Service Oriented Framework for Personal Health Application Development

by

Shirin Ghorbani

Previous Degrees (Bachelor, Azad University, 2010)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Masters of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s): Weichang Du, PhD, Faculty of Computer Science
Examining Board: Bradford G. Nickerson, PhD, Faculty of Computer Science,
Patricia Evans, PhD, Faculty of Computer Science,
Donglei Du, PhD, Faculty of Business

This thesis is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

September, 2013

©Shirin Ghorbani, 2013
Abstract

In this work, Personal Health Service Framework (PHSF) is proposed which is an open architecture for developing patient-centric health applications and monitoring systems that use personal health monitoring devices for data gathering. PHSF uses service-oriented architecture to provide a baseline for developing health-related applications. PHSF services support personal health self-monitoring and remote monitoring, and to link health care providers and patients. PHSF is implemented as a combination of web services in four layers: the front-end services layer to support end user interactions, the intermediate services layer to do various tasks for the front-end services, the data access services layer to access the databases, and the database layer. These services are grouped into 7 service components: device client services, patient application services, provider client services, provider application services, information storage and retrieval services, privacy and security services, and transformation services. A prototype implementation of the PHSF and a heart monitoring application implemented with PHSF are presented in the thesis.
Acknowledgements

This project would not have been possible without the support of many people. A special thanks to my supervisor, Dr. Weichang Du, for all the guidance and his patience throughout the entire process.

Also many thanks and appreciation to my parents, uncle and aunt and numerous friends who endured this long process with me, always offering support and love.
# Table of Contents

Abstract ........................................... ii
Acknowledgments .................................. iii
Table of Contents ................................ vii
List of Tables ..................................... viii
List of Figures ................................... xi
Abbreviations .................................... xii

## 1 Introduction ................................ 1
   1.1 Motivation .................................. 1
   1.2 Thesis Objectives and Research Problems .... 3
   1.3 Thesis solution overview ................... 4
   1.4 Structure of thesis .......................... 6

## 2 Background and Related Work ........... 8
   2.1 E-health ..................................... 8
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1 Personal Health Measurement Devices and Sensors</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 E-health Applications and E-health Portals</td>
<td>11</td>
</tr>
<tr>
<td>2.1.3 Health Monitoring Systems</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4 E-health Databases</td>
<td>14</td>
</tr>
<tr>
<td>2.1.4.1 Personal Health Record</td>
<td>15</td>
</tr>
<tr>
<td>2.1.4.2 Electronic Medical Record</td>
<td>15</td>
</tr>
<tr>
<td>2.1.4.3 Electronic Health Record</td>
<td>16</td>
</tr>
<tr>
<td>2.1.5 Health Level 7</td>
<td>17</td>
</tr>
<tr>
<td>2.1.6 E-health Contributions to Healthcare</td>
<td>19</td>
</tr>
<tr>
<td>2.2 Service Computing</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Web Services</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1.1 Service Oriented Architecture</td>
<td>22</td>
</tr>
<tr>
<td>2.2.1.2 Web Services in E-Health</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Related work</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1 Virtual Remote Nursing System (VRN)</td>
<td>27</td>
</tr>
<tr>
<td>2.3.2 A Cloud Computing Solution for Patient Data Collection in Health Care Institutions</td>
<td>30</td>
</tr>
<tr>
<td>2.3.3 E-Health Support Services Based on SOA</td>
<td>31</td>
</tr>
<tr>
<td>2.3.4 Cloud Computing Aware Ubiquitous Health Care System</td>
<td>33</td>
</tr>
<tr>
<td>3 Framework Design</td>
<td>36</td>
</tr>
<tr>
<td>3.1 Service description</td>
<td>37</td>
</tr>
<tr>
<td>3.1.1 Device Client</td>
<td>40</td>
</tr>
</tbody>
</table>
3.1.2 Patient Application
3.1.3 Provider Client
3.1.4 Provider Application
3.1.5 Information Storage and Retrieval
3.1.6 Data Transformation
3.1.7 Privacy and Security
3.2 Device Interfaces
3.2.1 Patient Device Interfaces
3.2.2 Provider Device Interfaces
3.3 Services Relationships and Dependencies

4 Framework Implementation
4.1 Web Service Implementation
4.1.1 Approaches to Web Service Development
4.2 Database implementation
4.2.1 Health Records Database
4.2.2 Provider Database
4.2.3 Device Database
4.2.4 Security Database

5 Case Study
5.1 Requirements
5.2 Design and Implementation
5.2.1 Implementation of Use Cases for Patients
6 Experiments and Evaluation 87
   6.1 Scenario Based Experiments 87
   6.2 Scalability Test Experiments 94
      6.2.1 Experiment results 95
   6.3 Evaluation on Framework quality 98
   6.4 Comparisons 100

7 Conclusion 103
   7.1 Summary 103
   7.2 Contributions 105
      7.2.1 Technical Contributions 105
      7.2.2 Practical Contributions 106
   7.3 Future Work 106

Bibliography 116

Vita
List of Tables

6.1 Response time for the sequence of services for data gathering scenario. ........................................ 93
6.2 Functionality comparison of PHSF and other systems. .... 101
6.3 Software quality comparison of PHSF and other systems. ... 102
List of Figures

1.1 Component view of the PHSF ........................................ 5
1.2 Layered view of the PHSF ........................................... 6

2.1 Web services architecture ............................................ 23
2.2 High level view of Virtual Nursing System (VRN) from ([32]). 28
2.3 Patient’s data collection in health care institutions from ([44]). 31
2.4 EHMS monitoring approach (from[37]). .......................... 32
2.5 General Architecture of the HIMS [42] ........................... 34

3.1 PHSF layers .......................................................... 38
3.2 PHSF components ..................................................... 39
3.3 Service dependency diagram for PHSF ............................ 52

4.1 Implementation of retrieve health record service as an example
interface ................................................................. 57
4.2 The code for the retrieve health records service ............... 58
4.3 Parts of the beans.xml file for the retrieve health records service. 59
4.4 Example of invoking a web service and developing a client. . 60
4.5 SOAP request for Retrieve Health Record for patientID=1. . 60
4.6 SOAP response for Retrieve Health Record for patientID=1.  . 61
4.7 The PHSF data access layer.  . 62
4.8 The PHSF health record database.  . 63
4.9 The PHSF provider database.  . 64
4.10 The PHSF device database.  . 66
4.11 The PHSF security database.  . 67
5.1 Use case diagram for a mobile heart monitoring system.  . 70
5.2 Patient register page in the heart monitoring application.  . 72
5.3 Patient home page in the heart monitoring application.  . 73
5.4 Health record page in the heart monitoring application.  . 73
5.5 Authorizing provider page in the heart monitoring application.  . 74
5.6 Schema configuration by Patient in the heart monitoring application.  . 75
5.7 Sensor configuration page by Patient in the heart monitoring application.  . 76
5.8 Monitoring configuration page by patient in the heart monitoring application.  . 77
5.9 Measurement notice reminder in the heart monitoring application.  . 79
5.10 Emergency alarm in the heart monitoring application.  . 81
5.11 Activity diagram for the monitor scheduling scenario in the PHSF.  . 82
5.12 Activity diagram for the gathering data scenario in the PHSF. 82
5.13 Provider report page in the heart monitoring application. . . . 83
5.14 Provider home page in the heart monitoring application. . . . 84

6.1 Data gathering scenario in a monitoring session for the heart
monitoring application Part 1. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
6.2 Data gathering scenario in a monitoring session for the heart
monitoring application Part 2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
6.3 Effect of increasing device request traffic on response time in
the heart monitoring application. . . . . . . . . . . . . . . . . . . . . . . . . . 96
6.4 Effect of increasing provider request traffic on response time
in the heart monitoring application. . . . . . . . . . . . . . . . . . . . . . . . . 96
6.5 Effect of increasing patient request traffic on response time in
the Heart Monitoring Application. . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.6 Effect of increasing mixed request traffic on response time in
the heart monitoring application. . . . . . . . . . . . . . . . . . . . . . . . . . 98
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Service</td>
</tr>
<tr>
<td>BP</td>
<td>Blood Pressure</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>DICOM</td>
<td>Digital Imaging and Communication in Medicine</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiogram</td>
</tr>
<tr>
<td>EEG</td>
<td>Electro-encephalogram</td>
</tr>
<tr>
<td>EHMS</td>
<td>E-Health Monitoring System</td>
</tr>
<tr>
<td>EHR</td>
<td>Electronic Health Record</td>
</tr>
<tr>
<td>EMG</td>
<td>Electro-myogram</td>
</tr>
<tr>
<td>EMR</td>
<td>Electronic Medical Record</td>
</tr>
<tr>
<td>ERDiagram</td>
<td>Entity Relationship Diagram</td>
</tr>
<tr>
<td>HAPI</td>
<td>HL7 Application Programming Interface</td>
</tr>
<tr>
<td>HIMS</td>
<td>Health Information and Management System</td>
</tr>
<tr>
<td>HIS</td>
<td>Hospital Information System</td>
</tr>
<tr>
<td>HL7</td>
<td>Health Level 7</td>
</tr>
</tbody>
</table>

xii
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ISR</td>
<td>Information Storage and Retrieval</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java Platform Enterprise Edition</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Messaging Service</td>
</tr>
<tr>
<td>JSON</td>
<td>Java Script Object Notation</td>
</tr>
<tr>
<td>LIS</td>
<td>Lab Information System</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PHM</td>
<td>Personal Health Monitoring</td>
</tr>
<tr>
<td>PHR</td>
<td>Personal Health Record</td>
</tr>
<tr>
<td>PHSF</td>
<td>Personal Health Service Framework</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RIS</td>
<td>Radiology Information System</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SEI</td>
<td>Service Endpoint Interface</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SOC</td>
<td>Service Oriented Computing</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description Discovery and Integration</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Indicator</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VRN</td>
<td>Virtual Remote Nursing</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WS</td>
<td>Web Service</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter presents the motivation for doing this research, the research problems that are targeted, and an overview of the solution.

1.1 Motivation

Health is one of the most important concerns in people’s lives. Computer science, networking and electrical engineering have changed the delivery of healthcare services through the concept of e-health. E-health is a promising field for improving healthcare quality by offering early symptom detection, early diagnosis, prevention, emergency case survival, and offering health monitoring either by patients themselves or healthcare providers [10][6][42].

One of the achievements of e-health is the use of electronic health measurement devices. Recently there has been increasing interest in wearable health
monitoring devices which can measure a variety of physiological parameters. Some of these devices have the capability of saving health data and transmitting it to health applications or remote health servers for future analysis and reports [26].

In real life situations, when patients are away from hospitals, they may not be able to measure their vital signs as much as they should. Health measurement devices are expected to play a role in alleviating this problem. Measurement devices can be used in e-health systems to gather health data and store it. The stored data can be analyzed and used to diagnose, advise treatment and find risky situations later or in real time. Health measurement devices also can be used in health systems including Personal Health Monitoring (PHM) and Remote Health Monitoring Systems to gather health data.

In this thesis, we propose a framework that can be used to build health related applications and systems that can work with personal portable health devices. This framework is based on a service-oriented architecture. Capabilities of SOA, coupled with a lack of general frameworks for developing e-health applications for personal health devices, motivate us for this research.

Currently existing personal health device-related software applications and systems are ad-hoc, however they have common features. In this thesis we are trying to unify these features to provide a general framework which offers reusable services and a common structure of the shared components of these software applications. Such a framework can be used to build e-health
applications for personal health devices.

1.2 Thesis Objectives and Research Problems

This thesis examines existing e-health systems, their subsystems, and the potential for using a service-oriented architecture to construct a flexible component-based framework to build e-health systems applications for personal health devices. Many personal health device related applications have been developed, most of which are ad-hoc systems that provide individual standalone solutions. They are not flexible, and they only offer an end-to-end infrastructure. Such systems are designed to offer pre-defined functionalities, and they do not have the ability to change or expand [34][25][9].

Existing e-health systems and applications for personal health devices have many common features from a technology point of view, and they have many reusable software components that can be shared, so they can share the same structure. The purpose of this thesis is to propose a general framework which can support building e-health applications. Shared components of these applications include:

- Patient subsystem: The components of this subsystem vary depending on the expected service level. The subsystem may include a variety of sensors to measure physiological data, one or more devices for collecting and transmitting data or simply an application for patients to insert health data.
• Infrastructure subsystem: Stores patient demographics, clinical data, and information that monitoring systems generate.

• Healthcare provider subsystem: Contains either mobile providers such as an ambulance or immobile providers such as hospitals.

Since the healthcare field and its demands are changing so quickly, offering health-related software and systems as a combination of web services gives them the flexibility to change or expand over time. A framework based on a service-oriented architecture can offer more flexibility and compatibility.

### 1.3 Thesis solution overview

In this research, we investigate a framework called Personal Health Service Framework (PHSF) based on a service-oriented architecture to provide the possibility to build efficient and distributed health-related software systems and applications. Health data can be gathered with mobile health measurement devices, and support multiple healthcare providers as well as multiple patients, to access and process the data. Patients who own a health device can register with the system, and use the personal device to measure and store health data. Later, a patient can give healthcare providers permissions to do tasks such as: accessing the patients’ health data, start/stop monitoring events for patients, and give patients some advice.

PHSF is a service-based framework and an open architecture for developing patient-centric personal health device-related applications and monitoring
systems. It consists of 7 components; each contains a set of web services. Each web service is responsible for a specific task. Services in each component have similar functionalities and closely followed goals. These services are integrated as an intermediary component, and together they cover the shared functionalities that are needed for building e-health systems and applications for personal health devices.

Figure 1.1: Component view of the PHSF.

Figure 1.1 depicts the overall architecture of the proposed framework. Provided PHSF services indirectly link patients with mobile health devices and providers, provide services to them, while storing all the information in the infrastructure. Different health systems and applications can be built by using different services including real-time or non-real-time, personal or hospital, and even emergency systems, of the framework.

Viewed from another perspective, PHSF contains 4 layers as depicted in
Figure 1.2. The first layer contains end-user services to be invoked from either the patient subsystem or the provider subsystem. The second layer contains intermediary application services which perform tasks for the upper layer. The third layer is data access control layer consisting of information storage and retrieval services, which are responsible for interacting with the database. The fourth layer contains the database.

![Layered view of the PHSF.](image)

1.4 Structure of thesis

The rest of this thesis is organized as follows. Chapter 2 provides the background study on e-health and its applications, as well as web services as enabling technologies. This chapter continues with an analysis of using service-oriented architecture to improve the quality of e-health services and how
e-health and web services are used in this thesis. This chapter also introduces related work that led us to design the PHSF.

Chapter 3 gives the architecture design of PHSF with a description of each component and the services it contains, as well as the relationships between these services. The end of this chapter features a dependency diagram between services. Chapter 4 details service implementation in the PHSF framework, containing both databases and web services.

As a case study, Chapter 5 contains an example application named the heart monitoring application that is built with PHSF structure and services. Chapter 6 first reports a scenario-based experiments as well as a scalability test experiments. This chapter also provides evaluation analysis for software quality of PHSF, and a comparison between PHSF and other e-health systems and applications related to personal health devices. Chapter 7 gives a summary of the thesis, contributions and future work.
Chapter 2

Background and Related Work

This chapter introduces concepts and technologies on e-health and web services that are the basis for designing the PHSF. This chapter also introduces some work related to PHSF.

2.1 E-health

In this thesis a framework is proposed for developing e-health software applications and systems that use personal electronic health devices to collect health data. The following provides an introduction to e-health concepts. E-health is the use of digital communication and information technologies to provide healthcare services [7]. Advances in pervasive and ubiquitous computing have given researchers an opportunity to use these technologies to resolve problems associated with healthcare, such as providing healthcare
services over the Internet, which plays an important role in the treatment of chronic diseases.

E-health contributes to improve access to efficient healthcare services, as a developing branch of medical informatics, and to address healthcare problems. These health services include personal medical care, medical advice, management of related data, communication between healthcare providers and patients, and remote monitoring [14][2].

Many patients can benefit from continuous monitoring because it helps in managing chronic conditions. Chronic diseases are becoming more widespread. For example, many international reports predict that in the near future about one in ten adults will have diabetes [46]. One of the most important ways to control these diseases is continuous monitoring of vital signs and analyzing them, and intervening in patients’ lifestyles depending on monitoring results. It also helps to find emergency situations by early detection of abnormal conditions [10][46][36].

One example of patients, who benefit from being remotely monitored, is a patient with heart disease who has had cardiac surgery. Real-time monitoring and continuous transmitting of data from a patient component to her healthcare providers after the patient is discharged can be helpful for recovery, and even for the patient’s survival, because any variation in signals can be detected immediately and alerts and treatment can be issued [47]. Providing a variety of health care services like this is possible through the e-health.
E-health is widespread in different fields including cardiology, radiology, rehabilitation, ophthalmology, psychiatry, surgery, diabetes, emergency medical services, and home care [22]. The extent of development in the electronic healthcare field will be reviewed in the following sections.

2.1.1 Personal Health Measurement Devices and Sensors

Personal Health monitoring devices and sensors are used to measure a variety of physiological parameters. In the following, a general description of these devices is presented. A major functionality for a personal health devices is for monitoring personal health. They may be designed to show test results to the patient, store data or have the capability to transmit data to health applications that reside on the patient’s smartphone, PC or PDA, or even transmit it to a remote health server or portals for further analysis and reports.

They may use body sensors close to or attached to a patient’s body to gather data (e.g., ECG and Accelerometer) or can be used by the patient as external devices for tests, such as a glucose meter. Examples of sensors include pulse oxygen saturation sensor, blood pressure (BP) sensor, serum glucose level sensor, electrocardiogram (ECG) sensor, electro-myogram (EMG) sensor, electro-encephalogram (EEG) sensor, temperature sensor, acceleration sensors, and a breathing sensor for monitoring respiration [26].
Because sensors are characterized by high heterogeneity, they may use different methods for measuring data which results in different kinds of data available to be stored in the database. To tackle this issue in developing concrete healthcare systems, interoperability among healthcare systems and domain standards such as Digital Imaging and Communications in Medicine (DICOM) and Health Level 7 International (HL7) are mandated [18][35].

2.1.2 E-health Applications and E-health Portals

Many e-health applications have been developed to help informing patients about their diseases, symptoms, and treatments. Some of them also help physicians to better diagnose. Existing e-health applications could be divided among the following categories: treatment, prevention, diagnosis or incidence detection management, nutrition, activity, medication, healthcare maintenance and checkups, reference, and emergency, short-term monitoring or home healthcare monitoring, long-term monitoring (nursing home) and personalized healthcare monitoring [50].

In some e-health applications patients have to insert health data by themselves and some can gather data automatically from devices in a wired or wireless way. Health applications can be installed on a patient’s device or may be available as a web application. Integrating these applications with personal health devices enhances the quality of these applications [10][46][23]. E-Health portals offer an integrated web application that provides users a unified interface to all the online health-related services such as reminders,
requesting consultations and referrals or receiving, provided by different medical organizations [30]. The WebHealthCenter portal is an example of such e-health portals [45]. It integrates various e-health applications and systems in the same place to share the resources [30].

2.1.3 Health Monitoring Systems

Using health measurement devices to measure health conditions, many e-health systems have been developed, including Personal Health Monitoring (PHM) and Remote Health Monitoring systems. Using personal health devices for gathering health information enhances the quality of these monitoring systems. The proposed framework in this thesis is to be used for developing health monitoring systems for personal health devices. In the following a general description of the concepts of health monitoring systems is introduced.

Using PHM, both patients and their relatives can measure and keep track of physiological signs and compare them with historical data so that they can control their health issues themselves. This capability would help individuals increase their independence and confidence, make better informed decisions, manage their diseases in a more practical way, and as a result, prevent disabilities [14][46][4].

Although personal health monitoring is a good start in e-health, the need for a patient to be monitored by specialists has spawned the idea of remote

monitoring, in which health data is retrieved from devices and transmitted to the providers' devices. Remote monitoring would enable physicians to monitor their patients in real time and provide a better diagnosis through accessing patients' medical records and current condition. Remote monitoring preserves the quality of traditional healthcare by providing geographically remote consultation and monitoring as well as providing pre-hospital care [27][51][26].

Several monitoring systems have been developed for the purpose of continuous monitoring which track the health state of patients from a distance. They allow physicians to offer health care at a distance through advanced electronic communications [55][25][7]. These systems use sensors, health devices, applications and communication technologies to capture and transmit vital signs to healthcare providers, and in some cases they can also provide feedback, online analysis and emergency detection services.

Health monitoring systems might measure multiple parameters simultaneously over a long term and are designed in a way that they can operate autonomously without requiring intervention and try to do not disturb the daily lives of the patients. Some of these systems are context-aware and help healthcare professionals to make a better decision based on patient's current conditions. They can be indoor, outdoor, in hospitals, nursing homes, assisted living, continuous or event-driven [28][50][43].

Many research projects have been conducted in developing good health monitoring systems such as AID-N (advanced health and disaster aid network),
MYOTEL, AMON and Health Gear [13]. TeleCardio is a continuous monitoring system for cardiac patients that uses mobile devices and wireless sensors for monitoring physiological signs such as heartbeat, blood pressure and ECG signals [21]. Health monitoring minimizes the time a patient spends in hospitals as well as reducing the number of readmissions for a patient with chronic health problems. It decreases the healthcare costs without reducing the quality of patient care [33].

There are many open issues and challenges in designing and developing such systems, including the lack of coverage and reliability of wireless and mobile networks, general limitations of handheld devices, privacy and security, management issues and network traffic of patients' health records which could cause end-to-end monitoring delay.

2.1.4 E-health Databases

In order to have a better diagnosis, current health status of patients should be compared with historical data to see the improvement or deterioration over time, so there should be a centralized database for e-health applications and systems to store and retrieve collected health data. The proposed framework supports storage and retrieval of personal health data that are collected from personal health devices into or from databases. Also, these personal health data can be transformed to the health level 7 (HL7) format which is a standard for health information to ensure interoperability. The
most important databases in e-health and a description for HL7 are explained in the following.

2.1.4.1 Personal Health Record

A personal health record (PHR), is a repository that maintains a complete and accurate summary of an individual patient's medical history which is accessible online. PHR can be drawn from various sources but is managed, shared, and controlled by patients. PHR is proposed as a strategy to make healthcare delivery increasingly patient-centered, as patients can view, edit, or discuss their own data including demographic information, health history and health data [48]. Being accessible by the Internet, PHR allows patients to retrieve health data anywhere and release it to others, such as pharmacists and therapists. Microsoft HealthVault \(^2\) is an example of such a PHR.

There are numerous benefits and barriers in using the PHR. PHR empowers patients to control their own health problems. It also enhances patient-provider relationships. Adoption, cost, privacy, and security issues are, however, more challenging [12].

2.1.4.2 Electronic Medical Record

An electronic medical record (EMR) is a storage of healthcare data in an electronic format managed and owned by an enterprise or institution such as a hospital. EMR contains health records for all patients of health enter-

\(^2\)https://www.healthvault.com/ca/en
prise systems or clinicians such as physicians, nurses and administrative staff. EMR is often integrated with other software such as billing and scheduling for that specific organization [16]. EMR improves the quality of patient care since it provides better access to health information for healthcare providers and ensures that data is accurate, appropriate and legible. It also decreases medical errors and eliminates lost data [52].

2.1.4.3 Electronic Health Record

EHR is a comprehensive, person-centered, and longitudinal collection of individuals' lifetime health data which lets health professionals access patients' health records that includes data from hospital information systems, community care clinics and other providers as well. Unlike EMR that belongs to a healthcare provider, EHR is a cross-in-situational repository [17][24]. One example of EHR is the Canadian Electronic Health Records developed in Canada. It is a territorial widespread database which stores patient lifetime data in a private and secure way. Canada Health Infoway has been mandated to implement EHRs in Canada. Each of Canada's provinces should have at least one or more EHRs system in place and all should follow the BLUE PRINT standards that are defined by Infoway to support interoperability [15].

There is a defined framework or architecture with some standards and all the regional and provincial EHRs should follow it so that they can be connected to each other and share patient health data including patient identity,
history of health and treatment across the country. The appropriate health information would be available for patient care no matter where providers or patients are [15].

Electronically accessing patients’ health data results in more efficient, faster and cheaper treatment compared to using paper-based data recording. Another benefit of having territorial EHRs is the possibility of data mining on the huge amount of available health data to track disease evolution, rehabilitation processes and effects of drug therapy which can help stop the spreading of diseases. For example, when several people have the same symptoms or the same disease over a small period of time from same locality, it will predict that the disease is spreading out in that local area, so immediate and suitable decisions can be made [47].

2.1.5 Health Level 7

The proposed PHSF framework supports transformation of personal health data collected from personal health devices to HL7 standard format to make personal health data understandable for other health applications. HL7 is also used to design PHSF’s database.

Increasing needs for exchanging and sharing data among clinical applications, and the nation wide demands for a centralized electronic health record, drive the requirements for a common language for e-health applications to ensure interoperability between them. Health Level 7 (HL7) standards is one of the proposed solutions. E-healthcare systems and applications that have
been adapted to the HL7 messaging standard, can communicate with each other even when they speak different languages. The HL7 standard is the most widely used messaging standard in the e-healthcare industry around the world [19][54].

HL7 is a framework with related standards that are used for managing health data exchange. HL7 creates flexible and cost effective approaches and guidelines with predefined logical formats for packaging healthcare data in the form of messages to be transmitted between computer systems. These efforts enable effective, efficient communication between the healthcare communities. HL7 attempts to establish flexible and worldwide standards that loosely define representation and movement of clinical data between different health systems such as a radiology information system (RIS), lab information system (LIS), hospital information system (HIS), and electronic medical record (EMR). In many cases, however, each of these systems speaks their own language [20].

In healthcare systems, information about each particular event such as a patient admission, is sent through an HL7 message [19]. Some systems don’t know how to speak the language of HL7 messages and require a translator. HL7 interface engines speak the language of HL7 and work as an interpreter for existing applications [6]. *Mirth Connect* ³ and *IGUANA* ⁴ are two examples of such an interface engine that translate HL7 messages.

³http://www.mirthcorp.com/.
2.1.6 E-health Contributions to Healthcare

E-health has improved and speeded up the quality of healthcare services by decreasing the workload for healthcare providers and freeing them from routine patient check-ups and visits. It allows them to concentrate on more difficult tasks, since it offers remote monitoring and self-managing health systems and applications [3][2][37].

A recent survey in the U.S. shows that almost 75 percent of primary care physicians indicated that errors have been reduced through the use of e-health applications, 70 percent said that their productivity has been increased by using IT services, and over 60 percent indicated that the cost of health care has been decreased with IT tools [26].

The proposed framework in this thesis provides an enhancement to personal e-health systems and applications, and as a result could enhance healthcare in general.

2.2 Service Computing

The architecture design and implementation of the proposed framework is based on the service computing model. The service-oriented computing (SOC) paradigm assembles services components into a network of services and utilizes them as elements of rapid and low-cost development of agile, interpretable and distributed applications. Performing business services more efficiently through the use of computing technology and IT services is the
goal of SOC. SOC bridges the gap between business services and IT services. It relies on the Service Oriented Architecture (SOA) for building a service model [41][40]. In the following section, web services as an enabling technology for developing PHSF are introduced.

2.2.1 Web Services

A web service is a unit of managed code with a public interface available via a network such as the Internet, performing a special task on behalf of other applications and users. These separated units can be accessed by URLs so they enable both applications and users to remotely invoke the functionality they provide using an HTTP request. Being available at a particular endpoint in the network, services perform invoked functions by receiving request messages and sending response messages according to their service specifications [39].

Since web services provide their business logic and data processing through programmatic interfaces, service consumers do not have any knowledge about their internal modules and operations [30][49]. Offering such an abstraction, web services promote the idea of making larger, more flexible integrated systems by combining network-available and independent modules into a network of services [11][39][1].

Web services are different from web pages. Web pages can be accessed by humans only, but web services can be accessed by humans and other automated applications so they do not provide users with a GUI, but they can be added
by developers into web pages or applications with a GUI [30][39]. They define business objects that execute a unit of work for a customer, either another application or a user, and wait for another request. The following are some advantages of using web services:

**Loosely coupled:** Using web services, loosely coupled services can work together to form a more complex system while each of them works independently without knowing about other's internal functionalities. So each part of the system can be modified and extended separately as a service, without affecting others.

**Ease of integration:** Web services can integrate different units of work provided as a piece of code and act as glue in large systems.

**Service reuse:** Through the use of web services a specific functionality can be implemented as a separate service and be used many times in different applications.

**Enhance total quality:** Web services can be used to increase the total quality of the system through rapid and low cost system development by breaking up the integration logic into distinct, easily manageable pieces [41][30].

There are also disadvantages associated with web services. Compared to transmitting data using binary codes, web services are not efficient, since they use an XML format to transmit all the data. Also, dynamically reconfigurable runtime architectures, validating the security aspects in SOA-based applications, lack of versatility and semantically enhanced service discovery are other challenges in this area.
2.2.1.1 Service Oriented Architecture

Service-oriented architecture (SOA) is a conceptual model and a design framework for loosely coupled web services that may be distributed across platforms, technologies, and physical topologies. SOA enables software applications running on different machines to communicate and exchange data using XML messages, regardless of programming languages, platforms, and protocols in which services are written, performed, or used.

SOA unifies business processes by structuring large software applications as a collection of smaller modules known as services, and brings the benefits of component-based thinking to them [38]. These smaller modules perform functions ranging from answering simple requests to executing complicated business processes.

Using SOA, any piece of code and any application component can be reused and transformed into a network-available service. The key desired attributes of SOA are being interoperable, componentized, composable, message-based, distributable and discoverable [41].

As depicted in Figure 2.1, SOA consists of three parts. Service registry, service provider, and service requester. Service registry is a searchable directory that provides service descriptions in the WSDL language. It offers a central place for publishing new services and finding the existing ones. It acts as a broker for web services.

A service provider implements and publishes one or more web services. The service provider is also responsible for registering new services and provid-
ing information about new services on the registry in order to make them available on the network to be used by other users or applications.

A service requester uses web services by requesting service providers to perform certain services and, as a service response, receiving results of requested services. A requester searches the service registry to find a desired service, binds it and utilizes it by sending an XML request over the network connection. Figure 2.1 depicts the architecture of web services.

![Figure 2.1: Web services architecture.](image)

The information provided in the service repository can be categorized as follows. Business information contains information about the service provider and the implementer, service information contains data about the nature of the web service, and technical information contains data about how the service is implemented and how it should be invoked. The following are technology standards being used in SOA.

**UDDI** stands for universal description, discovery and integration. It is a platform-independent and XML-based registry that is used to register and
describe available services, and to find web services which meet certain requirements [49][39].

**XML** stands for Extensible Markup Language. It is a format for exchanging structured data between two parties. The data provided in XML format is significantly structured by tags or elements in hierarchies [8][5].

**WSDL** stands for web service description language. It is an XML-based language for describing a web service and all the information needed to invoke and communicate with it, including two types of descriptions. It provides functional description that focuses on the details of the message syntax, and how to configure the network protocols to deliver and define details of how and where the web service is invoked. It also provides non-functional descriptions of the services such as a security policy [5][11][39].

**SOAP** stands for simple object access protocol. It is a W3C accepted protocol for exchanging XML-based messages over the network. SOAP specifies the format of requests to the servers and responses to the requesters. Although other service invocation languages can be used, SOAP is by far the most popular choice for web services [5][39].

A SOAP message is usually transmitted from a services requester to a service provider as follows: It starts when the SOAP client or service requester creates and sends a SOAP message to an ultimate SOAP server via HTTP or HTTPS. The validity of the request is checked by the server with the help of the SOAP processor. If the request is valid, it invokes the requested web service on the server. Then the response for that request will be sent back
to the client after the service processes the request and wraps it in a SOAP message.

A SOAP envelope, as the SOAP message format, encloses all requests and responses. The following are the parts of a SOAP envelope where envelope is the root element containing the following [5][39]:

- Header is an element for any optional attributes of the message being used in message processing.

- Body is a mandatory element being used to describe exceptional situations.

- Fault is an optional element that provides information about errors that occurred when the message is being processed.

2.2.1.2 Web Services in E-Health

Providing cost effective and high quality systems has moved healthcare towards IT solutions. Healthcare, however, has an extremely decentralized and fractured nature, so a single IT system will not serve all its needs. Web services have been attracting a lot of attention in the field of e-health since they allow heterogeneous systems integration and offer a generic model for implementing large scale enterprise applications. By doing so, SOA hides the complexity of distributed health services.

Using SOA, a variety of medical health services can be used for different tasks such as diagnosis, treatment, health data transmission for follow up with a
patient, connecting healthcare providers and patients as a bridge, and data assessment to find emergency situations [3]. Many health applications can be developed using different combinations of these services. They can also be used to enhance and extend existing e-health applications.

Another benefit of leveraging web services in the e-health field is that they are platform independent since they use open standards, such as XML, SOAP, WSDL and UDDI [38]. Being interoperable, SOA allows a variety of devices to be used for measuring, communicating, sending and receiving health data. Web services offer better flexibility and extensibility through reusable software modules and also allow one service to be replaced with another without effecting the whole system. For these reasons SOA is a perfect fit for healthcare [34] [38].

Healthcare is a growing field and lots of changes are being applied to it every day. In SOA, an individual service is free to change without affecting how the service is consumed. Thus, SOA helps the healthcare industry to develop cost efficient and dependable healthcare services and improves the standard of living, and reduce the complexity of integration and application development [34].

2.3 Related work

This section introduces several proposed e-health architectures and systems which are based on service-oriented architecture and/or cloud computing.
The idea behind these works has guided us to investigate the design and implementation of PHSF.

2.3.1 Virtual Remote Nursing System (VRN)

In this work a virtual nurse has been suggested which is an agent that is wrapped as a web service and acts like a remote full-time nurse [32]. VRN integrates mobile medical devices with ePHR systems. ePHR is a repository in the cloud to store health information, in order to obtain patients' health status updated continuously and automatically. This virtual nurse is installed on a patient's personal computer or smart phone and manages the patient's health condition continuously. Healthcare providers can assign different tasks to this agent. It performs these tasks by accessing a user profile as well as health information in the ePHR. Figure 2.2 presents a view of the VRN.

The VRN system consists of the following components. A Personal Health Record System (ePHR) which is used to save and retrieve health information. A Healthcare Provider System which is used by healthcare providers to view patients' health information, access the vNurse to assign a task, and observe the result of the assigned tasks.
Figure 2.2: High level view of Virtual Nursing System (VRN) from ([32]).
A *Personal Health Application* is a component which receives task results such as recommendations, warnings and reminders that are generated by the vNurse to inform the patient. A *Virtual Nurse* is modeled as a software agent wrapped into a web service. It receives a task request such as generating a report on certain patient health data, performs the task by using health information from ePHR, and sends the task response to the healthcare providers.

VRN handles emergency situations by sending timely information about the patient’s health status tagged with a risk flag, to the medical practitioner and call emergency centers if it is needed. Emergency centers also can assign tasks to the vNurse such as calling patient emergency contacts if the emergency situation is confirmed, and sending the required information about the patient to the emergency center before the patient arrives.

All the functionality in the VRN is provided in one service installed on the patient’s PC or smartphone, without flexibility and extensibility. The VRN provides a connection link between patients and healthcare providers, however it does not provide self-monitoring. Health data transformation to the Electronic Medical Record (EMR) that enables a medical practitioner to access patient medical data is provided in VRN, but as an application layer not as a service.

The focus of VRN is on developing a service which handles some tasks to be done on the health information, and returning results. The ability to communicate with health measurement devices (e.g. activate them, gather
data from them) are missing in vNurse. A healthcare provider defines a task to the vNurse to remind a patient to measure a specific health attribute in a specified interval, and save it to the database. Then, the healthcare provider can access health information and monitor the patient. Thus, automated monitoring is not supported in this system.

2.3.2 A Cloud Computing Solution for Patient Data Collection in Health Care Institutions

This work focuses on the problem of collecting patients’ vital data and delivering this information to the medical center’s cloud for storage and enabling remote access to this data by medical staff [44]. Platform as a service (PaaS) is used to deploy some standard services to promote reusability, and software as a service (SaaS) is used to allow other healthcare institutions to use these services.

Figure 2.3 depicts the proposed architecture in 4 parts. A sensor module consists of sensors with software installed on them to collect, encode, and transmit data. An exchange service module is responsible for dispatching the collected health data to the appropriate storage service hosted on the cloud, and acts like a broker between local and remote services. A cloud services module contains services responsible for storing collected data, retrieving it, and provides a platform for development, testing and deployment of applications needed by medical staff. A content services module is a com-
Figure 2.3: Patient’s data collection in health care institutions from ([41]).

A common interface that acts as a gateway to allow users to access all available information and applications.

This system supports what is lacking in the vNurse system [32], by providing a solution for automating data collection through integrating health measurement devices and cloud computing while using SOA. This system, however, does not provide a solution for monitoring remote patients, self-monitoring, and emergency control.

### 2.3.3 E-Health Support Services Based on SOA

Omar and Taleb-Bendiab [37] present an e-health monitoring architecture named e-health monitoring system (EHMS) that is based on a service-oriented architecture. EHMS transmits health data gathered from health sensors to
hospital applications. All the tasks that make monitoring possible are being performed through the provided services which are grouped into different units. Figure 2.4 depicts the EHMS and its services. Using SOA and grouping services into components makes the EHMS flexible and extensible. Grid computing is also used in this architecture since it offers services on a large scale enterprise-class system.

![Figure 2.4: EHMS monitoring approach (from[37]).](image)

Using EHMS, remote monitoring systems can be built to allow healthcare providers in hospitals to make requests to monitor a patient remotely on a specific health attribute during a special interval. Using such a monitoring system, a hospital can give patients health measurement devices to measure their health attributes, either in the hospital or in their homes. This framework also provides emergency support by analyzing health records, labeling them with a risk flag, and calling an emergency station or relatives
depending on the degree of the risk. The labeling is done by comparing a health record with the boundaries that healthcare providers have specified in certain rules for a normal range of the monitored attributes.

EHMS is used to develop e-health systems for hospitals. Applications that can be built on the EHMS are limited to remote monitoring for hospitals, but self-monitoring is not possible. In EHMS, health data is stored in the patient's log file inside the logger in the monitoring system unit, and there is no structured database for health data. A transformation component is also lacking in the EHMS to transform health data to a format that is understandable for healthcare providers and vice versa.

2.3.4 Cloud Computing Aware Ubiquitous Health Care System

In this work [42] an automated health care system named Health Information and Management System (HIMS) is proposed that monitors basic health parameters and stores them in the cloud, using a service-oriented architecture. This health data can be retrieved by either patients or physicians via web services hosted in the cloud in real time. Figure 2.5 shows the HIMS.
Figure 2.5: General Architecture of the HIMS [42]
The HIMS is a globally distributed system and contains different components that interact with each other as well as interacting with the central database server. Communication between these subsystems are through web services. All these web services and the central database are hosted in the cloud.

The HIMS contains three parts. The *U-health care centers* consist of health monitoring devices. The *Health centers and Hospitals* consist of health professionals and medical equipment. *HIMS mentoring* contains a central database as well as web services such as save/retrieve data services and communication services that are hosted in the cloud. This database contains patients' personal information and results coming from tests.

In HIMS, patients use health test devices to measure health signs, and the results are automatically saved in the central database in the cloud. Patients also can insert their daily activities. Physicians and health experts can access the health records and activity logs to advise the patients. Thus, this system provides health services based on both health data and healthcare activity history. These services include diagnosis, treatment, feedback and counseling services, as well as a service for viewing progress and health status of a patient in comparison to demographic norms. The HIMS also provides emergency support by comparing a patient's health status to the standard values of health attributes. It does not, however, provide a self-monitoring service.
Chapter 3

Framework Design

PHSF is an open architecture for building e-health systems and applications in which mobile personal health devices are used to measure health attributes. It provides various services that can be integrated to form different e-health systems and applications for delivering personal health monitoring and care. Services in this framework are grouped in different components based on their functionality, so all the services in one component are for almost the same goals. Each service in a component has a specific task that contributes to the overall framework. From another point of view, services in this framework are organized in four separate layers as shown in Figure 1.3.
3.1 Service description

Figure 3.1 shows a detailed view of the PHSF layers, their services and the relationships between services in different layers. There are four layers in this framework. The first layer includes front-end services that are used by end users, which can be either external web services or applications. The PHSF provides support for both healthcare providers (such as physicians, nurses, and other healthcare professionals) as well as individual patients.

The second layer includes intermediate services which are doing tasks such as risk assessment or data transformation to support the front-end services.

The third layer is the data access layer, which includes services to access the databases. Services in this layer are used to prevent other services from accessing the database directly. This layer contains five parts. Four of them are used to access each database, and the services in the cross DB access are used when data should be shared between these databases. The fourth layer is the database itself which is provided as four separated databases.
Figure 3.1: PHSF layers.
From another point of view, the PHSF consists of seven components. As shown in Figure 3.2, each component contains different web services. The following sections describe these components and their services.

Figure 3.2: PHSF components.
3.1.1 Device Client

The device client services component includes services for managing, manipulating, and communicating with health measurement devices.

**Device Adjustment Service:** This service is aimed to calibrate a device. It is invoked if the data obtained from the measurement device is not normal by checking against certain constraints or limitations which can be set at device registration time. When a user registers a device, she can set the expected normal reading range for that device. Intelligent approaches also can be used to find if data is out of range, by comparing reported health data with health data that has been saved before.

**Device Reporting Services:** These are services for providing reports on the registered devices. There would be different services for retrieving device information based on some desired attributes. For example, there is a service for retrieving all the devices that have been registered for a specific patient. This service gets patient ID as an input and returns an object that contains the information about the target devices. Another example is a service for retrieving all the devices that measure a specific health attribute such as blood glucose.

**Monitor Scheduling Services:** This services are for start and stop monitoring registered devices. The start monitoring service gets and analyzes requests to start monitoring and schedules the monitoring to be repeated in certain interval. After parsing the request and finding the state of all the desired devices, it will activate them and gather data from them based on
the duration and attributes given in the requests. The input for this service is a request for monitoring a registered device. The request contains the patient ID, the device and the attributes that are desired to be measured, as well as the duration and intervals that data gathering should be repeated. This service then returns an object which contains all the information about current monitoring sessions.

The stop monitor scheduling service is for terminating a monitoring session. This service takes an ID for the target session, ends the process of monitoring, and returns the stopped session’s ID along with its attributes.

3.1.2 Patient Application

This component contains services that are responsible for communicating with and providing services to patients. The services in this component are more analytic services as follows.

Risk Analysis Service: This service determines whether a patient’s situation is risky or not from the health records. The risk analysis service analyzes the health data based on the schema rules and risk boundaries that have been defined previously by the patient, for a specific health attribute. This service compares the result of a medical test and the values for that attribute in the schema rule, and reports any suspicious situations. The input to this service would be the health record and the output would be an object indicating the degree of risk, e.g. warning, emergency or healthy, for the targeted attribute. In high-risk situations, this service uses Emergency Call Service to find the
nearest emergency provider, and sends emergency messages to it and the physicians that has designated by a patient for receiving warning.

**Emergency Call Service:** This service is used by the risk analysis service in emergency situations. It gets the location of the patient, and uses the finds closest provider to find the closest provider to this location.

**Reminder Services:** These services are used to remind a patient to perform the tasks that have been given to the patient. These services are similar to monitoring services for starting and stopping reminders. Start reminder service takes a reminding request containing the reminding purpose and the interval that the reminding task should be performed as an input, and starts a reminder session for the patient. Stop reminder service is used to stop a reminding session.

**Patient Reporting Services:** This services are used to provide reports on patients. A patient reporting service takes a request for providing the report as an input, uses the services provided in the information storage and retrieval component for retrieving the patient’s information, and provides a report based on the retrieved information.

### 3.1.3 Provider Client

This component contains all services responsible for managing healthcare providers’ interactions, including saving, retrieving and manipulating basic information about healthcare providers. The following are services in this component.
Provider Reporting Services: These services are used for making reports based on retrieved information about healthcare providers which could include reports of the providers in one region or reports of the providers that have been involved with a particular patient.

Find Closest Provider Service: This service is responsible for finding the closest healthcare provider to the current location of a patient. This service first retrieves all providers’ locations from the database, compares them to the patient’s location and finds the closest one. This service takes a patient’s current location as an input object and returns an object containing all information about the closest providers.

3.1.4 Provider Application

This component contains services to serve healthcare providers’ requests through the access to the information storage and retrieval services. Services in this component include the following.

Risk Analysis Service: This service is for finding risky situations by comparing the value of a health attribute in a health record and value of that attribute in the schema rules and risk boundaries. These rules are defined by the provider previously. The input to this service would be a health record and the provider ID, and the output of the service would be an object that indicates the degree of risk for the specified attribute that is compared with the schema rules defined by the provider. In high-risk situations, this service uses Emergency Call Service to find the nearest emergency provider, and
send a warning message to the health provider and patient. In low risks, it sends warning message to the provider and the patient.

3.1.5 Information Storage and Retrieval

The information storage and retrieval component, which corresponds to the data access layer, contains services for accessing the databases to prevent services of other components from having a direct access to the database. This component adds a significant amount of robustness to the whole system. In the following, some of these services are presented.

**Save Health Records Service:** The service takes an object containing health record’s properties, inserts the received health record in the proper database, and returns the newly added health record and its ID.

**Retrieve Health Records Services:** These services are used to provide health records for both patients and providers by accessing the health database. The resulting report on health data could be different based on the incoming requests in which all the filtering parameters are specified. The outputs of the services would be the retrieved health records.

**Save Patient Service:** In order to register a new patient in the database, this service is used which takes the required parameters for a new patient including both basic and authentication information, and returns an object containing the same information plus the ID of the newly added patient.

**Retrieve Patient Services:** These services are used to retrieve patient demographics from the database. The result of these services might be different
based on different inputs. An example could be the information about all the patients older than a certain age, or information about patients who are living in a certain region. These services get objects containing the desired filtering parameters as input, and return objects containing desired patients' information.

**Save Schema Rule Service:** This service is used to add a new bounding rule for a health attribute. Such a rule is used to find risky situations. This service gets an attribute and all the restrictions for that attribute either for emergency or warning situations as input and returns an object containing the same information plus the rule ID.

**Retrieve Schema Rule Service:** This service is used to retrieve a schema rule for an attribute which shows the limited boundaries for the value of the specified attribute.

**Save Instruction Service:** This service is used to add a new advice to a patient by a provider. The service takes an object containing the patient ID, the provider ID, and the instruction message itself as inputs, saves it in the database, and returns an object containing the same information plus an advice ID.

**Retrieve Instruction Services:** These services get desired parameters for searching among existing advice records in the database, and returns the resulting advice and their properties filtered by the parameters given in the inputs.

**Save Provider Service:** This service is used to save a new provider. It
gets all the attributes or properties of the new provider and returns an object containing the properties of the newly added provider and its ID.

**Retrieve Provider Services**: These services are used to retrieve providers from the database. The services get objects that contain the desired filtering parameters as inputs and return the resulting providers' information.

**Save Device Service**: This service is used to register a new device in the database. It gets the parameters for the new device and returns an object containing the same information plus the ID of the newly added device.

**Retrieve Device Services**: These services are for retrieving devices from the database. It gets the desired parameters that are used for narrowing the search result as inputs, and returns information about the devices matching the parameters.

**Save Activity Service**: This service saves a new activity log for a patient. It gets an object containing a patient ID, the activity and its properties such as the date as an input, and returns an object containing the same information plus the ID of the newly added activity.

**Retrieve Activity Services**: These services are used to retrieve activities based on parameters of received requests. Example requests include getting all the activities for a patient which have been inserted on a specific date, or requesting all activities for patients that are monitored by a particular provider.

**Assign Device Service**: This service is used for assigning a new measurement device to a patient. Assign device service gets the patient ID for
whom this device is being assigned and a device ID, and returns an object containing all the properties of the newly assigned device.

**Revoke Device Service:** This service is used for revoking an assigned device from a patient. A revoke device service gets the patient ID and the device ID, and changes the status of the assigned device for the specified patient to false.

### 3.1.6 Data Transformation

Health Level 7 (HL7) is the most widely used messaging standard which is used as a common language between e-health applications. It enables e-health applications to communicate with each other. Health data in the PHSF might have various formats. To insure interoperability, there should be services for transforming health data from what is retrieved from the information storage and retrieval component to the HL7 format and vice versa. This transformation component integrates the PHSF and various health applications which speak different messaging languages, makes the health data understandable for those applications, and makes communications between those applications possible. There are 2 major services in this component.

**Parse HL7 Message Service:** This service can be used to parse a health record in the HL7 format, so health data can be stored in PHSF's information storage and retrieval services. This service takes health data in the HL7 format, transforms it to other formats such as XML, and returns health data with the new format.
**Build HL7 Message Service:** This service is used to create a HL7 message to make the health data understandable for external e-health applications. This service takes health data in other formats such as XML and returns health data in the format of the HL7.

### 3.1.7 Privacy and Security

This component contains services to make sure that patients’ personal health information can only be accessed by the proper and authorized patients and healthcare providers.

**Authentication Service:** This service checks the user credentials to determine if this is a valid user or not. This service takes a username and password, and returns a Boolean variable showing if the user passes authentication or not.

**Authorization Service:** This service checks if a healthcare provider is authorized to access a patient’s health information or not. It gets a provider’s ID and a patient’s ID as inputs and returns a Boolean variable which indicates the authorization status of the provider.

**Authorize Provider Service:** This service is used by a patient to authorize a healthcare provider to access his/her health information stored in PHSF. It gets the patient and provider IDs as inputs.
3.2 Device Interfaces

The following subsections describe the required interfaces that are expected to be implemented on devices to work with the PHSF framework.

3.2.1 Patient Device Interfaces

All the following interfaces are expected to be provided by personal health measurement devices.

**Activate Device:** The expected functionality for this interface is to activate the device and make it ready for tests. It receives a request to activate a device. One potential use of this service is for PHSF’s monitoring scheduling service that gets a request for monitoring a patient, finds the appropriate device, activates it, and measures the health attribute in given intervals.

**Deactivate Device:** This interface is used by PHSF services to deactivate a device.

**Gather Data:** This interface is for gathering health attributes’ values from a device. The expected input for this interface is the name of the attribute that is going to be measured. The monitor scheduling service is one of the services in the PHSF that uses this interface.

**Calibration:** This interface is for calibrating a device requested by PHSF’s calibration service when data measured by the device is not normal.

**Status:** The expected functionality for this interface is to return the status of the measurement device stating whether it is active or not. It gets a
request to identify the device status and returns an object containing the status of the device.

**Alarm:** This interface is used to send a notification to the patient who owns the device. The reminder service of PHSF is one of the services that use this interface.

**Retrieve Location:** This interface is used to retrieve a patient’s geographical location in an emergency situation. This location is used later to find the closest healthcare providers by PHSF services.

### 3.2.2 Provider Device Interfaces

This interface is expected to be provided by healthcare providers’ devices. The alarm interface is one of the provider interfaces which is used to call or inform a provider through her/his device in case of emergency. The find closest provider service uses this interface.

### 3.3 Services Relationships and Dependencies

In this section the relationships between services in the PHSF are presented. Although the framework has been designed in such a way that all functionalities operate separately through web services, relationships between them are inevitable and in some cases they have to invoke others to complete their functions. The following are the relations between services in the PHSF framework.
None of the services in the framework has direct access to the databases except the services in the ISR (Information Storage and Retrieval) component. To communicate with the databases, other services must use the services provided in the ISR component. For this reason the following pair of services are related to each other.

In the device client component, the device reporting service needs to use the retrieve device service in the ISR component to access information about devices and report on them. In the patient application component, the risk analysis service uses the retrieve schema rule service to retrieve the valid boundaries for a health attribute's value. The patient reporting service also needs to invoke the retrieve patient service to access patient information and report on them. In the provider client component, the find closest provider service has a relation with the retrieve provider service to access all authorized providers' addresses. The provider reporting services also need to use the retrieve provider services in order to access providers' information to report on them.

In the provider application component, the risk analysis service has relation with the retrieve schema rule service in order to retrieve boundaries and restrictions defined by a specific provider for health attribute values to find either warning or emergency situations.

Another type of service relations in the PHSF is between internal services above the data access layer. The risk analysis service in the patient application services component uses the get authorized providers service in the
security and privacy services component to get all the authorized providers, and then notify them that the patient is at risk. In order for a provider to use some services in the framework such as manipulating, saving and retrieving health records, first the provider’s identity should be checked through the authorization service in the security and privacy services component, so all the services that are used by providers have a relationship with the authorization service.

The following is a dependency graph of PHSF services. It is a directed graph representing dependencies of services upon each other. Some services do not operate without the existence of other services. Figure 3.3 shows the dependency diagram for the PHSF framework.

![Service dependency diagram for PHSF.](image)

Figure 3.3: Service dependency diagram for PHSF.

For example, in order to add a device for a patient, the patient has to be registered in the system so the service for registering a new device depends
on a service for registering a patient. In other words, this diagram shows the order in which services should be called during operation of PHSF.
Chapter 4

Framework Implementation

4.1 Web Service Implementation

Many programming or scripting languages can be used for implementing an e-health system using service-oriented architecture. Among these languages, Java has been used for implementing the PHSF services using the J2EE platform and the Spring framework. The main reason for choosing Java is because it can be deployed on both small wireless devices and powerful server computers.

Choosing a proper web service development framework is also a challenging task. There are many frameworks for developing web services such as Axis1, Axis2, Apache CFX, GlassFish Metro, Glue, JBossWS, and so on [5]. The purpose of these web service frameworks is to make building, deploying, and publishing of web services simpler and more convenient through a robust
infrastructure.

Among the above listed frameworks, Apache CFX was used for developing services in PHSF. Apache CXF provides an open source, fully featured, easy-to-use web service framework. It provides a simplified programming and flexible deployment model for developing SOAP and RESTful web services, built on the Apache Tomcat server. It also supports various application protocols like HTTP or JMS [5].

All PHSF services have been implemented, except two services in the transformation services component, the parse HL7 messages service and the build HL7 messages service. These two services can be implemented by using Java libraries for developers such as jL7 1 and HAPI 2 to create HL7 messages from the database or from an XML message, and to parse HL7 messages.

4.1.1 Approaches to Web Service Development

There are two approaches that are widely used for developing web services, SOAP (Simple Object Access Protocol) and the REST (Representational State Transfer) architecture style. SOAP is a protocol for exchanging XML-based messages over a network, typically using the HTTP protocol. The SOAP message format is comprised of a SOAP envelope containing a header and a body [5].

In SOAP-based web services, WSDL files are published by the web service

---

1 http://benohead.com/jl7-hl7-library-for-java/
2 http://hl7api.sourceforge.net/devbyexample.html
provider as a contract between the web service provider and the consumer. The WSDL file of a web service can be found over the internet either by looking up in the registry or directly from the server in which the service is deployed. One of the usages of the WSDL file is for service consumers to generate a web service client code using the interaction tools offered by the web service framework [5].

On the other hand REST (Representational State Transfer) is an architectural style that provides a set of guidelines for providing and consuming resources over the web. In RESTful web services there is no formal contract between the service provider and the service consumer. Using the REST architecture style, a resource can be identified and accessed through a Uniform Resource Indicator (URI). Using REST, the only thing that consumers need to know are the supported operations by the service provider and the format of the message in XML or JSON (Java Script Object Notation) [5].

Using either SOAP or REST for developing web services depends on the application requirements. When the web service is complex or prone to change and there are many rules that the web service consumers and providers need to follow for either consuming or providing web services in the form of contracts and negotiations, SOAP would better fit the development process. These contracts can be a WSDL file or WS specifications such as web service security specification. For transmitting and receiving simple XML messages, RESTful web services are a better choice. As the PHSF must accomplish many different interrelating services, SOAP is the chosen approach for the
PHSF.

Using the CXF framework, there are 2 ways of developing web services; either Bottom Up (code-first), or Top Down (contract-first). Using the code-first approach, which is used to develop the PHSF services, all the functionalities are created in a Java class and then converted into web service components using the corresponding tools. To achieve this, the following steps were taken:

1. Create a Service Endpoint Interface (SEI) which defines all the variables and methods that should be implemented in the web service. The web service will implement this interface. The code in Figure 4.1 shows the implementation of the retrieve health record service for the healthcare provider in the PHSF as an example interface.

   ```java
   @WebService
   public interface healthRecordRetrieve_PortType {
       public String Retriev_HR_Provider(
           int arg_provider_id);
   }
   
   Figure 4.1: Implementation of retrieve health record service as an example interface.
   
   2. Create the class that implements the service endpoint interface and annotates it as a web service. The lines of code depicted in Figure 4.2 show the service for retrieving health records in PHSF.

   3. Create a beans.xml file and define a Spring bean for this service, using the JAX-WS frontend. Beans.xml is used to publish web service endpoints.
as Spring-based configuration files. The file contains Java beans and their dependencies. Such configuration files make the development of a web service convenient and easy with CXF. The code depicted in Figure 4.3 shows part of a bean.xml file in PHSF.

Each ‘jaxws:endpoint’ element in the beans.xml configuration file specifies a web service as a JAX-WS endpoint. The term jaxws shows that JAX-WS frontend is used internally to publish the web service. In each endpoint element, id is a unique identifier for a bean and the actual implementation class for the web service is specified in the implementer section. The address specifies the endpoint’s published URL address.

4. Develop a client by invoking the web service. Example code is shown Figure 4.4.

5. Modify the web.xml file which contains the web application configurations. CXFServlet is a front-runner component that initiates how the CXF environment should be defined, and how Spring and CXF should be wired together. Also, for loading the server-side configuration beans.xml file, the
6. Create a WSDL file for the web service. As mentioned earlier, in SOAP-based web service development, a WSDL file can be used as a contract between the web service provider and the consumer. WSDL files typically can be created by the tools that the web service framework provides. In a WSDL file, web services are described as a set of communication endpoints which are called ports, as well as how the services are bound to a messaging protocol like SOAP [5].

Request and response messages in SOAP are encapsulated in an envelope.
QName serviceName = new QName
    ("http://ISR/", "healthRecordRetrieveService");
QName portName = new QName
    ("http://ISR/", "healthRecordRetrievePort");
Service service = Service.create(serviceName);
    service.addPort(portName,
        SOAPBinding.SOAP11HTTP_BINDING,
        "http://localhost:8080/PHSFproject/
            services/healthRecordRetrievePort");
ISR.healthRecordRetrieve_PortType client =
    service.getPort(portName,
        ISR.healthRecordRetrieve_PortType.class);
client.Retrieve_HR_Provider(l);

Figure 4.4: Example of invoking a web service and developing a client.

As an example, Figure 4.5 depicts the SOAP request for retrieving health
records for the patient with ID equal to 1.

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    <soap:Body>
        <ns2:Retrieve_Health_Record_Patient xmlns:ns2="http://ISR/"
            <patientID>1</patientID>
        </ns2:Retrieve_Health_Record_Patient>
    </soap:Body>
</soap:Envelope>

Figure 4.5: SOAP request for Retrieve Health Record for patientID=1.

Figure 4.6 shows the response SOAP message for the above request which
returns a string object that shows the date, the attribute and the value for
a specific record.
Figure 4.6: SOAP response for Retrieve Health Record for patientID=1.

4.2 Database implementation

MySql is the world’s most used open source relational database management system (RDBMS) [53] and is the database system selected for the PHSF. Data storage in the PHSF is organized in four databases. Each of these databases can be located on different distributed servers.

All of these four databases are service-based which means they cannot be accessed directly. The services in the data access layer are used to interact with the databases. Protecting a database against external access helps to improve security and also makes them easier to use. The data access layer, shown in blue in Figure 4.7, consists of 5 sub-components.

In Figure 4.7, the Health DB Access includes services that interact with the health record database; Provider DB Access includes services that communicate with the provider database. Services that interact with the device database reside in the Device DB Access and the ones that interact with the
security database are located in the *Security DB Access*. The last component in this layer is the *Cross DB Access* which correlates these databases together, in case one of them needs to access the other ones.

![Figure 4.7: The PHSF data access layer.](image)

The following subsections describe these four databases and their schemas in ER (Entity-Relationship) diagrams.

### 4.2.1 Health Records Database

The health records database is the main database in the PHSF. It is considered to be the most important part of the data storage because the main functionality of the framework is based on patient health information and this framework is meant to be patient centric. It contains sensitive health data, patients’ information, reminder logs and instructions given by the healthcare providers. Figure 4.8 shows the database schema of the health record database in the ER diagram.
Figure 4.8: The PHSF health record database.
This database also contains information about risk boundaries on health attributes which is used to analyze patients' risky situations, and log information of patients' activities such as walking and exercising. This database is used by services in most of the PHSF components through the Health DB Access sub-component in the data access layer.

### 4.2.2 Provider Database

The provider database is mostly used by services in the provider client component. It contains information about healthcare providers and their statuses. This database can be accessed through the services in the Provider DB Access sub-component. The database schema of this database is depicted in Figure 4.9.

![Figure 4.9: The PHSF provider database.](image-url)

Figure 4.9: The PHSF provider database.
4.2.3 Device Database

The device database contains information about health measurement devices, including the device characteristics and attributes that are measured with that device, information about assigning a device to patients and revoking it from patients, as well as the accessing address for the device.

It also contains information about monitoring logs and their current statuses. The device database provides data to the device client component services. Those services need to interact with the Device DB Access service sub-component in the data access layer to access the database itself. Figure 4.10 depicts the schema of this database.

4.2.4 Security Database

The security database contains information about both patients' and providers' security and privacy, such as their identities as well as authorization and auditing log data.

This database is being used by services in most of the PHSF components, since for using each service, the identity of the user must first be validated and authorized. The role in this database indicates whether the user is a patient or a provider. This database can be accessed through the Security DB Access sub-component in the data access layer. Figure 4.11 depicts the
Figure 4.10: The PHSF device database.

schema for this database.

All the operations on these databases are provided as stored procedures on the database side, which enhances security, makes it easier to update, and prevents redundancy.
Figure 4.11: The PHSF security database.
Chapter 5

Case Study

Heart disease is one of the major public health problems in the world. Recent improvements in heart monitoring systems are reducing the cost of treatment and the anxiety of cardiovascular patients [29]. To show PHSF as an effective development framework for developing e-health systems and applications, a heart monitoring application has been designed and implemented using PHSF. This chapter describes the requirements and implementation of the application using the PHSF services.

5.1 Requirements

In this section the requirements for the heart monitoring application is presented, which is based on the proposed application in [31]. The heart monitoring application continuously monitors heart conditions of a patient through
the use of a smart phone and a wireless blood pressure meter sensor. In each
test, the patient’s blood pressure and heart pulse per minute are measured.
The heart monitoring application offers both self-monitoring and remote­
monitoring in which heart patients are monitored by either patients them­
selves or authorized healthcare providers. This application supports multiple
patients and multiple healthcare providers. Figure 5.1 shows the use cases
diagram for this application.

Using the application, a patient can enter health data that is read from blood
pressure meter devices, as well as her daily activities. She can retrieve this
data later and make reports on the data. She can also start monitoring
sessions on a health attribute to read health data automatically in intervals.
She can authorize a healthcare provider to access this data, start monitoring
sessions, and issue reminders and advice to herself. A healthcare provider
is able to access health data, and set monitoring sessions and reminders for
patients who have authorized this health provider.

Both providers and patients should be authenticated, and providers should
be authorized before doing any task. This application analyzes sensor data
in real time and automatically alerts pre-assigned caregivers and the closest
emergency station when the patient is in danger. The location of the person
is retrieved by an application running on the patient’s device via the Google
maps API. This location is sent to the find closest provider service where the
location of the patient is kept and being updated. The providers’ locations
are retrieved from the provider database.
Figure 5.1: Use case diagram for a mobile heart monitoring system.
An emergency caregiver can access the health data, in order to better diagnose and report the data before the patient is admitted into hospital for emergency treatment.

5.2 Design and Implementation

There are three actors or types of users who interact with the heart application: patients, healthcare providers, and emergency caregivers. This section overviews implementation of functionalities of the heart monitoring application for various users, using the PHSF services.

5.2.1 Implementation of Use Cases for Patients

In order for a patient to use the application, she should register with the system first. The add patient service from the information storage/retrieval services component is used to add a new patient. Figure 5.2 shows the web interface for registering a new patient.

If the patient has registered before, the authentication service from the privacy and security services component is used to authenticate the patient. After login, the patient is directed to the application’s home page. The home page for patients contains basic information of the patient, a link to access historical health records, a link for authorizing a healthcare provider, and a link for editing demographic information.

By clicking on the show health data link, the patient can view historical and
current health records as shown in Figure 5.4. The result can be narrowed down by date and attributes through the textboxes and dropdown list provided on top of the page. The *retrieve health record service* in the information storage/retrieval services component is used to retrieve patients’ health data. Figure 5.4 shows the page that contains patients’ health records. Figure 5.3 shows the home page for patients and Figure 5.5 shows the page for authorizing a healthcare provider by a patient. There are seven links at the bottom of the page which are explained as follows.

- Home: this link directs the patient to the home page.
Figure 5.3: Patient home page in the heart monitoring application.

Figure 5.4: Health record page in the heart monitoring application.

- Schema: this link directs the patient to the schema configuration page, which is shown in Figure 5.6. In this page a patient can define new
schema rules for a health attribute. The patient chooses an attribute and then defines the limitations for the value of that attribute. Emergency thresholds are used to find emergency situations in which an ambulance will be called. Warning thresholds are used to find risky situations in which a warning is sent to the healthcare providers that have been authorized by the patient. The add schema rule service from the information storage/retrieval package is used to add a new schema rule for an attribute by a patient.

- Sensor: this link directs the user to the sensor configuration page as depicted in Figure 5.7. This page is for registering a new device and assigning it to a patient or assigning a registered device to a patient. The retrieve device service from the information storage/retrieval services component is used to retrieve a list of all registered devices to
show to the patient, and let her to assign one of them by using the *assign device service* from the same component.

The *add device service* from the information storage/retrieval services component is used to register a new device. The address filed in the sensor configuration page is used to find a device and send a request to retrieve health data in each monitoring session. To find out if the health device is functioning correctly, the measured health data should be confirmed as a correct data. The upper restriction and lower restriction fields are used later in the device adjustment service, to specify if the measured health data is correct or not.
Figure 5.7: Sensor configuration page by Patient in the heart monitoring application.

- Monitoring: this link directs the patient to the monitoring scheduling page which is shown in Figure 5.8. In this page, a monitoring session for a patient can be started or stopped. Running monitoring sessions for a patient are shown at the bottom of the page, where the patient can stop the session by clicking on the terminate button beside that session. At the top of the page, the patient can start a monitoring session by specifying a device and the attribute that the health data should be measured for, as well as the duration and intervals during
which the health data should be measured.

![Heart Monitoring](image)

**Figure 5.8: Monitoring configuration page by patient in the heart monitoring application.**

Once the start button is clicked, a monitoring session is started. In each monitoring session, a timer is set and when the timer counts down, a request would be sent to the patient’s health device to get the measured health attribute’s value.

The *monitoring scheduling service* from the device client services component is used to schedule a monitoring session for a specific health attribute that is measured by a device applied to a patient. This service starts a monitoring session and sends a request for measuring health attributes to the *gather data service* in the device. The device’s gather data service measures and returns the requested health value.
Since we did not have a wireless blood pressure measurement device, we used an Android application which can be installed on patients’ Android device to insert blood pressure. This application is implemented in Android 4.2.2. Figure 5.9 shows an example, when gather data service from patient’s device is called and the patient is asked to insert the blood pressure. The patient uses a blood pressure measurement device to obtain the reading and save it in the application.

After data is gathered, the device adjustment service from the device client services component is used to check if the result data from the test is normal, by comparing the result data with the expected range that the patient has specified during the device registration. If the result is incorrect, a message will be sent to the patient to calibrate the device and repeat the test by calling alarm service from the patient’s device. Otherwise the save health record service is used to insert the result data as a health record into the database.

After health data is saved, the risk analysis service in the patient application services component is used to find risky situations. This service compares the health value result from the test with the value that the patient has specified in the schema rule to find risky situations. If the situation is medium risky, then a warning alarm is sent to the healthcare providers who have been authorized by the patient, by invoking
Figure 5.9: Measurement notice reminder in the heart monitoring application.

the alarm service in the provider device.

If the situation is identified as an emergency situation, the emergency call service in the patient application services component is called which sends an alarm message to the authorized health providers. This service
also sends a message similar to Figure 5.10 to the patient using the
\textit{alarm service} from the patient’s device, indicating that the situation is
risky.

If the patient feels healthy, she can discard the message. Otherwise
the \textit{find closest provider} from the provider client services component is
called to find the closest emergency provider to the patient. This service
gets the patient’s location from the \textit{emergency call service}, retrieves the
emergency providers’ locations from the database and compares their
locations to find the closest emergency by using the \textit{distanceCalculation}
function of the Google maps API (JSON library). After the nearest
provider is found, the \textit{alarm service} from the emergency device is used
to notify the nearest emergency provider.

It is good to mention that an application on a patient’s device retrieves
and updates the location of the patient, as a combination of longitude
and latitude, in the \textit{emergency call service} by using the Google maps API.

The following figures are activity diagrams for monitoring scheduling.
In Figure 5.11, the gathering data activity consists of a sequence of
detailed activities as shown in figure 5.12), which are repeated at each
time tick.

\begin{itemize}
  \item Provider Report: this link directs the patient to a page to search for the
  registered healthcare providers and the ones that have been authorized
\end{itemize}
Emergency Situation

The monitoring system identified a emergency situation because of the following:

High Blood Pressure

Call Emergency (Calling in 8 Sec.)

Cancel

Figure 5.10: Emergency alarm in the heart monitoring application.

by the patient before. The provider reporting service in the provider client services component is used to provide these reports. Figure 5.13 depicts the provider report page.

- Activity: this link directs the patient to a page to search for the previously recorded patient’s daily health related activities, as well as adding a new activity. Routine activities such as eating, walking and exercising can be saved by the patient using the save activity service in the information storage/retrieval services component. These activities can
be reviewed by using the \textit{retrieve activity service} to find what activities have a better impact on the patient’s overall health.

- Advice: this link directs the patient to the advice page where the pa-
Figure 5.13: Provider report page in the heart monitoring application.

tient can see the instructions that authorized healthcare providers issued to her. The *retrieve advice services* in the information storage/retrieval services component is used to retrieve such instructions.

### 5.2.2 Implementation of Use Cases for Healthcare Providers

A healthcare provider should be registered in order to use the application. The *add provider service* from the information storage/retrieval services component is used to register a new provider into the application. The location of the providers can be obtained from their mobile device, same as how the location of a patient is retrieved, but for simplicity, the location of a provider as a combination of longitude and latitude is retrieved from the address that the provider inserts at registration time by using the Google maps API. This
location is used later to find the closest provider in emergency situations.

The authentication service in the privacy and security services component is used to authenticate the registered provider. The provider’s home page is the first page that a provider sees after login.

Figure 5.14: Provider home page in the heart monitoring application.

Figure 5.14 shows the home page for the provider. This page contains all the patients that this provider is authorized for. The get provider’s patient service in the security and privacy services component is used to get all the patients who have authorized this provider, and the retrieve patient service in the information storage/retrieval services component is used to retrieve these patients’ information.

By clicking on each patient, the provider is directed to another page which contains the chosen patient’s demographic information, plus a link for viewing
the patient’s health records (a page similar to the patient home page as shown in Figure 5.3). The provider can click on the *show health data* link to view historical and current health records for that patient as shown in Figure 5.4. In the provider home page there is also a link for adding a new patient which directs the provider to a page as in Figure 5.2. There are eight standard links at bottom of all pages that providers use. Those links are explained below.

- **Home**: this link directs to the provider’s home page.

- **Schema**: this link directs to the schema configuration page, which is similar to figure 5.6 for patients. The difference is that the provider should specify the patient that this schema is defined for.

- **Sensor**: this link directs to the sensor configuration page. The sensor configuration page is for assigning a registered device to a patient that has authorized this provider, or registering a new device and assigning it to the patient. This page is similar to what is depicted in Figure 5.7. The difference is that the provider should specify or select the patient that the device is going to be assigned to.

- **Advice**: this link directs to a page to assign an instruction or advice for a patient. The *save advice service* in the information storage/retrieval services component is used to fulfill this function.

- **Activities**: this link directs to a page that can review a patients’ activities by using the *retrieve activity service*, to find what activities have
a better impact on the patient’s overall health.

- Patient Report: this link directs to a page that contains reports on registered patients. The patient reporting service in the patient application services component is used to produce these reports.

- Provider Report: this link directs to a page that contains reports on registered providers. The provider reporting service in the provider client services component is used to produce these reports.

- Monitoring: this link directs to the monitoring scheduling page in which a monitoring session for a patient can be started and stopped. The monitoring scheduling page for providers is almost the same as shown in Figure 5.8, except for the following:

  - The provider should specify the patient that the provider wishes to monitor.
  
  - The risky situation is analyzed using the constraints that have been specified by this specific provider.
  
  - If the situation is risky, an alarm message is sent to this specific provider and not to all the authorized healthcare providers.
Chapter 6

Experiments and Evaluation

This chapter reports two kinds of experiments on the implemented heart monitoring application as well as an evaluation of the PHSF from a software quality point of view. This chapter also provides a comparison between the PHSF and other similar e-health systems and applications.

6.1 Scenario Based Experiments

This section presents a usage scenario for the heart monitoring application that is presented in chapter 5. This scenario has 3 actors involved including patients, healthcare providers, and emergency caregivers. Each actor uses a sequence of services to fulfill some tasks. This scenario is used to test if all the services of PHSF operate successfully, as well as to report the response time of each application step in single load or usage.
Figure 6.1: Data gathering scenario in a monitoring session for the heart monitoring application Part 1.
Figure 6.2: Data gathering scenario in a monitoring session for the heart monitoring application Part 2.
Figure 6.1 and Figure 6.2 depict the sequence of application activities that is repeated for each timer tick in a monitoring session that is started by a healthcare provider. As Figure 6.1 shows, the scenario starts when a data gathering request is sent to the patient’s device by the monitor schedule service. In each request, the gather data service from the patient’s device is called which asks the patient to insert the blood pressure through an Android application. The patient uses a blood pressure measurement device to do a health test and insert the result to the application and submit it.

After data is received from the patient, the device adjustment service checks if the result data is correct or not, by retrieving the constraints that have been defined in device registration time, using the query constraint service. Then, if the data is not correct, the service sends a calibration request to the device which asks the user to calibrate the device. If the data is correct, the add health data service is used to save the health data.

After the data is saved, the risk analysis service is used to confirm that the data is not risky. This service uses the query schema service to retrieve the criterion for risky data. If the data reaches its emergency limit, the service sends a warning to the patient by calling alarm service from the patient’s device. If the patient does not respond to this warning message by the pre-defined time, the emergency call service is called. The rest of scenario is shown in Figure 6.2.

The emergency call service keeps patient geographical location (this location is updated by a service running on the patient’s device every 5 minutes), uses
the *query health data service* to retrieve the risky health record, and uses the *find closest provider service* to find the closest emergency caregiver.

The *find closest provider service* retrieves all emergency caregivers' locations from the provider database, finds the closest emergency by using the distanceCalculation function of the Google map API (JSON library) to compare providers and patient locations. This service sends an alarm to the emergency caregiver with the patient's location and the risky health data by calling the *alarm service* which is running on emergency devices. This service also retrieves the patient's authorized healthcare providers and sends the alarm message to them by calling the *alarm service* running on providers' devices.

The computing environment for testing the above application scenario used three computers.

- The first one is used as a server to run the PHSF services, with Intel Core i5-3450 2 GHz CPU with 3 GB of RAM, Windows 7, JRE 1.7, tomcat 7.0, and MYSQL 5.6.

- The second one is used as a patient's device with the following configuration: Intel Core i5-3210M 2.5 GHz with 6 GB of Ram, Windows 8 running Android 4.2.2 emulator. There is an Android application running on the emulator which provides 3 services:
  
  1. A service which is used by the monitoring scheduling service to ask the patient to measure her blood pressure and heart beat through
the Android application on this computer. This application is implemented in Android 4.2.2.

2. A service that uses Google maps API (JSON library) to retrieve patient’s location (longitude and latitude), and update it by the emergency call service that is running on the PHSF server, every 5 minutes.

3. A service for receiving alarm messages.

- The Third computer is used as provider/emergency device and has the following configuration: Intel Core i5 2.4GHz, 4GB ram, OS X Lion, and JRE 1.7. There is a simple service running on this computer which sends an email to healthcare providers/emergency. This service can be called in emergency situations. The healthcare provider’s location (longitude and latitude) is extracted, using the Google maps API (JSON library), from the provider’s address which was given by the provider at registration time and saved in the database.

The response time to complete each step of this sequence diagram is presented in Table 6.1.

As shown in table 6.1, the total response time of running the scenario, starting from when the gather data request is issued and ending when the healthcare
Table 6.1: Response time for the sequence of services for data gathering scenario.

<table>
<thead>
<tr>
<th>Service</th>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather data</td>
<td>Device dependant</td>
<td>30 ms</td>
</tr>
<tr>
<td>Device adjustment</td>
<td>Data validation</td>
<td>27 ms</td>
</tr>
<tr>
<td></td>
<td>Query constraint</td>
<td>32 ms</td>
</tr>
<tr>
<td>Add health data</td>
<td>Insert health data to the database</td>
<td>53 ms</td>
</tr>
<tr>
<td>Risk analysis</td>
<td>Comparison Analysis</td>
<td>18 ms</td>
</tr>
<tr>
<td></td>
<td>Query schema</td>
<td>28 ms</td>
</tr>
<tr>
<td>Emergency call</td>
<td>Alert patient</td>
<td>17 ms</td>
</tr>
<tr>
<td></td>
<td>Retrieve providers locations</td>
<td>35 ms</td>
</tr>
<tr>
<td></td>
<td>Find closest emergency</td>
<td>55 ms</td>
</tr>
<tr>
<td></td>
<td>Alert emergency</td>
<td>19 ms</td>
</tr>
<tr>
<td></td>
<td>Alert provider</td>
<td>18 ms</td>
</tr>
<tr>
<td>Response time</td>
<td></td>
<td>332 ms</td>
</tr>
</tbody>
</table>

provider receives the alert, is 332 ms. Here the reported time for each service is the average time of 50 tests for this scenario.

The time for gathering data is the duration from sending the request to receiving the test result from the Android application on the patient’s device. Since we did not have the actual blood pressure meter, we assume that the result is ready when the request arrived, and we did not consider the time that the health device used to measure the health attribute. The time for emergency call includes the time for retrieving all providers’ locations from database plus the time to compare their locations with the patient’s location. We used the Google maps API to do the comparison between locations, so the response time for this operation contains the Google server’s response time plus our server’s response time.
6.2 Scalability Test Experiments

This section reports the experiments and analysis on the impact of increasing system loads in different dimensions in terms of the system latency of the heart monitoring application. These scalability tests are used to measure the response time for multiple loads (beyond a single request). The purpose of this section is to evaluate whether the PHSF can respond to a high volume of requests in a reasonable amount of time. Several experiments were conducted to investigate this evaluation. The dimensions that are used to evaluate the PHSF are as follows.

- Device requests (requests per second): Requests that are coming from health measurement devices such as using save health service.
- Patient requests (requests per second): Requests that are coming from patients such as using retrieve health service.
- Provider requests (requests per second): Requests from health providers such as the patient reporting service.
- Mixed requests (requests per second): Mixed requests from health providers, patients and health measurement devices.

The effect of variation of these dimensions on the system latency was measured. Such system latency was measured for the following services which are the most used and critical services.
- Save health records (ms): The time for saving a new health record in the database.

- Risk analysis (ms): The time that takes to indicate the degree of risk for a new health record.

- Emergency call (ms): The time that takes to find the closest provider and call it.

The computing environment for testing the above application scenario used the same three computers as the scenario based experiment. The first computer is used as a server to run the PHSF services. The second computer is used for producing request traffic to the PHSF server to simulate patients and providers’ requesting behaviour. The third computer for checking latency of service executions.

### 6.2.1 Experiment results

The following figures depict the results of these experiments. Figure 6.3 illustrates the average response times of the 3 services, for different request loads from measurement devices.

Figure 6.4 shows the effects of increasing the number of requests from patients on the system latency, and Figure 6.5 shows how the growth of the number of requests from providers affects the average response time of the system.
The above experimental results reveal that the impact of increasing the requests' volume on the latency of the system is low. It is also worth mentioning that the reason that the emergency call service produces more delay than other services because of its internal functionality to compare all providers'
locations to the patient’s current location.

So far in this section we reported the effect of increasing requests from devices, patients, and providers separately. Those experiments are used to test if increasing requests from any actor impacts the application’s performance abnormally. In reality, the application should respond to mixed requests from devices, patients, and providers together. In the following, the experimental results for the effect of increasing the mixed load on the application response time are presented.

In order to find the average response time for each request rate, the simulation ran for 30 seconds and average response time was calculated. The type of each individual request that is used for creating traffic, selected randomly from device, patient and provider. Figure 6.6 depicts the experiment result for mixed request, which is similar to those based on individual actors’ requests.

Figure 6.5: Effect of increasing patient request traffic on response time in the Heart Monitoring Application.
6.3 Evaluation on Framework quality

This section presents an evaluation of the PHSF in terms of PHSF’s supports to build quality software applications for e-health.

- Performance: the performance of the PHSF is examined through the experiments that are reported in the previous section. Those experiments show that the performance of the e-health applications that are built upon the PHSF is acceptable.

- Scalability: in order to check the scalability of the PHSF, several experiments have been conducted to check the effect of an increasing number of requests on the system response time. The results of scalability tests

Figure 6.6: Effect of increasing mixed request traffic on response time in the heart monitoring application.
that are presented in the previous section show that, the increased time latency is small as the number of requests increases.

- **Reusability**: since the PHSF is provided as reusable services, users or application developers can reuse different combination of PHSF services to build many different e-health applications. These applications can be developed using different programming languages and operation platforms, because PHSF services are implemented as loosely coupled web services by using a service-oriented architecture.

- **Ease of application development**: the PHSF offers a base line for developing e-health applications for personal health devices. Depending on the desired e-health applications requirements, different combinations of the PHSF service implementations can be directly used without re-development. So application developers do not need to start everything from scratch.

- **Extensibility and modifiability**: since the PHSF is based on SOA, new services can easily be added as new web services in PHSF. The PHSF as an open framework that lets users or application developers build their own services and add them to PHSF service and/or e-health applications built on PHSF.
6.4 Comparisons

This section presents comparisons between PHSF and other similar system, including a comparison on the functionalities that are related to personal health devices and a comparison based on the software quality attributes which are listed in the previous section. The following is a classification of the functionalities of the PHSF.

- Provider-patient link: this functionality indicates whether a link between healthcare providers and patients is provided or not. This linkage means the ability for a provider to access a patient’s health information, or to perform some tasks such as reminding the patient of some tasks.

- Self-monitoring: this functionality shows whether a patient can setup a monitoring session to measure her health status on a regular basis or not.

- Remote-monitoring: this functionality shows whether remote monitoring is provided or not (for example if a healthcare provider can set up a monitoring session from a distance or not).

- Risk analysis: this functionality is for indicating whether risk control and data processing to find suspicious health situations is provided or not.
• Emergency support: this functionality shows if emergency situations are handled or not.

The following comparative analysis shows a comparison of the PHSF and some similar systems in terms of the above functionality categories. Table 6.2 summarizes this comparison. Note that the comparison is not a complete functionality comparison.

Table 6.2: Functionality comparison of PHSF and other systems.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider-Patient Link</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Self-Monitoring</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Remote Monitoring</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Risk Analysis</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Emergency Support</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

The provider-patient link is supported in the PHSF, so all the patient records can be accessed by authorized healthcare providers, patients can search on existing providers and authorize them, and emergency situation can be issued to near-by emergency providers.

Self-monitoring and remote-monitoring links are supported in the PHSF through the use of the monitor scheduling service in which a monitoring session can be started and stopped by both patients and authorized healthcare providers. A risk analysis link is also supported in the PHSF through using the risk analysis service which is used after new health data is saved.
to figure out if the health data is risky or not. An emergency support link is supported by the PHSF by announcing risky situations to authorized health providers and the nearest emergency caregiver together with the risky health data and patient’s location.

Table 6.3 shows another comparative analysis between PHSF and other system, based on some of the software quality attributes.

Table 6.3: Software quality comparison of PHSF and other systems.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusability</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Ease of application development</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Extensibility</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion

In this chapter a summary of the thesis, contributions of the thesis research, and some future work are presented.

7.1 Summary

In this thesis, a framework named Personal Health Service Framework (PHSF) is proposed. The PHSF is a service-based framework for developing e-health applications and monitoring systems for personal health devices. The PHSF is designed based on the common structures and features of existing personal health-related applications. The PHSF supports e-health applications with multiple healthcare providers and patients. The proposed framework consists of 7 components. Each component contains a set of services which serves similar functionality. From another point of
view, the services in this framework are organized in 4 separated layers. The first layer contains services that are used by end users, the second layer includes application services to support the front-end services. The third layer is the data access layer which includes services to access the databases. The fourth layer consists of the four database for storing health data, health provider data, health device data, and security related data.

Java has been used for implementing the PHSF services using the J2EE platform and the Spring framework. Apache CFX was used for developing services in the PHSF and MYSQL is the database technology to implement the PHSF's databases.

A heart monitoring application for monitoring cardiac patients, as a case study, has been developed using this framework. This application mainly monitors patient blood pressure and other activities such as walking in regular intervals. The monitoring sessions can be set by patients for self-monitoring, and by healthcare providers as well. In each interval, the application gathers health records and checks if the data is risky or not after saving it in the database. It then sends an alarm to the healthcare providers or emergency caregivers based on the degree of risk. For providers to use the system, they should be pre-authorized by patients.

The scenario-based experiment and scalability test experiment have been conducted for the heart monitoring application. Results for these experiments show that the response time of the application based on PHSF is acceptable for various loads of the application.
7.2 Contributions

This section lists the technical and practical contributions of this thesis research as follows.

7.2.1 Technical Contributions

The PHSF improves technology for developing personal e-health applications by providing a systematic approach. There are different kinds of systematic approaches. In this thesis, we proposed a framework-based systematic approach for developing e-health applications for personal health devices. The PHSF prototype as a service-oriented framework supports high quality development of e-health applications in terms of being reusable, easy for rapid application development, and extensible.

The framework provides reusability since the needed functionalities provided through web services that can be shared or reused by multiple different e-health applications. It is easy for application development since the framework provides a common structure for e-health applications. By following this structure and using the provided reusable services, e-health applications can be rapidly developed. The PHSF is also extensible since it is as an SOA-based open framework, so new services can be added or composed to the framework.
7.2.2 Practical Contributions

Most of services of the PHSF have been implemented and deployed on a network connected server. Such services are available to users or e-health application developers to use.

- For end users, the framework itself is available as a running system to be used for basic functionalities that are provided as services. For example, patients can register their devices and store health data measured by the devices.

- For E-health application developers or practitioners, they can use PHSF services to build their own e-health applications. Based on the functionalities that are needed to build the application, different services might be used from the PHSF.

7.3 Future Work

Providing PHSF services in the cloud is one future direction. The PHSF provides a platform to build e-health applications. Providing such software to the public requires huge computing capacity. Offering the PHSF services on the cloud or distributed servers can be a solution to handle huge traffic usage.

The PHSF can also be further developed as a framework in which service components, as shown in the Figure 3.2, can be provided by different e-health
application providers in the cloud. So an e-health system can be built upon such a framework by using a *Device Client Services* component provided by one provider and a *Provider Client Services* component from another one. The PHSF can be further developed to directly store and retrieve health data in the electronic health record (EHR) format in the database, in addition to the personal health record (PHR) format. EHR is more general than PHR as it can contain more information, such as all patients' information and history, than PHR which is more individual patient related.

Since the PHSF provides the base line for building e-health applications, many functionalities can be added to this framework as services. One of the components (a set of services) that can be added to the framework is a safety services component. This component would contain services that are responsible to make sure that any instructions from healthcare providers do not cause safety problems for their patients, to prevent medical errors that often lead to adverse healthcare events. This service would check the healthcare providers' advises against rules that has been defined previously. Also, the PHSF framework can be tested in actual field trial, with actual patients and providers in future.
Bibliography


[27] Alex Mu-Hsing Kuo, *Opportunities and challenges of cloud computing to improve health care services*, Journal of Medical Internet Research 13 (2011), no. 3.


[31] Eun-Shim Nahm, Kay Blum, Barbara Scharf, Erika Friedmann, Sue Thomas, Deborah Jones, and Stephen S Gottlieb, *Exploration of patients’ readiness for an ehealth management program for chronic heart*


[36] Nuria Oliver and Fernando Flores-Mangas, Healthgear: a real-time wearable system for monitoring and analyzing physiological signals, Wearable


Vita

Candidate’s full name: Shirin Ghorbani
University attended (with dates and degrees obtained):

- Azad University-Parand, Tehran, Iran
  B.Sc. of Information Technology
  Dec 2010

Publications: