture was analysed against each of the architectural concerns and the extent to which each of these is addressed is discussed. Experience with the OpenHIM implementation in Rwanda is used to support this discussion. The successful implementation of the HIM architecture within the RHIE demonstrates that an ESB-based architecture may be suitable for use within LMICs.

The HIM architecture simplifies the interfaces between the point-of-care systems and the infrastructure services. It also enables complex orchestration logic to be encapsulated centrally and provides a framework for supporting syntactic and semantic interoperability challenges. While, these challenges were not solved entirely, the framework that the HIM provided was demonstrated to enable syntactic and semantics challenges to be addressed at implementation time.

A key concern of the HIM architecture was re-usability to enable it to be easily re-applied in multiple environments. Importantly, it has seen adoption within multiple environments within sub-Saharan African LMICs. Besides the Rwandan HIE, the HIM architecture has proved to be useful within the MomConnect project in South Africa. This finding is important as it shows that the HIM could form the basis of a generic framework for interoperability between HISs in LMICs. This preliminary framework enables the cost, both in time and resources, to be reduced as it provides patterns that are repeatable within multiple environments. It is also able to reduce the complexity of instantiating an HIE. This is particularly important within LMICs as cost and complexity are the two main barriers preventing interoperability between HISs.

The quality of the architecture was analysed formally using the Architecture Trade-off Analysis Method. Four vital qualities for a architecture suitable of health information mediation for the RHIE were identified. These are modifiability, performance and scalability, security and re-usability. The analysis identified the architectural approaches that enables the HIM architecture to achieve each of its vital quality attributes.

Modifiability is handled by prescribing a high level of encapsulation of the mediation components, by allowing messages to be dynamically routed to mediator components and by prescribing a clear separation of concerns for each mediator such that they only deal with a single particular problem. This separation of concerns allows the mediators to be simple and allows them to be easily re-used or replaced as needed.

Performance and scalability are enabled by employing a stateless design of the different components of the architecture as well as encapsulating components such that they may be spread over separate servers. An initial empirical analysis of the performance and scalability for a national HIE deployment for RHIE is performed and the OpenHIM reference application proves to be able to handle national level load using modest hardware.

To enable the security attribute, the HIM architecture employs an access control component to ensure that patient information is kept secure. It also specifies that data at rest should be encrypted to ensure it is protected from intruders. Re-usability is enabled through an extensible mediator design and through encapsulation of the interface component of the architecture.

The architecture was also compared against other existing approaches. The HIM architecture was found to incorporate three novel features that are not found in previous architectures: it is agnostic of messaging formats used to communicate health data; it is agnostic of the health information transactions that it supports; and it allows transaction to be mediated at both the inbound and outbound interface. This makes it suitable in a number of environments where health information mediation is required and enables flexibility such that existing legacy systems can be connected more easily.

The HIM architecture provides a useful platform for solving health information mediation problems and enables an HIE to be more easily constructed as a result. However, it does not solve all interoperability challenges. Core parts of the problem are left to the implementing party to solve. This includes syntactic and semantic interoperability problems between disparate systems. The implementing party must still implement the details of the normalisation and de-normalisation components to enable messages to be exchanged between systems that accept different message formats or represent data in a semantically different way. The HIM architecture provides the platform on which these solutions can be built, however, it is currently not possible to automatically convert message syntax or semantics between heterogeneous systems. This remains an open problem worthy of further investigation.

Additional future work includes extensions and formalisation of the generic framework to better enable interoperability between HISs in LMIC. Particularly, the infrastructure services that are most commonly required could be explored along with a study of the standard data exchange formats and standards that could be prescribed to enable syntactic and semantic interoperability.

The HIM architecture was able to overcome many of the key concerns when facilitating interoperability between disparate health information systems in low resource settings. It provides mechanisms to scale in terms of performance both horizontally and vertically, it simplifies the complexity needed to enable both existing and new client systems to connect to an HIE infrastructure, it is adaptable and extensible to future requirements, it protects systems participating in the HIE from changes made to other systems and it facilitates interoperability between the components of an HIE. It has also been applied and validated in real world settings. The HIM architecture was able to usefully simplify the mediation of health information which in turn can enable HIEs to be more easily constructed, particularly for LMICs where resources are constrained. In addition, it has seen adoption in multiple projects within sub-Saharan Africa. These projects are ongoing and will likely drive the evolution of this architecture. Further, the architecture and its concerns show promise in forming the foundation of a general framework for the construction of HIEs within LMICs.

Bibliography

[1] AbouZahr, C., and Boerma, T. Health information systems: the foundations of public health. *Bulletin of the World Health Organization 83*, 8 (Aug. 2005), 578–583. Available from: `http://dx.doi.org//S0042-96862005000800010`.

[2] Adebesin, F., Foster, R., Kotzé, P., and Van Greunen, D. A review of interoperability standards in e-Health and imperatives for their adoption in Africa. *South African Computer Journal*, 50 (2013), 55–72.

[3] Allen, C., Jazayeri, D., Miranda, J., Biondich, P. G., Mamlin, B. W., Wolfe, B. A., Seebregts, C., Lesh, N., Tierney, W. M., and Fraser, H. S. Experience in implementing the OpenMRS medical record system to support HIV treatment in Rwanda. *Studies in health technology and informatics 129*, Pt 1 (2007), 382–386. Available from: `http://view.ncbi.nlm.nih.gov/pubmed/17911744`.

[4] Azubuike, M., and Ehiri, J. Health information systems in developing countries: benefits, problems, and prospects. *The Journal of the Royal Society for the Promotion of Health 119*, 3 (Sept. 1999), 180–184. Available from: `http://rsh.sagepub.com/cgi/doi/10.1177/146642409911900309`.

[5] Bengtsson, P., Lassing, N., Bosch, J., and van Vliet, H. Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software 69*, 1-2 (Jan. 2004), 129–147. Available from: `http://linkinghub.elsevier.com/retrieve/pii/S0164121203000803`.

[6] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. Simple object access protocol (SOAP) 1.1. Tech. Rep. May, World Wide Web Consortium (W3C), 2000. Available from: `http://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W000520N.pdf`.

[7] Braa, J. r., Kanter, A. S., Lesh, N., Crichton, R., Jolliffe, B., Sæ bø, J., Kossi, E., and Seebregts, C. J. Comprehensive yet scalable health information systems for low resource settings: a collaborative effort in Sierra Leone. *AMIA Annual Symposium proceedings 2010* (Jan. 2010), 372–376.

[8] Braa, J. r., and Muquinge, H. Building collaborative networks in Africa on health information systems and open source software development - Experience from the HISP/BEANISH network. *IST Africa* (2007).

[9] Canada Health Infoway. EHRS Blueprint: An Interoperable EHR Framework - Executive overview. Tech. Rep. April, 2006. Available from: `http://www.infoway-inforoute.ca/working-with-ehr/solution-providers`.

[10] Chappell, D. *Enterprise Service Bus: Theory in Practice*. O'Reilly Media, July 2004.

[11] Chen, R. *Towards interoperable and knowledge-based electronic health records using archetype methodology*. PhD thesis, Department of Biomedical Engineering, Linköpings universitet, 2009. Available from: `http://www.worldcat.org/isbn/9789173935043`.

[12] Cimino, J. Review paper: coding systems in health care. *Methods of information in medicine* (1996). Available from: http://people.dbmi.columbia.edu/cimino/Publications/1996-MethInfMed-CodingSystemsinHealthCare.pdf.

[13] Cios, K. J., and Moore, G. W. Uniqueness of medical data mining. *Artificial intelligence in medicine 26*, 1-2 (2002), 1–24. Available from: http://www.ncbi.nlm.nih.gov/pubmed/12234714.

[14] Coiera, E. Building a National Health IT System from the middle out. *Journal of the American Medical Informatics Association : JAMIA 16*, 3 (2009), 271–3. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2732241&tool=pmcentrez&rendertype=abstract.

[15] Darke, P., Shanks, G., and Broadbent, M. Successfully completing case study research: combining rigour, relevance and pragmatism. *Information Systems Journal 8*, 4 (Oct. 1998), 273–289. Available from: http://doi.wiley.com/10.1046/j.1365-2575.1998.00040.x.

[16] Degoulet, P., Sauquet, D., and Jaulent, M.-c. Semantic interoperability in health information systems. *Proc of the IMIA WG 6 Conf on Natural Language and Med Concept Represenation* (1997). Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.2932&rep=rep1&type=pdf.

[17] Dixon, B. E., Zafar, A., and Overhage, J. M. A Framework for evaluating the costs, effort, and value of nationwide health information exchange. *Journal of the American Medical Informatics Association : JAMIA 17*, 3 (2010), 295–301. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2995720&tool=pmcentrez&rendertype=abstracthttp://dblp.uni-trier.de/db/journals/jamia/jamia17.html#DixonZO10.

[18] Dogac, A., Namli, T., Okcan, A., Laleci, G., Kabak, Y., and Eichelberg, M. Key issues of technical interoperability solutions in eHealth. In *Proceedings of eHealth 2006 High Level Conference Exhibition* (2006), pp. 1–11. Available from: http://www.ehealthconference2006.org/pdf/Dogac_proc.pdf.

[19] Dolan, T. J. *Architecture assessment of information-system families: a practical perspective*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[20] Eichelberg, M., Aden, T., Riesmeier, J., Dogac, A., Laleci, G. B., and Riesmejer, J. A survey and analysis of Electronic Healthcare Record standards. *ACM Computing Surveys 37*, 4 (Dec. 2005), 277–315.

[21] Emery, D., and Hilliard, R. Updating IEEE 1471: Architecture Frameworks and Other Topics. In *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)* (Washington, DC, USA, Feb. 2008), IEEE, pp. 303–306.

[22] Ferranti, J. M., Musser, R. C., Kawamoto, K., and Hammond, W. E. The Clinical Document Architecture and the Continuity of Care Record: A Critical Analysis. *Journal of the American Medical Informatics Association 13*, 3 (May 2006), 245–252. Available from: http://dx.doi.org/10.1197/jamia.M1963.

[23] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol–HTTP/1.1, RFC 2616, June, RFC 2616, June, 1999.

[24] Fielding, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine - Irvine, CA 92697, USA, 2000.

[25] Forrey, A., and McDonald, C. Logical observation identifier names and codes (LOINC) database: a public use set of codes and names for electronic reporting of clinical laboratory test results. *Clinical Chemistry 42*, 1 (1996), 81–90. Available from: `http://www.clinchem.org/content/42/1/81.short`.

[26] Garde, S., Chen, R., Leslie, H., Beale, T., McNicoll, I., and Heard, S. *Archetype-Based Knowledge Management for Semantic Interoperability of Electronic Health Records*. IOS Press, 2009, pp. 1007–1011.

[27] Garlan, D. Software architecture: a roadmap. *Proceedings of the Conference on the Future of Software Engineering* (2000), 91–101. Available from: `http://dl.acm.org/citation.cfm?id=336537`.

[28] Gibbons, P., Arzt, N., Burke-Beebe, S., Chute, C., Dickinson, G., Flewelling, T., Jepsen, T., Kamens, D., Larson, J., Ritter, J., Rozen, M., Selover, S., and Stanford, J. Coming to Terms: Scoping Interoperability for Health Care. Tech. rep., Health Level Seven EHR Interoperability Work Group, Feb. 2007.

[29] Gruber, T. A translation approach to portable ontology specifications. *Knowledge acquisition*, April (1993).

[30] He, H. What is service-oriented architecture. Tech. rep., O'Reilly Press, 2003.

[31] Health Level Seven International. HL7 version 2, Health Level Seven International. Available from: `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185`.

[32] Health Level Seven International. HL7 version 3, Health Level Seven International. Available from: `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=186`.

[33] Huhns, M., and Singh, M. Service-oriented computing: Key concepts and principles. *Internet Computing, IEEE*, February (2005), 2–8.

[34] IEEE Standards Board. IEEE Standard Glossary of Software Engineering Terminology. Tech. rep., 1990. Available from: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342`.

[35] IHE International Inc. IHE IT Infrastructure (ITI) Technical Framework: Volume 1 (ITI TF-1) Integration Profiles. Tech. rep., 2012.

[36] IHE International Inc. IT Infrastructure Technical Framework: Volume 2a (ITI TF-2a) Transactions Part A - Sections 3.1 - 3.28. Tech. rep., 2012.

[37] IHE International Inc. IT Infrastructure Technical Framework: Volume 2b (ITI TF-2b) Transactions Part B - Sections 3.29 - 3.51. Tech. rep., 2012.

[38] Ionita, M., Hammer, D., and Obbink, H. Scenario-based software architecture evaluation methods: An overview. *ICSE/SARA* (2002). Available from: `http://ceit.aut.ac.ir/islab/courses/lss/files/92/Arch_Eval.rar`.

[39] ISO. ISO/IEC FDIS 42010 IEEE P42010/D9. Systems and software engineering - Architecture description. Tech. rep., ISO, Mar. 2011.

[40] Kazman, R., Bass, L., Abowd, G., and Webb, M. SAAM: a method for analyzing the properties of software architectures. *Proceedings of 16th International Conference on Software Engineering* (1994), 81–90. Available from: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=296768`.

[41] Kazman, R., Klein, M., and Clements, P. ATAM: Method for architecture evaluation. Tech. Rep. August, 2000. Available from: `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA382629`.

[42] Lucas, H. Information and communications technology for future health systems in developing countries. *Social science and medicine (1982) 66*, 10 (May 2008), 2122–32. Available from: `http://www.ncbi.nlm.nih.gov/pubmed/18343005`.

[43] Mamlin, B. W., Biondich, P. G., Wolfe, B. A., Fraser, H., Jazayeri, D., Allen, C., Miranda, J., and Tierney, W. M. Cooking up an open source EMR for developing countries: OpenMRS - a recipe for successful collaboration. *AMIA Symposium* (2006), 529–533. Available from: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1839638/`.

[44] Michael, M., and Moreira, J. Scale-up x scale-out: A case study using nutch/lucene. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International* (2007). Available from: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4228359`.

[45] Ministry of Health Rwanda. Health Sector Strategic Plan. July 2009 - June 2012, Government of Rwanda Ministry of Health, July 2009. Available from: `http://www.unicef.org/rwanda/events_9491.html`.

[46] Moodley, D., Pillay, A. W., and Seebregts, C. J. Position Paper: Researching and Developing Open Architectures for National Health Information Systems in Developing African Countries. In *International Symposium on Foundations of Health Information Engineering and Systems* (Aug. 2011), vol. 7151 of *Lecture Notes in Computer Science*, Springer.

[47] Mudaly, T., and Moodley, D. Architectural frameworks for developing national health information systems in low and middle income countries. *Enterprise Systems Conference (ES), 2013* (2013), 1–9. Available from: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6690083`.

[48] National Department of Health - South Africa, and Council for Scientific and Industrial Research - South Africa. Health Normative Standards

Framework for Interoperability in eHealth in South Africa. Version 2.0. Tech. rep., 2014.

[49] National E-Health Transition Authority. NEHTA Interoperability Framework v2.0, National E-health Transition Authority, Australia, 2007. Available from: `http://www.nehta.gov.au/implementation-resources/ehealth-foundations/ehealth-interoperability-framework`.

[50] National E-health Transition Authority. NEHTA Blueprint - Blueprint v2.0, National E-health Transition Authority, Australia, 2012. Available from: `http://www.nehta.gov.au/implementation-resources/ehealth-foundations/nehta-blueprint`.

[51] Object Management Group (OMG). OMG Unified Modeling Language, Object Management Group, 2011. Available from: `http://www.omg.org/spec/UML/2.4.1/`.

[52] Ouksel, A. M., and Sheth, A. Semantic Interoperability in Global Information Systems: A brief introduction to the research area and the special section. *SIGMOD record 28*, 1 (Mar. 1999), 5–12.

[53] Ouksel, A. M., and Sheth, A. Semantic Interoperability in Global Information Systems: A brief introduction to the research area and the special section. *SIGMOD record 28*, 1 (1999), 5–12.

[54] Papazoglou, M. Service-oriented computing: Concepts, characteristics and directions. *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE '03)* (2003).

[55] Park, J. Information systems interoperability: What lies beneath? *ACM Transactions on Information Systems (TOIS) 22*, 4 (2004), 595–632.

[56] Peltz, C. Web services orchestration and choreography. *Computer 36*, 10 (Oct. 2003), 46 – 52.

[57] Postel, J. User datagram protocol. *RFC 768, USC/Information Sciences Institute* (1980). Available from: `http://tools.ietf.org/html/rfc768?ref=driverlayer.com`.

[58] Ryan, A., and Eklund, P. A framework for semantic interoperability in healthcare: a service oriented architecture based on health informatics standards. *Studies in health technology and informatics 136* (Jan. 2008), 759–64.

[59] Ryan, A., and Eklund, P. The Health Service Bus: an architecture and case study in achieving interoperability in healthcare. *Studies in health technology and informatics 160*, Pt 2 (2010), 922–926. Available from: `http://view.ncbi.nlm.nih.gov/pubmed/20841819`.

[60] Schmidt, M. T., Hutchison, B., Lambros, P., and Phippen, R. The Enterprise Service Bus: Making service-oriented architecture real. *IBM Systems Journal 44*, 4 (2005), 781–797. Available from: `http://dx.doi.org/10.1147/sj.444.0781`.

[61] Seebregts, C. J., Mamlin, B. W., Biondich, P. G., Fraser, H. S. F., Wolfe, B. A., Jazayeri, D., Allen, C., Miranda, J., Baker, E., Musinguzi, N., Kayiwa, D., Fourie, C., Lesh, N., Kanter, A., Yiannoutsos, C. T., and Bailey, C. The OpenMRS Implementers Network. *International Journal of Medical Informatics 78*, 11 (Nov. 2009), 711–720. Available from: `http://dx.doi.org/10.1016/j.ijmedinf.2008.09.005`.

[62] Sheth, A. Changing focus on interoperability in information systems: From system, syntax, structure to semantics. *Interoperating geographic information systems* (1999), 5–29.

[63] Siegel, E., and Channin, D. Integrating the Healthcare Enterprise: A Primer. *Radiographics 21*, 5 (2001), 1339–1341. Available from: `http://inderscience.metapress.com/index/B08062038T4L70Q6.pdf`.

[64] Valipour, M. H., Amirzafari, B., Maleki, K. N., and Daneshpour, N. A brief survey of software architecture concepts and service oriented architecture. *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on* (2009).

[65] Wiederhold, G. Mediators in the architecture of future information systems. *Computer* (1992).

[66] World Health Organization. Everybody's business–strengthening health systems to improve health outcomes: WHO's framework for action. Tech. rep., World Health Organization (WHO), 2007. Available from: `https://extranet.who.int/iris/restricted/handle/10665/43918`.

[67] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.1, World Wide Web Consortium (W3C), 2006. Available from: `http://travesia.mcu.es/portalnb/jspui/handle/10421/2507`.

[68] World Wide Web Consortium (W3C). SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2007. Available from: `http://www.w3.org/TR/soap12-part1/`.

[69] Wright, G., and Stevens, W. *TcP/IP Illustrated*, vol. 1. 1995. Available from: `http://books.google.com/books?hl=en&lr=&id=6H9AxyFd0v0C&oi=fnd&pg=PT4&dq=TCP/IP+Illustrated&ots=b6gJ3z5QmG&sig=7wJPPantrBUf_DCKuN7ZblyMdso`.

[70] Xu, Y., Sauquet, D., Degoulet, P., and Jaulent, M.-C. Component-based mediation services for the integration of medical applications. *Artificial Intelligence in Medicine 27*, 3 (Mar. 2003), 283–304.

[71] Xu, Y., Sauquet, D., Zapletal, E., Lemaitre, D., and Degoulet, P. Integration of medical applications: the "mediator service" of the SynEx platform. *International journal of medical informatics 58-59* (Oct. 2000), 157–166.

[72] Yendt, M., Bender, D., and Minaji, B. Developing an Open Source Reference Implementation of the Canadian Electronic Health Records Solution. *Open Source Business Resource November 2* (Nov. 2008).