UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE AMPOSDRU
Fondul Social European POSDRU 2007-2013
Instrumente Structurale 2007-2013
MINISTERUL EDUCAŢIEI CERCETĂRII TINERETULUI ŞI SPORTULUI OIPOSDRU
UNIVERSITATEA "POLITEHNICA" din BUCUREŞTI

FONDUL SOCIAL EUROPEAN

Investeşte în oameni!

Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013

Proiect POSDRU/88/1.5/S/61178 – *Competitivitate şi performanţă în cercetare prin programe doctorale de calitate (ProDOC)*

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI

Facultatea de Automatică şi Calculatoare

Catedra de Calculatoare

Nr. Decizie Senat 11463 din 23.07.2012

TEZĂ DE DOCTORAT

Standarde de Comunicare pentru Sisteme de e-Sanatate: Extensii de Securitate si Noi Metode de Testare a Conformitatii

eHealth Communication Standards: Security Enhancements and New Conformance Testing Methods

Autor: Ing. Alexandru-Ionut Egner

COMISIA DE DOCTORAT

Preşedinte	Prof. dr. ing. Adina Florea	de la	Universitatea POLITEHNICA din Bucureşti
Conducător de doctorat	Prof. dr. ing. Florica Moldoveanu	de la	Universitatea POLITEHNICA din Bucureşti
Referent	Prof. Dr.-Ing. Alexandru Soceanu	de la	Munich University of Applied Sciences, Germania
Referent	Assoc. Prof. Carlo Ferrari, Ph.D.	de la	University of Padova, Italia
Referent	Prof. dr. ing. Nicolae Ţăpuş	de la	Universitatea POLITEHNICA din Bucureşti

Bucureşti 2012

EHEALTH COMMUNICATION STANDARDS: SECURITY ENHANCEMENTS AND NEW CONFORMANCE TESTING METHODS

A Dissertation

Submitted to the Faculty of Automatic Control and Computers

of

University POLITEHNICA of Bucharest

by

Egner Alexandru Ionut

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor in Computer Science

November 2012

University POLITEHNICA of Bucharest

Romania

ACKNOWLEDGEMENTS

I am very fortunate to have a wonderful family, awesome friends and to be part of an excellent research team. I am very grateful for all the support they offered, for their patience, advices, and for the fact that they mould me into a much better self.

I would like to express my gratitude to my advisor, Prof. Florica Moldoveanu. She was the person who guided me throughout this interesting journey and taught me what research is. I am thankful for the opportunity to be involved in several research projects from which I learnt a lot. I am grateful not only for supporting my work, but also for making it even enjoyable.

I consider myself very fortunate to have met Prof. Alexandru Soceanu and there are many things for which I would like to thank him. I enjoyed every talk we had. I cannot forget that he believed in my ideas and sponsored my research many times. I am also thankful for initiating me into the beauty of art and architecture and for showing me that sometimes is better to detach from work in order to clear your mind. I would also like to thank Marion Soceanu for her support and advices. She offered me great insights on how to structure the thesis and how to present my work.

Special thanks go to the University of Padova, Italy, and especially to professors Carlo Ferrari, Michele Moro and Sergio Congiu. They offered me all their support. I am grateful for the time they spent discussing about this research and for the valuable ideas that resulted from these brainstorms.

I am very lucky to have been involved in the excellent research team from University POLITEHNICA of Bucharest. I consider Victor, Alin and Anca not only colleagues, but also close friends and I enjoyed every time spent with them.

I would like to thank Diana Vega for many things. She offered me all her support when I was still trying to define a research direction and she spent much time discussing with me and reviewing my work.

I am very grateful to Prof. Radu Virbanescu, who inspired me as a student and always encouraged me to follow the academic path. I am very thankful for his efforts to stimulate me achieve my full potential.

My deepest gratitude goes to my family. My mom, dad, brother and grandparents have always been close to me and I could always count on them. I would also like to express my gratitude to my wife's parents, who were very supportive throughout these years.

Many thanks go to my best friends, Catalin, Radu and Vasile.

I am especially grateful to Gia for her love and affection. She always believed in me, gave me strength and encouraged me, no matter what. I am very thankful for every single thing she does, because my life is wonderful when she's with me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Motivation

eHealth is a concept that reflects the symbiosis between Information and Communication Technology (ICT) and healthcare. The term can be traced back to the late 1990s, when it was mainly referred by industry and marketing specialists. Since then, the meaning and scope of eHealth underwent significant changes to reflect the emergence of new technologies and methodologies.

In 2001, Gunther Eysenbach, a German researcher, founder and editor-in-chief of the Journal of Medical Internet Research, gave one of the first eHealth definitions: "eHealth is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information *delivered or enhanced through the Internet and related technologies*". In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology" [EYS01].

In 2005, Hans Oh and a small group of researchers made a systematic review on the definitions of eHealth published in English [OH05]. The review mentions more than 50 unique definitions, highlighting the wide variety of meanings assigned to the term. The review also points out that "there is no clear consensus about the meaning of the term eHealth".

The difficulty to map eHealth to a unified view was accentuated also by the exponential growth of the domain. Factors such as the growing number of patients with chronic diseases, the aging population worldwide, the high costs and low effectiveness of the traditional healthcare have determined the explosion of the eHealth domain.

Therefore, in 2012, the World Health Organization (WHO) redefines its own definition of eHealth from 2005 as "the transfer of health resources and health care by electronic means. It encompasses three main areas:

- the delivery of health information, for health professionals and health consumers, through the Internet and telecommunications;
- using the power of IT and e-commerce to improve public health services, e.g. through the education and training of health workers;
- the use of e-commerce and e-business practices in health systems management."

The definition from WHO underlines an encompassing view on the eHealth domain, highlighting the key differences between eHealth and traditional healthcare.

The view of eHealth presented in the thesis is in the line with the one presented by WHO. The thesis emphasizes the relationship between eHealth systems and the need for complex network infrastructures to ensure interconnectivity within the Internet and communication standards to provide interoperability amongst eHealth services. In addition to the definition from WHO, the thesis underlines the importance of standards and policies for ensuring interoperability and security of data.

Regardless of the view of the eHealth domain, however, the future of worldwide healthcare without eHealth solutions is not possible, as predicted by the customer-dependent market research and analysis of Frost & Sullivan after an extensive research [FRO09].

The importance of eHealth systems is substantiated by their crucial role in providing patient care. Eysenbach [EYS01] believes that the "e" in eHealth does not only stand for "electronic", but also for:

Efficiency

eHealth systems improve the efficiency of standard healthcare systems in two ways by:

- reducing the time required to perform health tasks and processes;
- reducing the costs;

Enhancing of quality care

The improvement of quality care is an important outcome of adopting eHealth solutions. Special devices can monitor patients' vital signs and report information to Healthcare Information Systems (HIS). This means that patients' Electronic Health Records (EHR) can be constantly updated. Moreover, continuous monitoring offers a different perspective on patients' health status. There are significant differences, for instance, when monitoring patients' heart activity in a particular environment, for a particular period of time as opposed to continuous monitoring in the patients' environment.

Empowerment

eHealth systems empower patients by giving them access to their medical records, which enables them to be more involved in the decision making process.

In recent years, the direction of the eHealth domain has been undergoing significant changes, from an emphasis on hardware and system architectures toward innovative uses of technology for facilitating communication and clinical decision-making. This change was directly influenced by the cooperation between healthcare associations and standardization organizations that focused research and development on technologies that promote the various communication standards.

The potential of eHealth is vast and there are many opportunities for researching innovative solutions designed to enhance the security of medical data, the interoperability and communication between eHealth systems, and ultimately to improve patients' lives.

1.2 Objectives

The eHealth field can be viewed from two different standpoints, as shown in [FRO09]: applications and services. Applications cover various information systems used in particular healthcare practices, such as Electronic Health Records (EHR), Picture Archival and Communication Systems (PACS), Electronic Prescription Systems (EPS) or Electronic Health Card (EHC). In this thesis the applications are referred to using the generic term of Healthcare Information System (HIS). The services pertain to telehealth services, such as telemedicine and telecare.

This research focuses on improving the way telehealth services interact and communicate medical information through the use of standardized communication protocols. The primary objective is to identify the problems that occur when creating a healthcare environment derived from the reference architecture proposed by Continua Health Alliance (Continua). The environment will be validated against real-life scenarios. Consequently, solutions to the identified problems will be searched.

Continua views itself as "a non-profit, open industry organization of healthcare and technology companies joining together in collaboration to improve the quality of personal healthcare" [CON12]. It consists of more than 220 member companies around the world. For several years, Continua concentrated its efforts towards the development of Version One Design Guidelines, collecting information from its members about the functionality that a personal telehealth system should deliver. The analysis performed by Continua determined the release of the Continua Reference Architecture [JAR11].

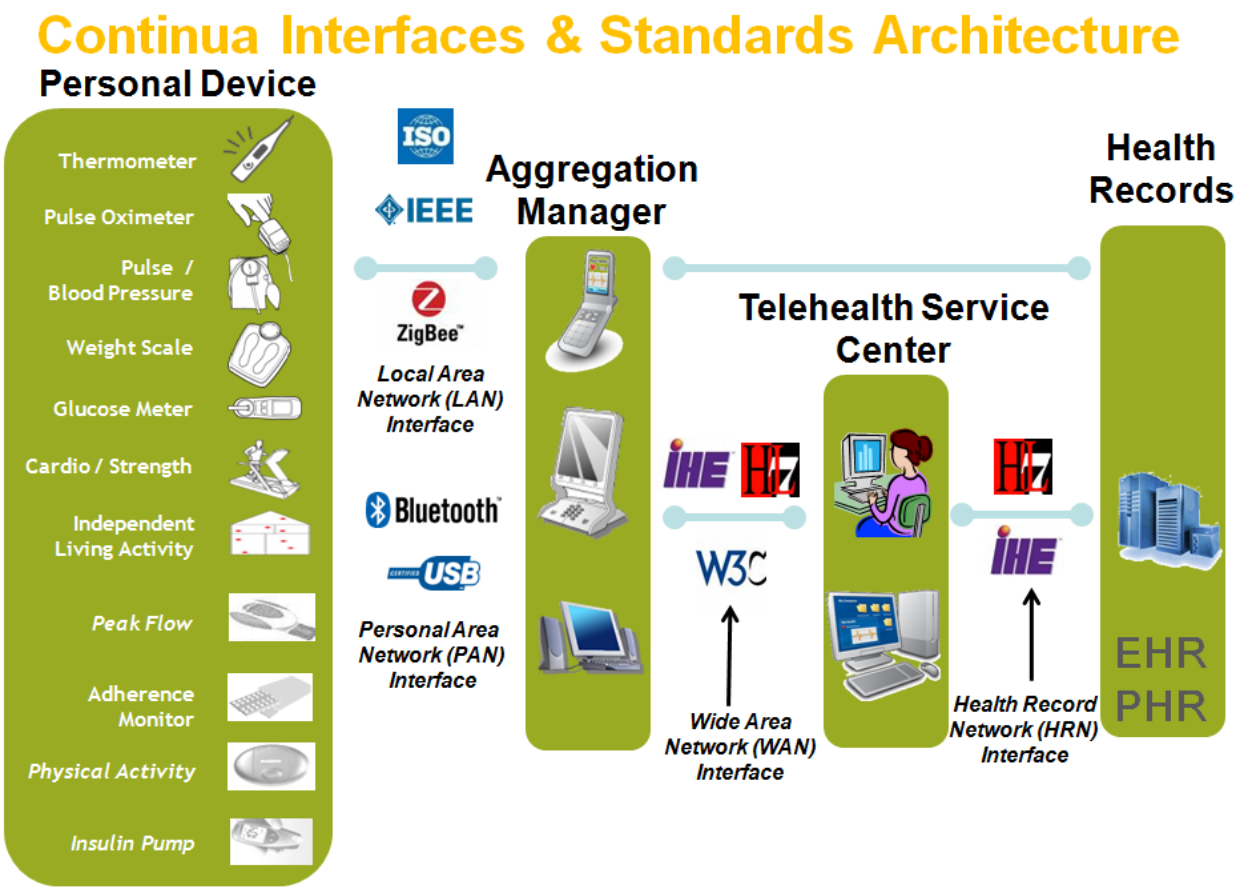

Figure 1.1. Continua Reference Architecture (source: Continua Health Alliance [JAR11])

Continua Reference Architecture defines an ecosystem that enables end-to-end delivery of health information between patients and HISs. The generic design of the Reference Architecture is depicted in Figure 1.1.

The Continua Reference Architecture consists of five reference-device classes and four network interfaces that interconnect the devices. The main goal of Continua is to improve the interoperability between the reference-devices. For this reason, Continua concentrates its efforts in providing guideline specifications and test suites for each network interface.

The network interfaces that form the communication model described in the Continua Reference Architecture are:

- Personal Area Network Interface (PAN-IF) connects sensors (e.g. thermometer, glucose meter, etc.) or actuators (e.g. alarm, device that switches light to on/off, etc.) with application hosting devices (e.g. personal computer, smartphone, etc.);
- Local Area Network Interface (LAN-IF) connects application hosting devices with LAN devices; a LAN device can implement a sensor or an actuator;

- Wide Area Network Interface (WAN-IF) connects application hosting devices with WAN devices, which usually implement network-based services, such as monitoring services;
- Health Record Network Interface (HRN-IF) connects WAN devices with health record devices; as opposed to the other three interfaces, HRN-IF is patient-oriented, instead of device-oriented.

The research presented in the thesis focuses on the PAN-IF and HRN-IF interfaces and the interoperability between the three entities involved:

- the personal device (sensor or actuator);
- the aggregation manager (the application hosting device);
- the Healthcare Information System (HIS).

Communication between the three entities is influenced by their characteristics. Different communication protocols are, therefore, appropriate for transferring medical information within PAN-IF and HRN-IF. For this reason, establishing methods for ensuring the interoperability between the involved eHealth applications is the primary objective.

The communication protocol chosen for transferring the patients' vital signs between the personal device and the aggregation manager is ISO/IEEE 11073-20601 – Optimized Exchange Protocol (OEP). In OEP terminology, personal devices are called Agents and aggregation managers are called Managers. The standard was only recently released and arguably lacks important features, such as methods for ensuring privacy, authorization, authentication, data encryption, etc. Since the responsibility for implementing these features is passed to developers, eventual interoperability problems should be studied. This research aims to extend OEP by adding two enhancements that: 1) provide support for authentication and 2) enable a mutual association initiation. The extensions should be designed in a way that promotes interoperability between Agents and Managers. The authentication should be envisaged as an integral component of an identity management system that additionally enables policies for access control.

Another objective of this research is to find a solution to an important problem identified by the OEP developers' community: the lack of an OEP message analyzer.

The protocol used for transferring information between the aggregation manager and the HIS is Health Level 7 version 3 (HL7 V3). Even though HL7 is more mature than OEP, the redesign of the fundamental model from versions 2.x to version 3 has led to inconsistencies and interoperability problems.

Another research objective is testing the conformance of eHealth applications with the HL7 V3 standard. The particularities of HL7 V3 make testing a very challenging process. The thesis aims to design a methodology for testing the conformance of HL7 V3-based applications that is independent of a specific HL7 V3 integration profile and a testing framework based on Testing and Test Control Notation version 3 (TTCN-3) technologies to support the methodology. The result targeted by this approach is a generic design of a testing framework that renders it applicable for testing eHealth applications based on virtually any HL7 V3 integration profile.

Lowering testing costs is one of the most important desiderates in software engineering. Automating testing processes can significantly reduce costs and errors. Another important objective covered by this research is therefore the automation of the testing process proposed. In order to achieve this, several methods are analyzed for:

- code generators;
- definition of the TTCN-3 type system;
- TTCN-3 template generator.

These automation tools enable test designers to automatically create the setup of the testing environment, which is an important focus of the research.

Arguably one of the most important challenges of test designers is to define the input test data. This is especially difficult for testing HL7 V3-based applications using TTCN-3. The thesis aims to tackle this problem by discovering a method for defining customized input test data. This is an ambitious objective, as the research community has not yet solved the problem of generating HL7 V3 messages.

The final step in ensuring full interoperability within the communication chain depends on the conversion between OEP and HL7 V3 performed at the aggregation manager level. The results of recent research have shown that concepts can be mapped between the two standards. Moreover, semantic interoperability can be achieved. Thus, the thesis considers this aspect covered and focuses solely on the identified interoperability problems.

1.3 Structure of the thesis

This section outlines the structure of the thesis. It provides an overview of the topics covered in this research.

Chapter 2 discusses the need for supplementing the ISO/IEEE 11073-20601 Optimized Exchange Protocol (OEP) to include mechanisms that provide security and at the same time promote interoperability. The main contribution presented in this chapter is the proposal of a novel methodology for generating robust biometric keys derived from the patients' fingerprint scans. The biometric keys can be used as cryptographic keys in algorithms designed to ensure the privacy of patients' medical data and the security of communication between mobile medical devices. Chapter 2 highlights the challenge of creating biometric keys. This process differs fundamentally from matching biometric data for patient verification or patient identification. This chapter introduces several innovative algorithms for generating the biometric key. The algorithms are analyzed and compared with each other. Chapter 2 also shows the applicability of biometric keys in an authentication mechanism designed for the OEP protocol. The proposed mechanism is the mutual challenge-response authentication.

Chapter 3 introduces two enhancements for the OEP protocol. The first enhancement is the extension of OEP to include the mutual challenge-response authentication proposed in Chapter 2.

The emphasis is on extending the protocol while also providing support for interoperability. This chapter elaborates on the redesign of OEP's data model and communication protocol with the aim of offering support for authentication. Specific implementation details, from the extension of the nomenclature and finite state machine that describes the communication protocol with support for authentication through to the illustration of how messages carry authentication information, are provided. The second enhancement presented is the support for mutual association initiation between OEP-supporting medical devices. Chapter 3 also offers a solution to one of the most important problems of the OEP protocol: the need for a message analyzer useful for validating the implementations of the standard and the protocol enhancements. The thesis proposes a solution for this problem – a Wireshark plugin that is capable of capturing and dissecting OEP messages exchanged between Agents and Managers for content analysis.

Chapter 4 introduces the fundamental concepts of testing the conformance of eHealth applications. It provides an overview of the HL7 standardization organization and presents its most important interoperability standards. It emphasizes the HL7 V2.x and V3 standards, highlighting the fundamental differences between them. This chapter introduces the concept of conformance testing and presents cutting-edge technologies and tools used in conformance testing of HL7-based applications. It also introduces the Testing and Test Control Notation version 3 (TTCN-3) technologies that form the basis of the conformance testing framework proposed in the thesis. The special features of the TTCN-3 test system and its data model are discussed here.

Chapter 5 proposes a new conformance testing methodology for HL7 V3-based applications and the resulting TTCN-3-based testing framework. The complex architectural design of the testing framework is described and implementation details and guidelines are provided. In addition, Chapter 5 presents several automation mechanisms integrated in the testing framework that assist test designers in the process of test definition and execution. The most important automation mechanisms described in this chapter are:

- a methodology for generating data types in any programming language that define specific HL7 V3 data types;
- a methodology for generating TTCN-3 type system components;
- a methodology for creating and editing TTCN-3 templates, which are the TTCN-3 test language components used for defining testing inputs and outputs.

Chapter 5 also puts forward an original approach to generating input test data. The novelty of the methodology presented in this chapter lies in the use of customizable distance-based generators in conjunction with high-level customization rules that delivers astonishing flexibility in terms of defining input test data. The result of this approach is the Test Data Generator (TDG). In this thesis TDG is used as an HL7 V3 message generator – a generator long awaited by the HL7 V3 developer community.

Finally, Chapter 6 closes the thesis with a summary of the results obtained. It recapitulates the achievements and their usefulness in accomplishing the overall objective of improving the security and interoperability of eHealth applications.

2 AUTHENTICATION BASED ON BIOMETRIC KEYS

Authentication is an important security feature the protocol ISO/IEEE 11073-20601 lacks. The standard's definition intentionally omits this security aspect and passes this responsibility to the implementer. However, this may lead to mobile medical devices that cannot interoperate. This research proposes an authentication protocol based on biometric keys. The proposed authentication is employed at two different levels: patient-to-device and device-to-device authentication. This chapter analyzes several methods of generating biometric keys that can be used in the authentication protocol, proposes a mutual authentication procedure and presents a real-life scenario with a system that implements the proposed authentication protocol.

2.1 Biometric data

When choosing the type of biometric data used for generating the biometric key, several factors must be considered. This section analyzes the most common biometric technologies available, with focus on their advantages and disadvantages.

Fingerprints

Fingerprint technologies are the most common biometric technologies used today. Fingerprints can be used both for identification and verification. There are two main approaches for processing data contained in a fingerprint template: image-based and minutiae-based. The image-based processing requires the whole information contained in the fingerprint template and needs more processing power. In the other approach, only a small set of special points, called minutiae, which form the unique pattern of the fingerprint, are needed.

A typical fingerprint template contains an average of 30 to 40 minutiae points. The Federal Bureau of Investigation (FBI) has shown that the maximum number of common minutiae points between any two different persons is 8 [MAI97], [JAI01].

Advantages:

- very high accuracy;
- reduced costs;
- mature technology;
- easy to use;
- standardized;
- small storage space required;
- less invasive.

Disadvantages:

- less accurate when scanned finger is wet, greasy, dirty, etc.;
- less suitable for children (fingerprints change quickly in this case);
- can be definitively deteriorated (burns, skin lesions, etc.).

Facial recognition

Face recognition is probably the most natural type of biometric identification. Recently it developed in two distinctive research areas: facial metrics and eigenfaces. The first one studies the measurement of specific facial features and the relationship between these measurements. Eigenfaces are a set of eigenvectors used in face recognition and face classification.

Advantages:

- non-intrusive;
- low costs.

Disadvantages:

- affected by lighting, facial expression, facial hair, glasses, etc.;
- infancy stage of development.

Retinal scanning

This technique uses the unique blood vessel patterns of a person's retina for identification. The retina is illuminated with an infrared light. Blood vessels absorb the energy faster than the adjacent tissue, which brings out the blood vessels patterns. In the recent years, retinal scanning's usage has spread from government agencies to more commercial purposes, such as ATM identity verifications.

Advantages:

- very high accuracy;
- there is no known method for retina replication;
- easy to determine liveness.

Disadvantages:

- very intrusive;

- perceived as potentially harmful for the eye;
- very expensive;
- affected by diseases such as cataracts or severe astigmatisms;
- can be considered an invasion of privacy as the retina may indicate health problems.

Hand geometry

This technology is based on the fact that person's hands have virtually different shapes and they don't significantly change their shape, especially after a certain age. Compared to other means of biometric identification (such as fingerprints), it does not produce a large data set. This is a drawback when a large number of records are stored, as the hand geometry may not be able to distinguish between two hands with similar characteristics.

Advantages:

- accepted by subjects (is commonly associated with authorization access);
- can be integrated, even though the scanner is not small.

Disadvantages:

- affected by diseases such as arthritis;
- small amount of biometric data;
- very expensive.

Voice recognition

This technology relies on the production of a "voice template" that is used for matching a spoken phrase. This technique relies more on the behavior of the subject than on its physical characteristics.

Advantages:

- non-intrusive;
- accepted by subjects;
- inexpensive.

Disadvantages:

- low accuracy;
- easy to duplicate;
- affected by common voice changes determined by hoarseness or cold.

Table 2.1 shows a comparative analysis between the common biometric technologies described above. The healthcare environment imposes a set of constraints that have to be considered when choosing the biometric technology. This research focuses on a novel methodology of authentication between ISO/IEEE 11073-20601 devices based on biometric keys derived from fingerprints. The motivation of the biometric choice consists on the fact that fingerprint readers

can be easily embedded into the mobile medical devices and the fingerprint technology is mature, does not imply high costs and is highly reliable.

Table 2.1. Comparison between different biometric techniques

	Fingerprint identification	Face recognition	Retina scanning	Hand geometry	Voice recognition
Reliability	High	Average	Very high	High	High
Easiness of usage	High	Average	Low	High	High
Acceptance	Average	Average	Average	High	High
Security	High	High	Very high	High	Average
Identification	Yes	Yes	Yes	No	No
Verification	Yes	Yes	Yes	Yes	Yes
Interference	- dirtiness - moisture - burns	- image quality - lighting - hair	- cataract - astigmatism	- arthritis - rheumatism	- hoarseness - cold

2.2 Biometric key generation algorithms

Generating robust biometric keys is a difficult challenge because of the inherent features of the fingerprints. Different scans of the same finger usually provide different images. For this reason, assuring biometric key repeatability is difficult to achieve. Methods for generating unique and repeatable keys have been described in the literature [BHA10], [JUE02], [ZHA04]. The key generation process is, however, still an open problem.

2.2.1 Overview of the novel approach

The proposed algorithm is based on the fact that even though minutiae points belonging to different scans of the same finger don't have exactly the same coordinates, their relative distances differ only by a small offset.

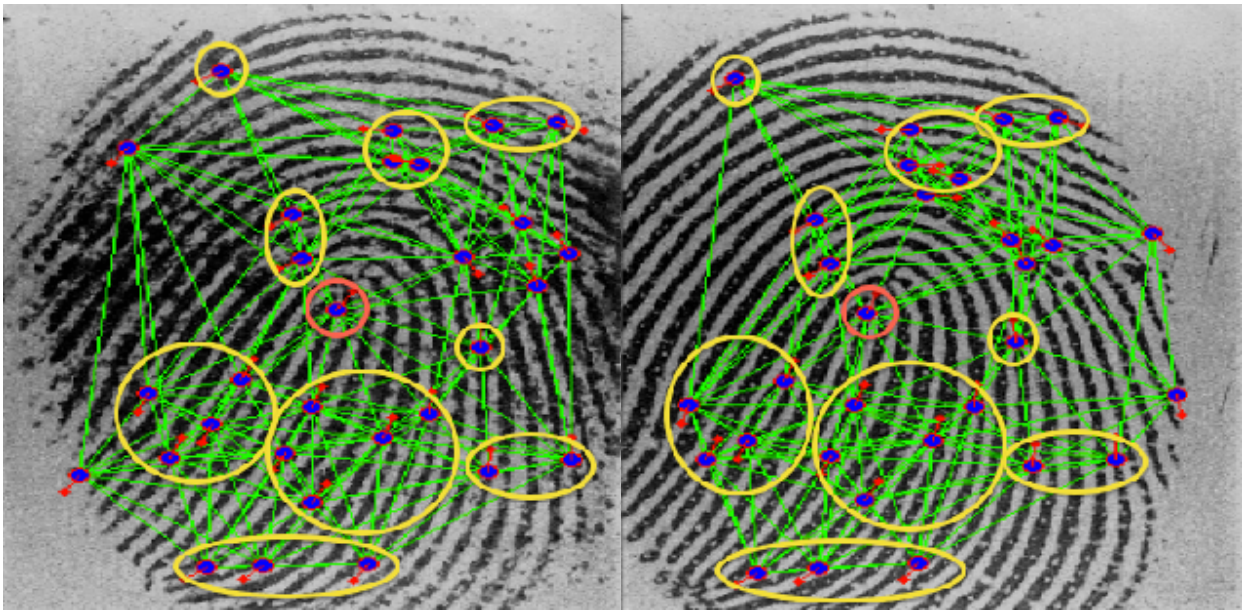

Figure 2.1. Similarity between different scans of the same fingerprint

Table 2.2 also shows there are small offsets between similar minutiae points. Two sets of 10 different fingerprint scans of the same finger were taken. The common minutiae points were extracted for each set, resulting 13 similar minutiae. A minutia point is considered common if present in more than 80% of the input scans, in this case in more than 8 out of 10. Two minutia points from two different scans are considered similar if there is only a small offset between their positions relative to the core.

Table 2.2's column headers describe the minutia point, its coordinates in the feature space (X and Y), its number of occurrences, its distance to the core point and the distance difference between similar minutiae, respectively.

The maximum offset recorded was 2.714 (for P11). This is a small offset in a feature space of 256x360. For comparison, the distances between the closest minutiae points P7 and P9 are 24.198, for the first set of fingerprint scans and 25.607, for the second set. This means that the smallest distance between any minutiae is at least 8.915 times greater than the biggest offset.

Table 2.2. Small offsets between minutiae from different scans

Min.	X coord.	Y coord.	Occ.	Dist. to core	Min.	X coord.	Y coord.	Occ.	Dist. to core	Diff.
Core	128	180			Core	128	180			
P1	42.875	85.5	80%	**127.186**	P1	46.142	84.42	80%	**125.842**	**-1.344**
P2	110	126.444	90%	**56.499**	P2	111.7	124.8	90%	**57.556**	**1.057**
P3	114.666	151.444	90%	**31.515**	P3	116.9	150.1	100%	**31.893**	**0.378**
P4	92.2	219.7	100%	**53.457**	P4	89.8	218.2	100%	**54.022**	**0.565**
P5	54.857	232.428	80%	**89.992**	P5	50.6	229	80%	**91.606**	**1.614**
P6	120.5	235.6	100%	**56.103**	P6	118.4	234.7	100%	**55.536**	**-0.567**
P7	168.777	237.111	90%	**70.174**	P7	167.1	236.8	100%	**68.956**	**-1.218**
P8	78.7	249.1	100%	**84.884**	P8	75.1	245.7	100%	**84.349**	**-0.535**
P9	151.666	254.222	90%	**77.903**	P9	148.5	254.4	100%	**77.172**	**-0.731**
P10	110.4	262	100%	**83.867**	P10	106.3	259.2	100%	**82.118**	**-1.749**
P11	192.571	269.714	80%	**110.535**	P11	190.125	268.125	80%	**107.821**	**-2.714**
P12	187.42	198	90%	**62.086**	P12	188.33	197.33	80%	**62.769**	**0.683**

As Figure 2.1 and Table 2.2 show, different scans do not always reflect exactly the same pattern of the fingerprint. Usually, the core is present in all scans, but the topology of the other minutiae may differ from one scan to another. However, there are many similar minutiae points that can be used in the generation process.

Typically, a high quality fingerprint scan contains 30-40 minutiae points. Experiments were conducted for identifying the approximate number of common minutiae points that can be found in several scans. Table 2.3 shows these results. The Zvetco P5000 fingerprint reader [ZVE] and Griaule Fingerprint SDK [GRI] were used for this experiment.

The first experiment was conducted with a set of 10 different fingerprint scans of the same fingerprint. Common minutiae points were considered the ones that were found in more than 50%, 80% and 100% of the fingerprint scans, respectively. The experiment repeated with larger sets of fingerprint scans.

Table 2.3. The relation between the number of common minutiae and the number of scans

Fingerprint Scans	Minimum number of minutiae occurrences	Number of common minutiae points (average)
10	5 (50%)	30
	8 (80%)	15
	10 (100%)	5
20	5 (50%)	26
	8 (80%)	12
	10 (100%)	3
30	5 (50%)	25
	8 (80%)	13
	10 (100%)	3

High numbers of common minutiae points are obtained when the occurrence percentage is small (50%). A large number of minutiae points can be correlated to a larger input domain for the biometric key generation process, which ensures the uniqueness of the process. However, the tradeoff is that the probability of occurrence of the same set of minutiae points when re-scanning the fingerprint to generate a new biometric key is not very high.

This research considers that 10 fingerprint templates are sufficient for determining the set of common minutiae points that will occur with a high probability in other scans, as well. The threshold considered was 80%.

2.2.2 Analysis of novel biometric key generation algorithms

The proposed biometric key generation process follows the steps below:

Step 1. Scan fingerprint i_{max} times; let $S = \bigcup_{i=0}^{i_{max}} T_i$ be the set of all templates T_i, with $0 \leq i \leq i_{max}$

For each $T_i \in S$, having $0 \leq i \leq i_{max}$, execute steps 2-4.

Step 2. Process the image representing T_i to obtain its corresponding byte stream, FMD_i, in ISO-IEC-19794-2 Finger Minutiae Data (FMD)

Step 3. Parse FMD_i and run the feature extraction algorithm

Step 4. Calculate the rigid transformation matrices TS_i(translation) and RT_i (rotation) and apply TS_i and RT_i, to align T_i so that the core point of each T_i coincides with a point

of specific coordinates, and the templates have the same orientation; $T_i = T_i \oplus TS_i \oplus RT_i$

Step 5. Overlap all the transformed matrices $T_i \in S$; it results a super-template T, containing all the features from each $T_i \in S$

Step 6. Apply DBSCAN on the super-template T to determine the clusters of minutiae points of the same type K_j, with $0 \leq j \leq nc$, where $nc = number\ of\ clusters$

Step 7. For each cluster K_j, calculate its center point C_{K_j}, and the distance from the center point to the core C; the distance is $d_{K_j} = dist(C_{K_j}, C)$

Step 8. Approximate each C_{K_j} with a value v_{K_j}

Step 9. Generate the biometric key based on the values v_{K_j}

Step 1 is performed on a set of multiple scans of the same finger in order to obtain a higher degree of repeatability of the biometric key. Generating the key based on a single scan may lead to erroneous result. The use of several biometric measurements ensures that only the common features are used in the process, which leads to a more robust process.

The methodology was tested with $i_{max} = 10$ samples of the same fingerprint, but the threshold can be set to a different number, depending on aspects such as the image quality of the fingerprint reader. The chosen fingerprint reader was Zvetco P5000. Griaule's Fingerprint SDK 2009 was also used for extracting the fingerprint's features, which are stored in an ISO-IEC-19794-2 Finger Minutiae Data (FMD) [FMD11] record.

The number of scans taken directly influences the result of the proposed algorithm. A larger i_{max} leads to a more accurate result. However, patients are reticent at the idea of providing their fingerprints so many times. The fingerprint reader used provides the means to automatically record different fingerprint scans without having to remove the finger from the surface, which eliminates patients' discomfort.

Step 2 represents the image processing of each template T_i scanned by the fingerprint reader. Griaule SDK was used for preprocessing and processing the image in order to obtain the biometric features, stored in a standardized format.

Step 3 is represented by the parsing of the FMD byte streams of each T_i and extraction of the cores' and minutiae's coordinates, angles and types. The information stored in the byte stream was translated to a matrix that maps 1:1 with the image of the template T_i. This transformation is a bijective function that maps T_i to a matrix.

The image area of Zvetco P5000 is 256x360 pixels. Every pixel in the image is mapped to a matrix element. Thus, 256x360 matrices are created. Figure 2.2 shows the resulting mapped matrix.

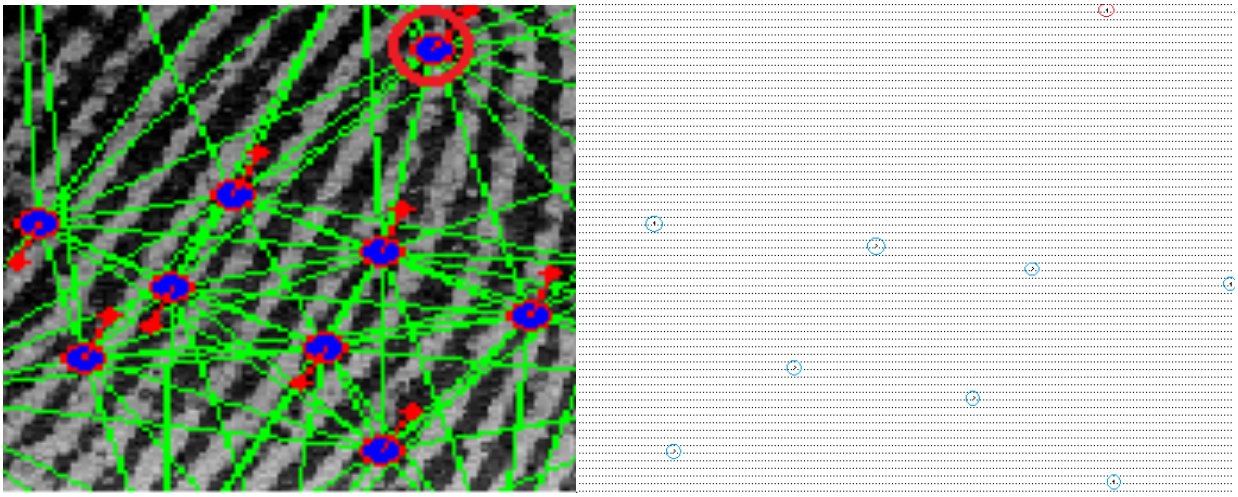

Figure 2.2. Mapping minutiae from the feature space

Step 4 carries two transformations applied to the matrices, which aim to align the templates so that their core points overlap to a predefined position and they have the same orientation. Firstly, the translation TS_i is applied to the template T_i so that the core point C is translated to $P(x, y)$, with $x = \frac{x_{max} - x_{min}}{2}$ and $y = \frac{y_{max} - y_{min}}{2}$, where x_{min}, x_{max}, y_{min} and y_{max} represent the coordinates of the Regions of Interest (ROI). Secondly, the rotation RT_i is applied. This transformation ensures that the templates T_i have the same orientation. The rotation is performed so that the direction of the core point C (marked with a red circle in Figure 2.2) is aligned with the vertical axis.

In **Step 5**, the transformed templates T_i are overlapped. By overlapping the templates, a single template T, called super-template, is obtained, which contains the minutiae points of every $T_i \in S$. The minutiae points of the super-template T tend to accumulate in clusters, forming a sparse matrix with high densities of points in various regions. This is helpful in determining which minutiae points are relevant to the key generation algorithm, and what their estimated coordinates are. Figure 2.3 depicts the clustering behavior of the minutiae.

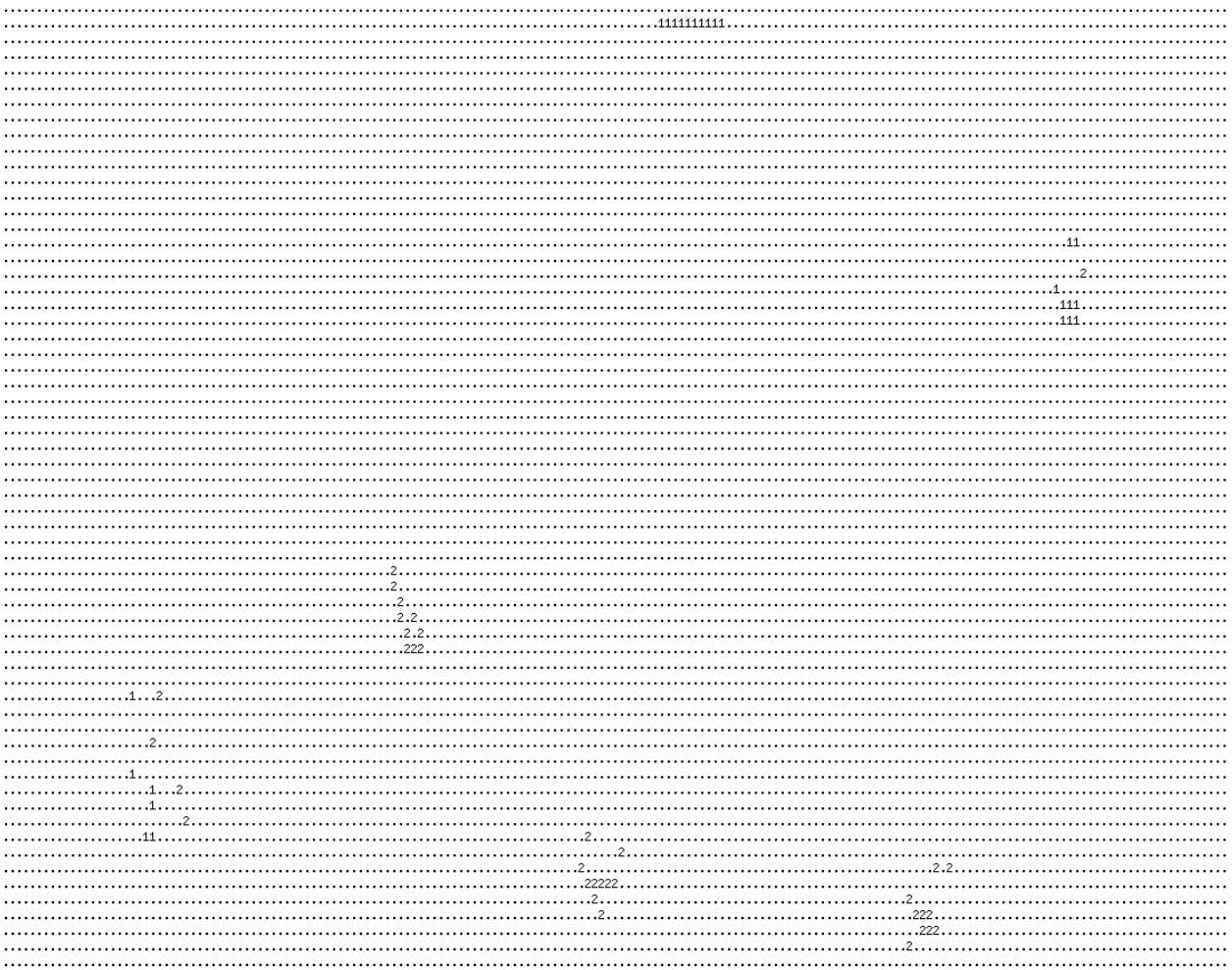

Figure 2.3. Same minutiae points from different scans tend to form clusters

Step 6 is responsible for clustering the minutiae points. The minutiae points that represent ridge endings have been mapped to 1, while the ones representing ridge bifurcations have been mapped to 2. A clustering algorithm, i.e. DBSCAN, is used. The clustering algorithm detects the minutiae points of the same type (mapped either to 1 or 2) that are located in the same proximity.

A validation process is done in order to eliminate inadequate clusters. A cluster is validated only if it meets the following criterion: $0.8 * i_{max} \leq card(K_j) \leq i_{max}$, which means that the number of elements in the cluster K_j is sufficiently large to consider that the cluster represents an area of the image where minutiae points will be detected in future scans as well, with a very high probability. Figure 2.4 depicts the clusters obtained after running the DBSCAN algorithm.

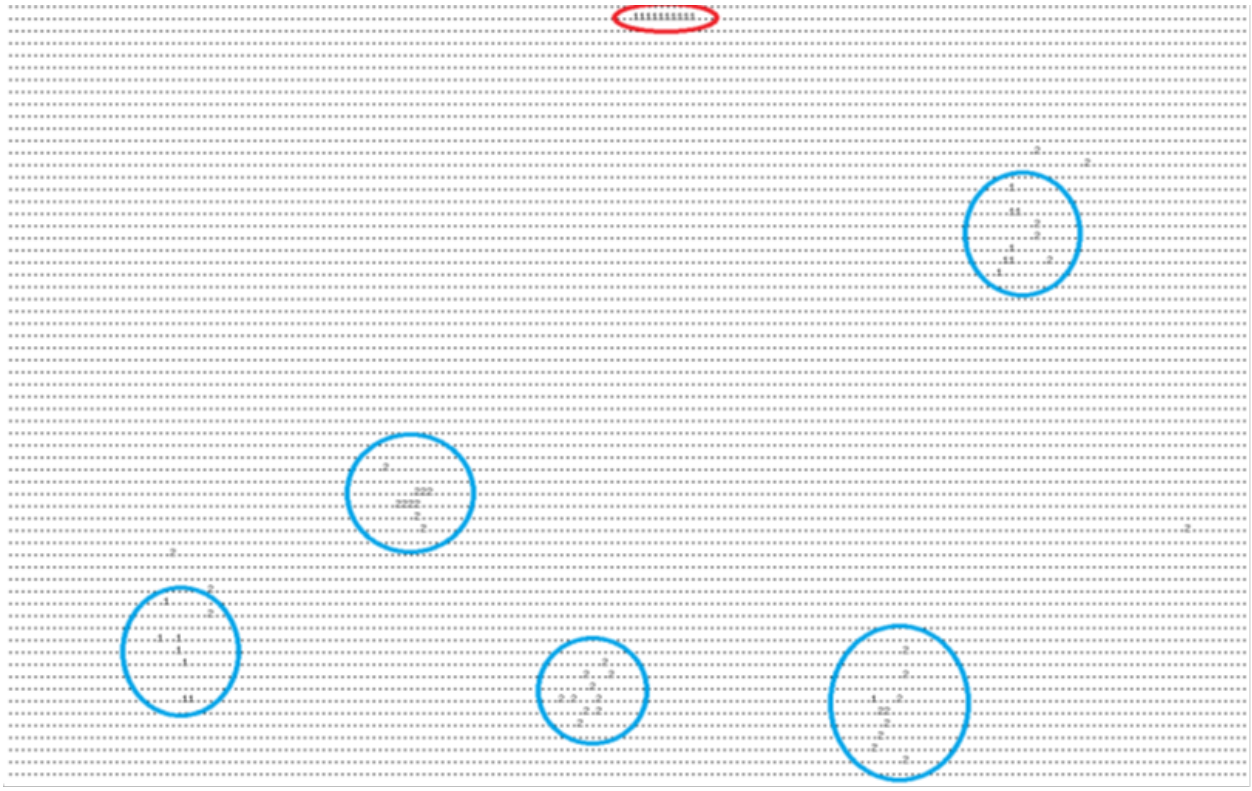

Figure 2.4. Clusters of common minutiae points from 10 different scans

Step 7 calculates the points C_{K_j} that best represent all the minutiae points contained in the cluster K_j. Initially, an average of all the minutiae of the cluster is calculated, based on the arithmetic mean of their coordinates. Then, the distance from each point of the cluster K_j to the one determined by the average coordinates is obtained. Based on this distance, each point of the cluster is assigned a different weight. Bigger weights are assigned to the points closer to the average. Finally, a new average C_{K_j}, named the center of the cluster K_j, is calculated, this time using a weighted mean.

After determining the centers C_{K_j}, the Euclidean distance between these points and the core is computed. The distance is calculated as $d_{K_j} = dist(C_{K_j}, C)$.

Step 8 is the most important step of the biometric key generation process. It involves approximating each of the C_{K_j} points in a way that is invariant to the inherent changes of the topology of the minutiae points. Several methods have been investigated.

Method 1: static division of the feature space

The first attempt to approximate the values of C_{K_j} implied dividing the feature space in a polar coordinate system statically [EGN12a], as shown in Figure 2.5. The core represents the pole of

the coordinate system, while the polar axis may be chosen from either the vertical axis or the orientation of the core. Thus, each area is defined by its radial and angular coordinates.

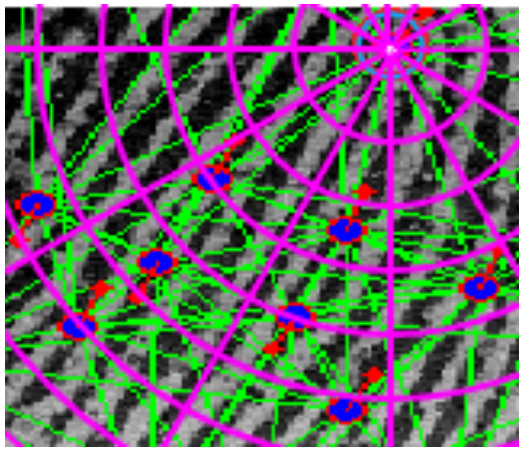

Figure 2.5. Static division of the feature space

Each cluster center C_{K_j} is assigned a value, according to the area that contains it. The expectation was that even though the centers have small offsets, these wouldn't affect the membership of the centers to areas, so that centers would always be mapped to the same value, thus generating the same key.

However, experiments showed that the algorithm is affected by the center points located in the proximity of a border between two areas. These points may be mapped to one value, when generating a biometric key and to another value when regenerating the key, thus lowering the reliability of the process.

A better approach is the approximation of each C_{K_j} in such way that is invariant to the small deviations that inherently occur. The closest solution is to approximate each C_{K_j} based also on the topology of the other C_{K_j} points.

Method 2: Voronoi diagrams

As opposed to the first method, which implies a static decomposition of the feature space, this method relies on a dynamic decomposition determined by the distances between the clusters' centers. There are several variations of the Voronoi decomposition [OKA00], based on the distances used. This research has focused on the standard Voronoi tessellation that uses the Euclidean distance. An example of Voronoi tessellation is shown in Figure 2.6.

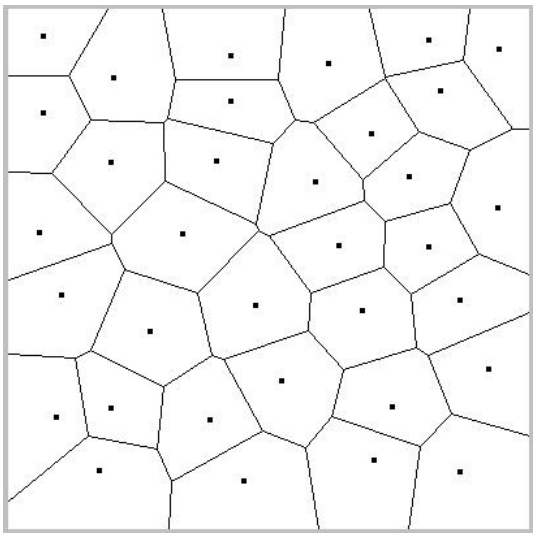

Figure 2.6. Example of Voronoi decomposition

(source: Washington University in St. Louis [VOR])

This method is better than the first one, since the feature space is dynamically decomposed and the problem of border-located points is solved. Moreover, data about neighbors can also be involved in the formula used for calculating the approximation values. This leads to small corrections of the offset, since the error is dispersed locally.

Several formulas have been tried for the approximation. Best results were obtained when mapping each center C_{K_i} to a value calculated as:

$$v_{K_i} = \left[\sum_{p \in P_{K_i}} S_p \right] * d_{K_i}, \text{where}$$

- K_i – cluster obtained after the DBSCAN clustering (**Step 6**);
- C_{K_i} – center of the cluster K_i;
- P_{K_i} – set of Voronoi polygons that are neighbors with the polygon that contains C_{K_i};
- S_p – area of the Voronoi polygon p;
- $d_{K_i} = dist(C_{K_i}, C)$ – distance between the center of the cluster K_i and the core C.

The results showed improvements over the first method [EGN12b]. Its advantage over the first method lies in the fact that each center is mapped to a value that depends on the center's location in the feature space and on the topology of its neighbors. However, there are scenarios where the process is affected by rare configurations of the minutiae points. As shown in Figure 2.7, there are situations where minutiae are isolated and the influence of their neighbors is insignificant.

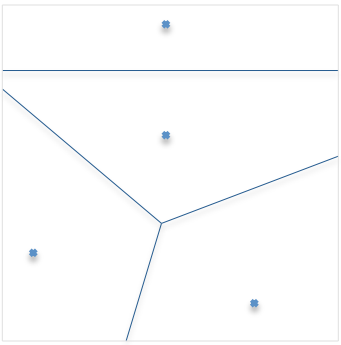

Figure 2.7. Isolated minutia point

Method 3: Delaunay triangulation

This methodology is based on the Delaunay triangulation [HJE06], which is used for representing the topology of the center points and for calculating invariants that contribute to the biometric key generation process. An important property of Delaunay triangulation is that insertion of a new point affects only the triangles whose circumcircles contain that point. Thus, noise affects the Delaunay triangulation only locally. Figure 2.8 shows an example of a Delaunay tessellation.

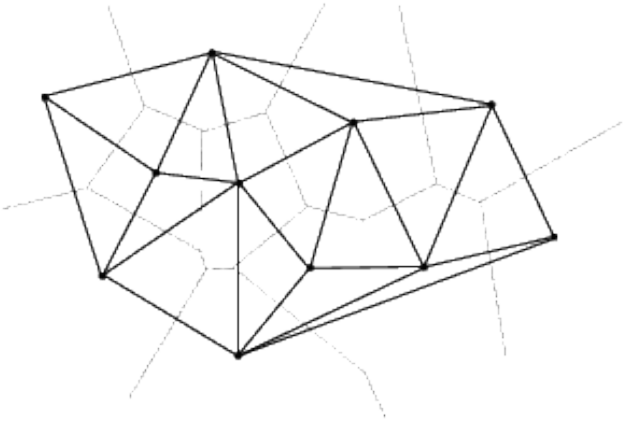

Figure 2.8. Example of Delaunay triangulation

(source: R.B. Muhammad, Kent State University [MUH])

Step 8 handles the Delaunay triangulation. The set of cluster centers C_{K_j} is fed to the algorithm, which calculates the minutiae topology.

Two invariants, namely I_{t_1} and I_{t_2}, are computed for each triangle $t \in D$. The rigid transformations applied in **Step 4** do not affect the values of I_{t_1} and I_{t_2}. The three edges of the triangle are first sorted, so that $l_1 \leq l_2 \leq l_3$. The invariants are then calculated:

$$I_{t_1} = \frac{l_1}{l_3} \text{ and } I_{t_2} = \frac{l_2}{l_3}.$$

Each cluster center C_{K_j} is then mapped to a value that depends on the two invariants. This is performed so that each center's value is also influenced by the location of the center and the topology of its neighbors.

The value is being calculated as $v_{K_j} = \left[\sum_{t \in D}\left(I_{t_1}, I_{t_2}\right)\right] * d_{K_j}$, where I_{t_1} and I_{t_2} represent the invariants of the triangle $t_l \in D$, which contains the vertex C_{K_j}. Thus, every triangle that contains the center C_{K_j} affects its mapping value. Also, the distance between C_{K_j} and the core is determinant.

Table 2.4 shows how minutiae points are approximated to values that depend on the sum of the invariants of each Delaunay triangle that contains C_{K_j} and the distance between C_{K_j} and C. The C_{K_j} points represent the centers of the clusters in the first super-template. The approximation of every v_{K_j} is then used to form the biometric key, which is performed as simple concatenation: **57241937564049544150352 7**.

Table 2.4. Approximation values for the clusters' centers – first super-template

Minutiae No	$\sum_{t \in D}\left(I_{t_1}, I_{t_2}\right)$	d_{K_j}	v_{K_j}	Approximation of v_{K_j}
C_{K_1}	4.47664129	127.186	569.366099	57
C_{K_2}	4.17872274	56.499	236.093656	24
C_{K_3}	5.79147003	31.515	182.518178	19
C_{K_4}	6.86866323	53.457	367.17813	37
C_{K_5}	6.05255423	90.992	550.734015	56
C_{K_6}	7.21171642	55.103	397.38721	40
C_{K_7}	6.8925385	70.174	483.676997	49
C_{K_8}	6.52878346	81.884	534.602904	54
C_{K_9}	5.32188173	76.903	409.268671	41
$C_{K_{10}}$	6.17688044	80.867	499.505791	50
$C_{K_{11}}$	3.18630518	108.535	345.825632	35
$C_{K_{12}}$	4.1857846	63.086	264.064407	27

Table 2.5 presents the results of generating the biometric key with a different set of templates T_i. The key generated in this case is similar to the one generated based on values contained in Table 2.4: **57241937564049544150352 7**.

Table 2.5. Approximation values for the clusters' centers – second super-template

Minutiae No	$\sum_{t \in D}(I_{t_1}, I_{t_2})$	d_{K_j}	v_{K_j}	Approximation of v_{K_j}
C_{K_1}	4.5029157	125.842	566.655917	57
C_{K_2}	4.16325008	57.556	239.620022	24
C_{K_3}	5.82412615	31.893	185.748855	19
C_{K_4}	6.79934665	54.022	367.314305	37
C_{K_5}	6.01843154	91.606	551.32444	56
C_{K_6}	7.12296182	55.536	395.580808	40
C_{K_7}	7.01495015	68.956	483.722903	49
C_{K_8}	6.37341493	84.349	537.591176	54
C_{K_9}	5.28300442	77.172	407.700017	41
$C_{K_{10}}$	6.0700612	82.118	498.461285	50
$C_{K_{11}}$	3.17884488	107.821	342.746234	35
$C_{K_{12}}$	4.20687909	62.769	264.061594	27

2.3 Authentication protocol for ISO/IEEE 11073-20601

There are many solutions that can be adopted for implementing authentication procedures between Agents and Managers. With systems that employ mutual authentication procedures, unauthorized Agents cannot send medical data to Managers and unauthorized Managers cannot collect clinical data communicated by Agents. Thus, authentication procedures would prevent:

- storage of erroneous patient data;
- medical data leakage.

The authentication procedure represents the first step in assuring security of medical data communication between mobile medical devices. While authentication is an important security

issue for OEP, other mechanisms such as discovering identities or setting roles and privileges should also contribute to the achievement of a secure identity management system. Identity recognition is especially important, since transferred medical data must be correlated to patients when Agents are shared among them.

An authentication mechanism based on biometry would meet the two objectives, i.e. assuring the authentication and the identity recognition. This section presents a mechanism for reliably identifying patients when using Agent medical devices. The identification procedure is based on fingerprint measurements. This approach ensures authentication and identity recognition, which are two key security features the protocol lacks.

General approaches on authentication

Persons usually prove their identity by one of the following:

- something they know;
- something they have;
- something they are.

Something they know refers to any type of knowledge persons have: usernames, passwords, pins, etc. In a medical environment, however, this type of information is not reliable. There may be cases where patients are unconscious, in shock, in a coma, or have mental disorder. A type of authentication based on the knowledge of such patients is inappropriate.

Something they have refers to an item that a person carries. It can be a key, a card, a token, etc. An authentication procedure based on something the patient has is dependent on the availability of the item used in authentication. Moreover, this type of authentication requires special devices, such as readers for the authentication items, which may be difficult to embed in small medical devices such as thermometers.

Something they are refers to unique psychological or behavioral features that identify a person. The characteristics of these features recommend them for the authentication procedure. They cannot be lost, stolen, forgotten, guessed, or shared. Furthermore, capturing devices such as fingerprint readers can be easily embedded in medical devices. In terms of the psychological factor, patients accept systems based on biometric measurements more easily in a medical environment.

OEP specific authentication requirements

This section presents a novel method of generating biometric keys based on fingerprints. The method was designed for authentication between ISO/IEEE 11073- 20601 devices. Thus, the authentication was constrained by the following requirements:

R1. *The biometric data acquisition process should not be expensive*

The devices used for acquiring the biometric data used in the algorithm should not be more expensive than the OEP devices. The cost of processing the acquired biometric data should be considered when designing the solution, as well.

R2. *The biometric data scanners must be small enough to be integrated in the OEP devices*

The solution should be based on biometric data that can be acquired with the help of small, embeddable readers. For instance, palm or vein scanners are not suitable for integration in small devices such as thermometers.

R3. *The key generation protocol depends only on the biometric data acquired by the scanner*

The solution should not depend on classifiers that have to be initially trained with large sets of biometric scans. The literature describes various solutions for generating biometric keys based on classifiers such as Support Vector Machines (SVM). However, classifiers are not flexible and need to be adapted every time new biometric data is acquired.

Using biometric keys derived from fingerprints

The main motivation for using biometry in the authentication process comes from the need for authenticating not only devices, but patients, as well. This section presents several advantages of involving biometry in the authentication process.

Identification vs. Verification

In a classical approach, authentication is done either by identification or verification. When a biometric verification is performed, a person's feature is compared with another feature from the database. The matching is one-to-one. Verification is usually used to check if the claimant is the person he/she pretends to be. Identification, on the other hand, implies matching the person's feature to all the features stored in the database. The matching is, in this case, one-to-many. Identification is used for detecting the identity of a person from a group of candidates.

Fingerprint characteristics

There are several techniques for measuring characteristics associated to fingerprints. Two of the most common are minutiae-based and image-based. The approach herein relies on using the minutiae representation of the fingerprints. Aging, injuries, or diseases can alter fingerprints. However, our approach ensures that fingerprints are constantly updated, to avoid unnoticed changes that could affect the authentication process.

Accuracy

The accuracy of an authentication system based on biometry is measured using two parameters: false rejection rate (FRR) and false acceptance rate (FAR). *FRR* shows how often an entity is incorrectly denied authentication, while *FAR* shows how often an entity is incorrectly authenticated.

Approaches on fingerprint-based authentication

In the OEP context, several aspects must be considered when choosing the method of authentication:

1. The first consideration is that authentication is done at the Manager, since this entity is being responsible for ensuring identity management. This raises the problem of sending authentication information through an unsecure communication channel, which is vulnerable to exploits from eavesdroppers.
2. The second consideration is that fingerprint templates cannot be revoked. Unlike cards or passwords that can be canceled or replaced, fingerprint templates are permanently associated with a person and cannot be restored. Thus, the authentication process based on fingerprint measurements should be designed in such a way so that an attacker cannot obtain any biometric data by inspecting captured messages.
3. The third consideration is that the authentication mechanism should be mutual, i.e. Agents and Managers authenticate to each other. This is an important constraint. Usually, authentication based on biometry reduces to verification or identification, which are not mutual. Thus, an alternative to the classical identification based on pattern matching of fingerprint templates should be considered.

Considering the three objectives, the thesis proposes an approach based on the *mutual challenge-response authentication* procedure. In the *mutual challenge-response authentication* entities prove the knowledge of a secret, which can be a password, or a pre-shared key, without actually sending it to the other party. The thesis proposes that the secret is a biometric key, *BIO_KEY*.

This type of authentication offers little information to attackers since the challenge is changed each time authentication is required. The solution should be implemented in a session-based manner to inhibit attackers to run brute-force or dictionary attacks.

The *BIO_KEY* used in the mutual challenge-response authentication is generated based on fingerprint minutiae. This key is not pre-shared between the Agent and the Manager. It is generated each time the authentication process starts, i.e. whenever the patient uses the Agent device. The *BIO_KEY* is used when computing the response to the other entity's challenge. In other words, the biometric data is not used just for identifying patients. Instead, it is used for generating a biometric key that acts as any cryptographic key. The difference between the *BIO_KEY* and other cryptographic key is that the former is directly linked to the patient and cannot be shared.

The inherent features of the biometric data raise the question of uniqueness and repeatability of the *BIO_KEY*. In general, there are differences between subsequent biometric measurements, which make the key generation process a difficult challenge. This topic has been the focus of intensive research lately and although is still an open problem, studies show optimistic results. Bhargav-Spantzel et al. [BHA10] showed identical keys can be generated for different biometric

readings. Several other methods for generating the biometric key, including our novel approach are discussed in Section 2.2.

This protocol procedure assures both authentication and identity discovery. It does not significantly affect the number and size of exchanged messages and it does not require high processing power. Recently developed wireless controllers allow implementation of the OEP protocol and different Bluetooth profiles such as Health Data Profile (HDP) in small, embeddable chips such as the ones developed by Bluegiga [BLU]. Consequently, it is expected that the processing power of medical devices will be able to satisfy the requirements needed for capturing fingerprint images and generating the biometric key.

2.3.1 Mutual challenge-response authentication

With respect to ISO/IEEE 11073-20601 environment, the authentication mechanism plays a vital role in establishing the source of the transferred clinical data. Authentication is seen as a process in which the Manager recognizes and authorizes the tuple consisting of the Agent and the patient. Two aspects that compose this authentication can be distinguished:

- the mutual authentication between devices (Agent and Manager);
- the association between the device (Agent) and the patient who uses it.

The first phase can be implemented following the existing specifications of the OEP protocol. Agents authenticate to Managers through the *association* mechanism, which is already defined in OEP. The communication between the two devices can start only if an association between the Agent and the Manager is performed beforehand. This section discusses the second phase of the authentication: the association between the patient and the Agent.

Let's consider the case where one Agent is located in a hospital room and records data from several patients for a defined period of time. The data recorded by this Agent must be assigned to each individual patient of that room. The simplest way to do that is to link the patient with the Agent whenever the patient uses the device, so that the Manager can establish a correspondence between the received medical data and the patients.

Another case where the link between patients and devices is needed is when patients want to be sure the device used acquires their data exclusively. To ensure security of medical data, patients may want to block Agents from external use, enabling them for acquiring their medical data, exclusively. The result is that the patient data history recorded by the Manager is updated with proper medical information.

The link between patients and Agents is established with the help of the Manager. One of the reasons for this is that the Agent has less computational resources than the Manager. Another reason is that in this situation the Manager can implement device management procedures, or access policies based on the patient that uses the devices.

We refer to the link between the Agent and the patient as *patient authentication*. A patient is considered authenticated when the Manager acknowledges his use of a specific Agent and grants him a set of usage permissions.

The authentication mechanism proposed is based on the *challenge-response authentication* protocol, which follows the next steps:

Step 1. Alice and Bob ensure they have the same biometric key (*BIO_KEY*). This key can be individually generated by each of them, or exchanged through a trusted communication channel, before the authentication starts
Step 2. Alice asks Bob a question (sends a challenge)
Step 3. Bob responds to the question. The response is calculated based on the challenge and *BIO_KEY* they share
Step 4. Alice calculates its own response to the same question. If its response matches the one received, it authenticates Bob

The presented authentication mechanism is unilateral. As discussed before, a mutual authentication mechanism would provide a higher degree of security in terms of exchanging medical data. This study proposes, therefore, that the authentication procedure between the Agent and the Manager is a *mutual challenge-response authentication* [EGN12a]. The message flow in a *mutual challenge-response authentication* is described in Figure 2.9.

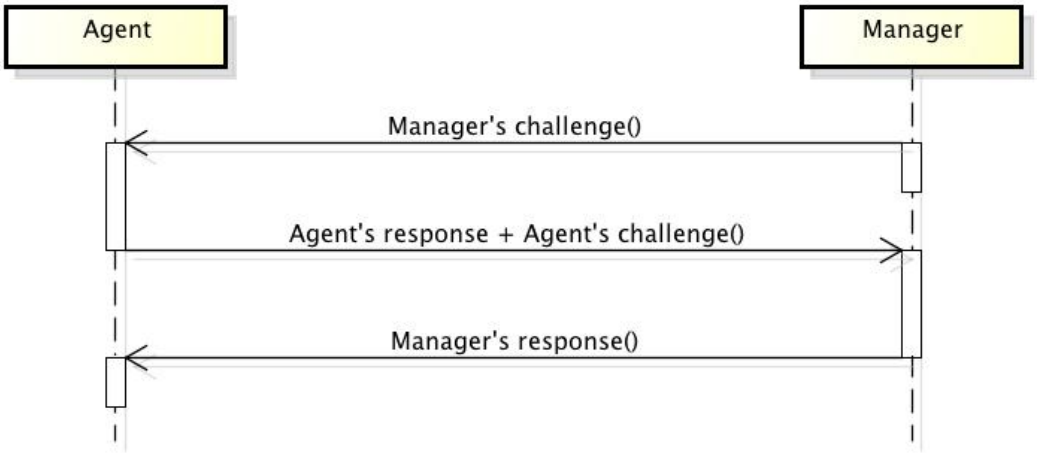

Figure 2.9. Overview of the mutual challenge-response authentication protocol

Before describing the authentication protocol, the following terms are defined:

- let Σ be an alphabet, defined as $\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$;
- let Σ^l be the set of all strings of length l over Σ;

- let ch_{gen} be a challenge generator function, defined as $\boldsymbol{ch_{gen}}: \mathbb{N} \rightarrow \boldsymbol{\Sigma^l}$, with $\boldsymbol{ch_{gen}(n)} = \boldsymbol{RN_n}$, where RN_n is a random value from Σ^n.

In the following exemplification, random number challenges of 64 bits are considered. The *mutual challenge-response authentication* consists of the following steps described in Table 2.6.

Table 2.6. The steps of the mutual challenge-response algorithm

Manager	**Step 1:** The Manager generates a challenge C_M $C_M = ch_{gen}(64)$ **Step 2:** The Manager sends challenge C_M to the Agent
Agent	**Step 3:** The Agent generates its own challenge C_A $C_A = ch_{gen}(64)$ **Step 4:** The Agent calculates response R_A $R_A = hash(C_A \oplus C_M \oplus BIO\_KEY)$ **Step 5:** The Agent sends the Manager an authentication message containing the response R_A and its challenge C_A
Manager	**Step 6:** The Manager calculates the expected response ER_A to its own challenge C_M, using as input the challenge C_A received from the Agent $ER_A = hash(C_A \oplus C_M \oplus BIO\_KEY)$ **Step 7:** The Manager compares the responses R_A and ER_A. If they match, the Manager authenticates the Agent **Step 8:** The Manager calculates response R_M $R_M = hash(C_M \oplus C_A \oplus BIO\_KEY)$ **Step 9:** The Manager sends the Agent an authentication message containing the response R_M
Agent	**Step 10:** The Agent calculates the expected response ER_M to its own challenge C_A, using as input the challenge C_M received from the Manager $ER_M = hash(C_M \oplus C_A \oplus BIO\_KEY)$ **Step 11:** The Agent compares the responses R_M and ER_M. If they match, the Agent authenticates the Manager

The challenge used in the *mutual challenge-response authentication* is a cryptographic nonce. It is used to ensure that each authentication sequence is unique. Even if an eavesdropper intercepts

messages exchanged between the Agent and the Manager, it cannot replicate the messages and authenticate itself using the same responses.

Since entities involved in the authentication do not reveal the keys, it is impossible for eavesdroppers to derive responses from previously exchanged messages. These facts protect the protocol against replay attacks.

The communication should also be session-based, meaning that the authentication allows message exchange only for defined period of times, such as 5 or 10 minutes. The length of a session should be set by the implementer of the protocol and should differ from one device to another. For instance, it takes much less time to measure the temperature with a thermometer than to filter the blood with a dialysis machine.

The mutual challenge-response protocol may be vulnerable to dictionary and brute-force attacks. However, if the response is long enough, i.e. 256 or 512-bits, the probability of guessing the response during a session is nearly zero.

The factors that influence the mutual challenge-response authentication process are the keys, the challenges, the cryptographic hash function and the operator $\oplus$. Each of these parameters is analyzed and guidelines on how they should be used are offered below. The key generation is discussed in Section 2.2.

The challenge is a time-varying value, usually in form of a unique random number, which influences the output of the cryptographic hash function. A new challenge is generated for every authentication attempt, thus making replay attacks virtually impossible. The challenge should be at least 64-bits random number and the generating function should ensure a probabilistically insignificant chance of repeating a previously generated number.

The cryptographic hash function maps input strings of arbitrary lengths (usually large strings) to fixed length outputs, i.e. hash values. Hash functions must satisfy at least the following three criteria:

- preimage resistance;
- second preimage resistance;
- collision resistance.

A hash function h is *preimage resistant* if given $y = h(M)$, it must be extremely difficult for an eavesdropper to find any message M' so that $y = h(M')$. If the hash function is not preimage resistant, an eavesdropper can intercept the digest $h(M)$, create a new message M' having the same digest and send it to one of the entities pretending it was message M.

A hash function is *second preimage resistant* when messages are extremely difficult to forge. Given a message M and its digest $h(M)$, an eavesdropper is unlikely to find a message $M' \neq M$ such that $h(M') = h(M)$ if this criterion is met.

The third criterion, the *collision resistance*, ensures that eavesdroppers cannot find two messages that have the same digest. In other words, it is very difficult for an eavesdropper to find two

messages M and M' so that $h(M) = h(M')$. This is a very important property of the hash function.

In general, the complexity of an attack to the hash function is proportional to the digest length. MD5 has been proven to be vulnerable to collision attacks. Whirlpool or SHA-512 are good hash function choices, both producing 512-bits digests.

The operator $\oplus$ used in the cryptographic hash function should form a non-abelian group with the set of all possible challenge numbers and biometric keys. It is important that the operator $\oplus$ is not commutative, so that the results of $C_A \oplus C_M \oplus BIO\_KEY$ and $C_M \oplus C_A \oplus BIO\_KEY$ are different.

The above analysis of the parameters involved in the mutual challenge-response authentication only offers guidelines or recommendations for implementers. However, there are cases when implementers may decide to approach different solutions.

2.3.2 Authentication scenario

To illustrate a complete authentication scenario, the case where several Agents and one Manager are part of a hospital environment is considered. The Agents have different specializations and are shared among several patients. The Manager collects and stores medical data from all the Agents.

2.3.2.1 Registration

When admitting into the hospital, during the registration phase, the patient is required to place a finger on a fingerprint reader device. The fingerprint template will be used for identifying the patient when using the Agent devices. The Manager uses the fingerprint template to generate the *BIO_KEY*, which will be used in the *challenge-response authentication* procedure. The *BIO_KEY* and other patient information are stored in a database. The fingerprint template is not stored anywhere, to avoid possible attacks and the *BIO_KEY* is generated in such way that it does not provide any information about the patient. This type of registration is suitable for patients who are unconscious, in shock, or in a situation where they cannot communicate.

2.3.2.2 Authenticating and using the Agents

After admitting the patients, different Agents may be used for monitoring them. The acquiring medical data process follows the next steps:

Step 1. The patient places the finger on the fingerprint reader embedded in the Agent. It must be the same finger used in the registration phase

Step 2. The Agent uses the provided template to generate the *BIO_KEY*

Step 3. The Agent tries to authenticate to the Manager. The authentication mechanism used is the *mutual challenge-response authentication*. The *BIO_KEY* is used when calculating the response to the Manager's challenge

Step 4. The Manager calculates one possible response for each already registered patient. If the response from the Agent matches one of the expected responses, the patient is identified and the Agent is authenticated

Step 5. The Agent authenticates the Manager, as well

Step 6. A new communication session starts

From this point onwards, the Agent and the Manager can continue with the normal flow of OEP message exchange. Data may be acquired as long as the session does not expire. If it does, the Agent needs to re-authenticate.

2.3.2.3 Discharge

After the patient is discharged, his record in the database is removed. This way, the database is kept small, increasing performances of the biometric identification process. If the same patient returns to the hospital, a new process of registration would be required. This assures that the patient database is updated to reflect any change in the biometric characteristics.

3 SECURITY EXTENSIONS FOR ISO/IEEE 11073-20601

3.1 Terminology and definitions

Overview of the ISO/IEEE 11073 standard suite

ISO/IEEE 11073 is a family of standards that enable communication between mobile medical devices and external information systems. It contains the following important standard specifications, as described in Figure 3.1:

- ISO/IEEE 11073-10101: *Nomenclature*, which defines the vocabulary of terms used in Medical Device Communication (MDC);
- ISO/IEEE 11073-10201: *Domain Information Model (DIM)*, which contains the definition for structuring information transferred between entities;
- ISO/IEEE 11073-104zz: *Device Specialization*, which defines specific medical device specialization. For example, the device specialization for a Blood Pressure Monitor is ISO/IEEE 11073-10407, while the specialization for a Glucose Meter is ISO/IEEE 11073-10417;
- ISO/IEEE 11073-20601: *Application Profile - Optimized Exchange Protocol*, which defines the communication protocol for exchanging medical data.

For readability reasons, the standard "ISO/IEEE 11073-20601: Application Profile – Optimized Exchange Protocol" will be henceforth referred to as the ISO/IEEE 11073-20601 and the communication protocol defined within this standard will be referred to as the OEP (Optimized Exchange Protocol).

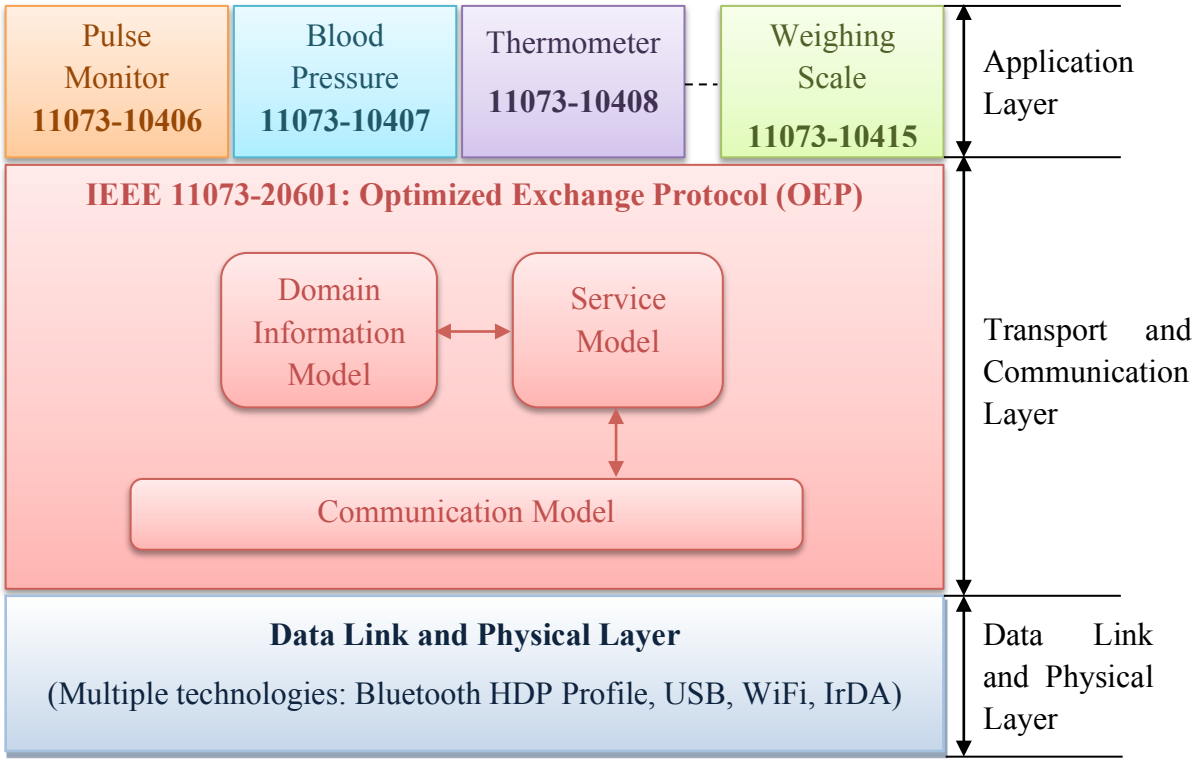

Figure 3.1. The ISO/IEEE 11073 standard suite

Entities

The ISO/IEEE 11073-20601 standard defines a point-to-point communication protocol between two entities, which are called Agent and Manager.

The Agent represents the device that collects personal health data directly from patients. It can be a thermometer, a blood pressure monitor, etc. In order to define the behavior of these heterogeneous devices, specific medical device specializations were defined: the ISO/IEEE 11073-104zz specializations.

The Manager represents the device that collects personal health data from the Agents. The Manager is represented by a local hosting device, which can be a smartphone, a notebook, a PC, etc.

Architecture of ISO/IEEE 11073-20601

ISO/IEEE 11073-20601 [OEP08] defines the means of creating abstract models for communicating personal health data between Agents and Managers over Personal Area Networks (PAN). The standard definition consists of three main sections:

- Domain Information Model (MDIB);
- Service Model (CMDISE + ACSE);

- Communication Model.

These three sections describe the *data model*, the *operations* supported by the entities involved in the communication and the finite state machine describing the *communication process*, respectively. Figure 3.2 shows the relationship between these components.

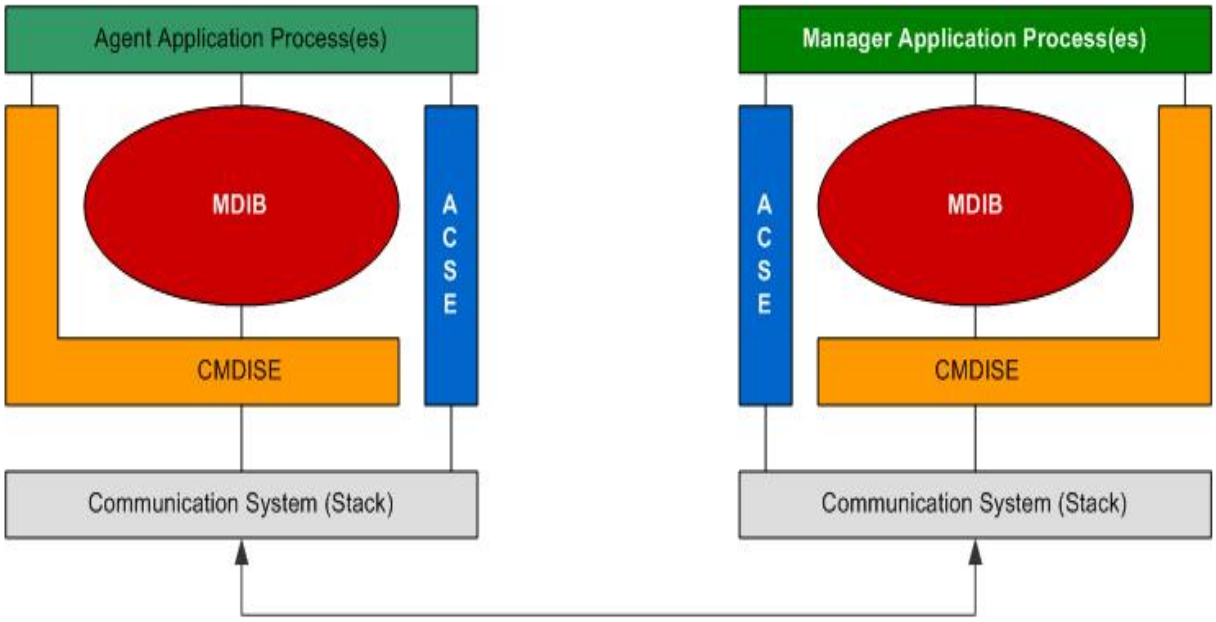

Figure 3.2. Relationship model between Agents and Managers as defined in ISO/IEEE 11073-20601 (adaptation of Figure 6.1 from 11073-10201-DIM [DIM04], pg. 10)

The standard was designed to provide means for short-distance communication, through the use of standardized wireless networks. The OEP protocol is flexible and lightweight. It can be implemented for different types of Agents, even for those with limited processing power and storage space.

Communication model

The OEP communication model supports a one-to-many topology. One Manager can communicate with one or more Agents over point-to-point connections. The communication model is defined by two finite-state machines (FSM), which describe all the possible states the Agent and Manager may be in. Figure 3.3 shows a simplified version of the state machine diagram that describes the communication between the Agent and the Manager (adaptation of the Agent's and Manager's state machine diagrams described in [OEP08] – Figure 10, pg. 57 and Figure 11, pg. 60, respectively).

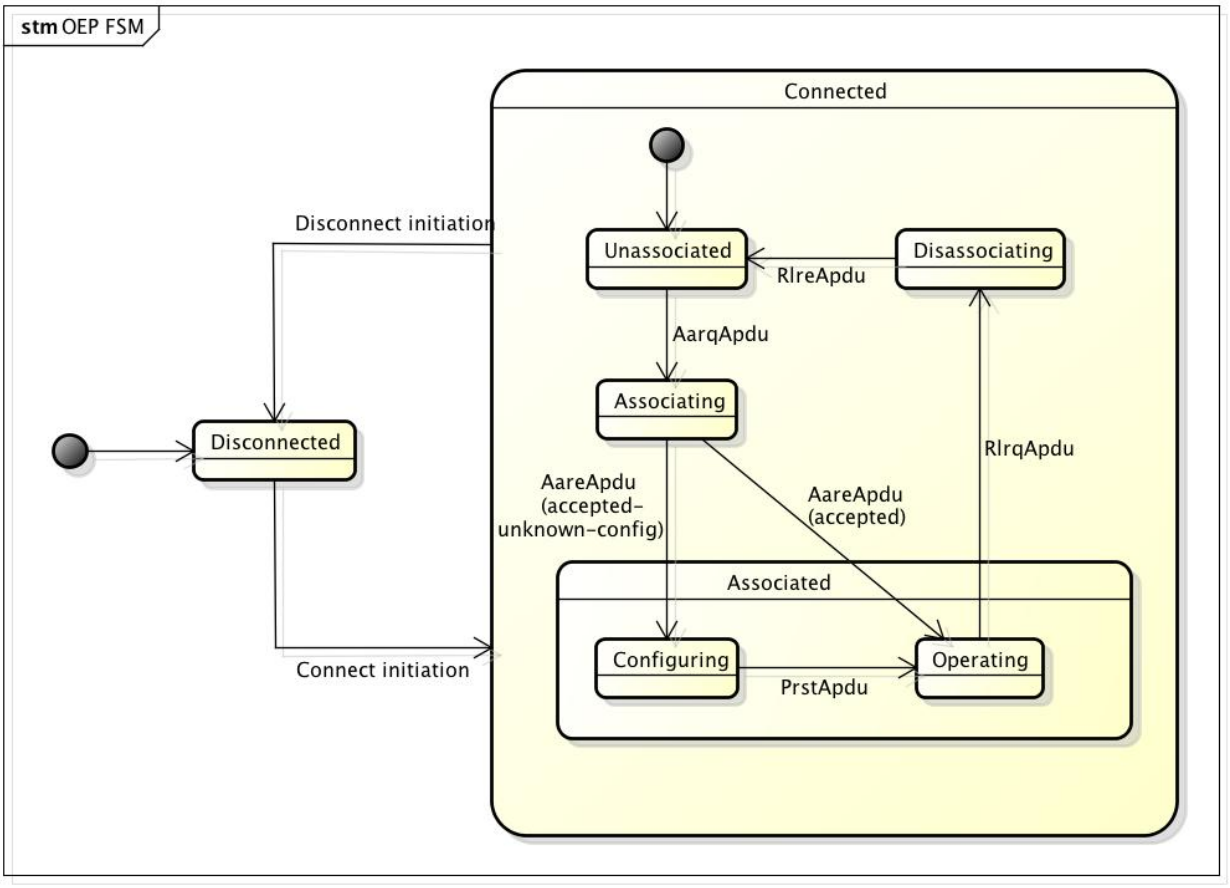

Figure 3.3. Simplified communication state machine diagram

Table 3.1 summarizes the states defined in the OEP communication model (adaptation of the Agent state description from [OEP08] – Table 19, pg. 57-58).

Table 3.1. States defined in the communication model

State	Description
Disconnected	When powered on, the Agent and the Manager start in the *Disconnected* state. At this point, there is no transport connection established between them. After a transport connection is established, it is possible to return to the *Disconnected* state if the transport connection is intentionally terminated or unintentionally disconnected.

State	Description
Connected	When a transport connection is established, the entities transit into the *Connected* state. Initially, they start in the *Unassociated* state, a sub-state of the *Connected* state. The entities remain in the *Connected* state as long as there is a transport connection established between them.
Unassociated	The entities are in the *Unassociated* state if there is no application layer association between them. This situation can occur when: - A new connection was just established - The Manager rejects an *Association Request* - Either party releases or aborts an active association at any time while connected The entities remain in the *Unassociated* state until the Agent sends an *Association Request* to the Manager.
Associating	Whenever the association procedure starts, i.e. the Agent sends an *Association Request* to the Manager, both entities transit to the *Associating* state. If the association fails, but alternative association parameters are possible, the Agent may attempt to associate with each new set of association parameters. If timeout is reached, the Agent may attempt to re-associate until the maximum retry count is reached or the association succeeds.
Associated	When the Manager determines that both entities share a common version of protocol, it sends the Agent an *Association Response* message with an *accepted* parameter and moves to the *Associated* state. When the Agent receives this message, it also moves to the *Associated* state. The entities remain in this state until the Agent sends or receives a release or abort request for the association. The initial sub-state when entering the *Associated* state depends on whether the Manager responded to the *Association Request* with an indication that the Agent's configuration is recognized or not.
Operating	When the Manager recognizes the Agent's configuration, it sends the Agent an *Association Response* with the outcome set to *accepted*. Both entities move into the *Operating* state. Alternatively, if the configuration is not recognized, the configuration must be transferred to the Manager. If the configuration is accepted, the entities enter the *Operating* state.

State	Description
Configuring	When the Manager does not recognize the Agent's configuration, it informs the Agent by sending an *Association Response* with the outcome set to *accepted-unknown-config* to indicate that the association was accepted but that the configuration needs to be transmitted. Both entities remain in the *Configuring* state until the Agent transfers the configuration information and the Manager acknowledges the configuration.
Disassociating	Whenever the Agent determines it should release the current association, it sends an *Association Release Request* to the Manager. Both entities transit into the *Disassociating* state. In case of a timeout, the Agent sends an *Abort Request* and both entities move to the *Unassociated* state.

Configuration

The device configuration of an Agent is a set of objects and attributes that characterize that Agent. The device configuration is associated with a *dev-config-id* value. An Agent may own multiple device configurations, which may change in time. During the association procedure, however, the configuration cannot change. If the association with the Manager is already established, the Agent may change the configuration only by releasing the association first and then re-associating with the new desired configuration.

Association

The associating procedure allows the Agent and Manager to agree on a common data protocol and a common set of operating parameters. Both entities transit from the *Unassociated* state into the *Associating* state when the Agent decides to associate and sends an *Association Request* message. Figure 3.4 shows the sequence diagram of the associating procedure between the Agent and the Manager.

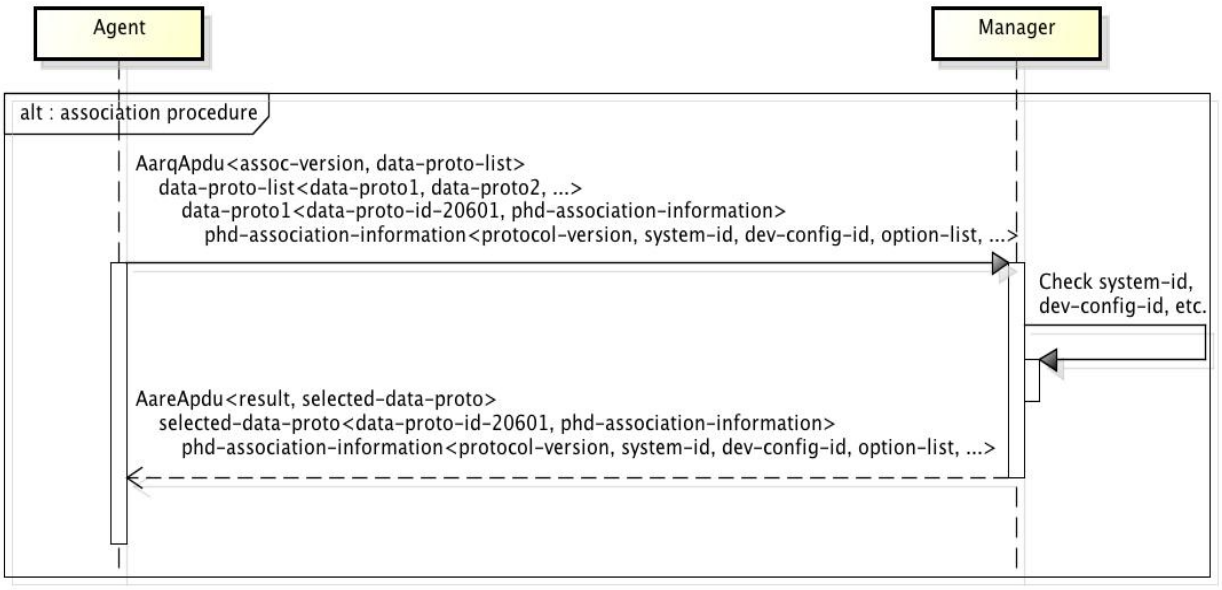

Figure 3.4. Association procedure when no errors occur

There are two situations that may occur when the Agent initiates the association with the Manager:

- the Manager knows the Agent's configuration;
- the Manager does not know the Agent's configuration.

If the Manager already knows about the Agent's configuration, the *result* field of the *AareApdu* message is set to *accepted*. This situation happens if a prior connection with the Manager had been established, or when the Agent has a standard configuration (i.e. a predefined configuration that is specified in a specialization standard). In this case, both entities transit into the *Operating* state.

If the Manager does not know the Agent's configuration, the Manager informs the Agent that the association request is accepted, but that the configuration is unknown. The *result* field of the *AareApdu* message is set to *accepted-unknown-config*. In this case, both entities enter in the *Configuring* state.

ASN.1 and MDER

Abstract Syntax Notation One (ASN.1) [LAR99] was used for representing data types and exchange formats defined in OEP. ASN.1 is a standard notation used for describing rules and structures for information data.

Many communication protocols specifications define messages as binary values of sequences of octets. ASN.1 provides means of defining complex data types and messages without necessarily determining their binary representation. This notation is supplemented by the specification of

one or more algorithms called *encoding rules*, which determine the value of the octets that carry the application semantics (called the transfer syntax).

ISO/IEEE 11073-20101:2004 [BS04] defines the Medical Device Encoding Rules (MDER) used in this standard.

3.2 Security analysis of OEP

As OEP's popularity increases, its security raises more and more interest. The authors of the OEP specification recognize the importance of security for the protocol. They included the following important notice in the final version of the standard: "*This standard is not intended to assure safety, security, health, or environmental protection in all circumstances*".

Our opinion is that OEP should not be used in practice without enhancements designed to provide appropriate security mechanisms.

In order to highlight the importance of the security in the context of this standard, let's consider two common communication scenarios covered by the standard:

- there is only one Agent and one Manager; both devices belong to the same person and they are located at the person's home;
- there are several Agents and one Manager; the devices are used in a dedicated medical environment, such as a hospital or a laboratory; the Manager acquires data from all the Agents, which are shared between hospital patients.

Analyzing the proposed common scenarios, some important security problems that must be addressed can be highlighted:

1. The OEP protocol is based on standardized wireless networks.

Security problem: various researchers have issued that wireless networks are vulnerable to many types of attacks

2. There is no device authentication mechanism defined in OEP.

Security problem: During the OEP association process, Managers may accept or reject communication with Agents. However, the purpose of the association procedure is to establish common communication parameters and not to perform security validations

3. There is no patient authorization mechanism defined in OEP.

Security problem: The existence of a patient authorization procedure is especially important when Agents are shared among different patients. For example, patient authorization is mandatory when Agents may only be used under the strict supervision of a specialist

4. There is no mechanism to correlate acquired data to the corresponding patients.

Security problem: In the current state of OEP, it is impossible to detect the source behind the medical data collected by Agents

In order to analyze the first security problem, let's suppose that Bluetooth is the choice for the communication channel between the Agent and the Manager. This prerequisite is easy to assume, since Bluetooth is the most common choice for the existing OEP implementations.

The McAfee Mobile Security Report 2009 [MCA09] shows that Wi-Fi/Bluetooth connections are ranked fourth among highest security concerns in mobile usage area, according to a survey conducted by Informa Telecoms & Media. The security concerns related to Bluetooth communications outrank the ones related to other areas, such as downloading multimedia content, internet browsing, or email messaging.

Bluetooth hacking gained momentum in 2003, when BlueSnarfing was released. Since then, many tools and methods have emerged to exploit different Bluetooth vulnerabilities. Dunning [DUN10] provides a Bluetooth threat taxonomy as a framework for classifying all Bluetooth-based threats. Table 3.2 lists the classifications and some examples of attacks and exploit methods.

Table 3.2. Bluetooth attacks classification

Attack classification	Threats	Threat level
Surveillance	Blueprinting, bt_audit, redfang, War-nibbling, Bluefish, sdptool, Bluescanner, BTScanner	Low
Range extension	BlueSniping, bluetooone, Vera-NG	Low
Obfuscation	Bdaddr, hciconfig, Spooftooph	Low
Fuzzer	BluePass, Bluetooth Stack Smasher, BlueSmack, Tanya, BlueStab	Medium
Sniffing	FTS4BT, Merlin, BlueSniff, HCIDump, Wireshark, kismet	Medium
Denial of service	Battery exhaustion, signal jamming, BlueSYN, Blueper, BlueJacking, vCardBlaster	Medium
Malware	BlueBag, Caribe, CommWarrior	Medium
Unauthorized direct data access	Bloover, BlueBug, BlueSnarf, BlueSnarf++, BTCrack, Car Whisperer, HeloMoto, btpincrack	High
Man in the middle	BT-SSP-Printer-MITM, BlueSpooof, bthidproxy	High

As shown in the Dunning taxonomy, the man-in-the-middle (MITM) [DUN10] attack has a high impact on the security and privacy of the medical data communicated through Bluetooth technology. Even the newest Bluetooth security mode, i.e. Secure Simple Pairing (SSP), which is used to create service-level security, is still vulnerable to MITM attacks. Haatja et al. discusses two new MITM attacks on SSP [HAA08]:

- falsification of information sent during the IO capabilities exchange;
- misleading the user to select a less secure option instead of using the Out Of Band (OOB) channel.

If an eavesdropper captures the messages exchanged in communication, it can impersonate any of the Agent and Manager. When impersonating the Agent, the eavesdropper can start an attack that leads to the case where the Manager collects improper medical data. A common behavior is that the Manager passes the acquired data to an external information system. If a doctor makes decisions based on the data stored within this information system, than the life of the person who is mistakenly linked to this data could be in jeopardy.

When the eavesdropper impersonates the Manager, it can acquire unauthorized personal medical information from Agents. Moreover, if the rightful Manager does not receive the data, it is unable to update the patient's medical history. This situation could lead to erroneous medical decisions, as well.

The impossibility to detect the source behind the medical data collected from the Agent is a very important security issue. The situation where a single Agent acquires medical data from several patients may lead to incorrect patient histories that affect the diagnosis of a person, or the medication prescription. Authentication is also a crucial aspect of identity management, especially when Agents are shared among patients.

Conclusion:

In this thesis we considere that the ISO/IEEE 11073-20601 should be extended to include two mechanisms that tackle the mentioned security issues:

- authentication;
- encryption.

This chapter presents a feasible authentication procedure and provides implementation details, a proof of concept and a set of conclusions. The encryption is not part of this research.

The authentication should be implemented as an extension of the OEP. Solutions that address the mentioned security issues can be implemented outside the scope of this protocol. However, this would lead to many inconsistencies and interoperability problems. The security mechanisms should be integrated into the standard, offering common solutions for these problems, instead of encouraging the development of proprietary implementations.

The authentication proposed:

- does not affect the performance of communication;
- does not modify the current version of OEP, thus enabling backward compatibility with the existing implementations that are currently in use.

The authentication mechanism is designed as optional. When situations do not require, authentication may be deactivated, to keep the complexity of the system to a minimum. For instance, authentication may be unnecessary when the Agent and Manager belong to the same person, and it is unlikely that someone else uses the devices. However, it should always be implemented when Agents are shared among patients in a healthcare environment, because of the high probability of erroneous data collection and data storage.

3.3 Enhancing OEP with Authentication Procedure based on Biometric Keys

OEP specifies a mandatory association procedure. The purpose of the association is to establish a common data protocol and a common set of operating parameters between Agents and Managers. The association procedure allows the Manager to reject connections with certain Agents. However, the association phase does not contain authentication.

A successful association procedure consists of the following two types of messages, as exemplified in Figure 3.4:

- AarqApdu – the Association Request;
- AareApdu – the Manager's response to the Association Request.

When the Agent initiates the association procedure, it transits into the *Associating* state and sends an *Association Request* to the Manager. The *AarqApdu* definition describes the *Association Request* message. The *AarqApdu* message contains information about the version of the association procedure and a list of supported communication protocols. If one of the supported protocols is OEP, then the message carries information that is used for announcing and negotiating the protocol version, profile, configuration, etc.

After the Agent sends the *Association Request* message, it waits either for an *Association Response* message from the Manager or for a timeout. The *AareApdu* definition describes the format of the *Association Response* message. The *AareApdu* message contains information about the outcome of the association procedure and the chosen communication protocol. The outcome can be set to accepted (either *accepted* or *accepted-unknown-config*) or rejected (*rejected-permanent*, *rejected-no-common-protocol*, *rejected-unauthorized*, etc.).

The association procedure is depicted in Figure 3.4.

3.3.1 Association extension for authentication

Section 3.2 highlighted the need for authentication between Agents and Managers. Section 2.3 described the proposed solution for the authentication, i.e. the *mutual challenge-response authentication* based on biometric keys. This section presents the enhancement of the OEP association procedure with the proposed authentication mechanism [EGN12b].

Figure 3.5 depicts the messages involved in the enhanced association procedure.

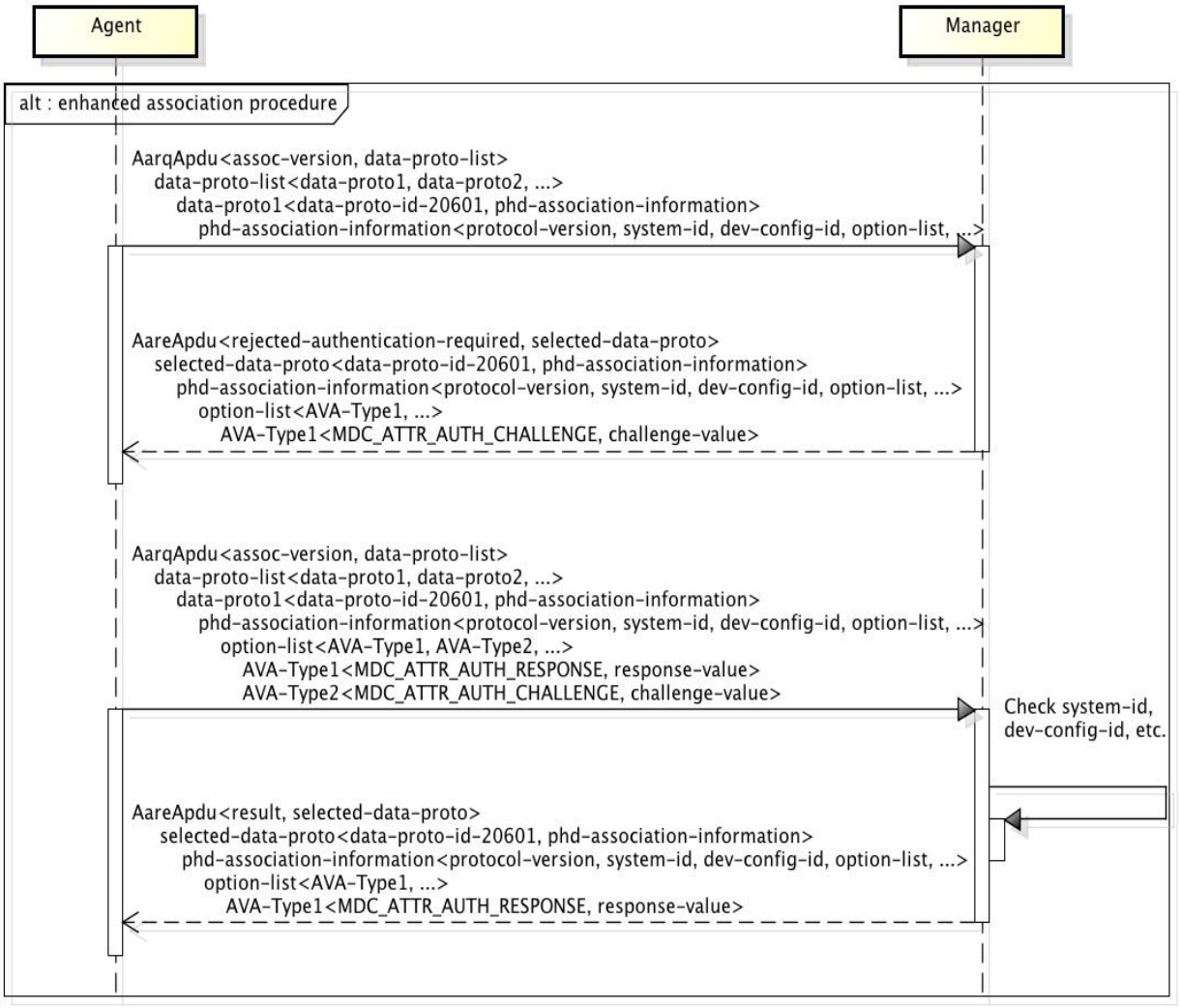

Figure 3.5. The association procedure with support for authentication

Protocol message flow

1. The first message represents an *Association Request*.

The Agent sends an *AarqApdu* message to the Manager. This message is identical to the one defined in OEP.

2. The second message represents the Manager's response to the *Association Request*.

This is an *AareApdu* message that is similar to the one used in the current version of OEP (second message in Figure 3.4). As opposed to the latter, the outcome of the *Association Request* in this case is set to a new value, named *rejected-authentication-required*. The *AareApdu* represents the authentication request. The message also contains the challenge, which is stored as an attribute in an optional list.

3. The third message represents the Agent's authentication attempt.

This message is another *Association Request* of type *AarqApdu*. However, in addition to the first *Association Request*, this message contains the response to the Manager's challenge, i.e. the authentication information. The message also contains a challenge for the Manager. The authentication information, i.e. the response and the new challenge, are stored as attributes in the available optional list.

4. The fourth message represents the *Association Response*.

This is an *AareApdu* message that contains the following:

- the outcome of the association procedure;
- the authentication verdict;
- the response to the Agent's challenge.

If the authentication fails, the outcome of the association procedure is set to *rejected-authentication-required* and a new challenge is sent to the Agent. In this case there is no response to the Agent's challenge.

If the authentication is successful, the outcome is usually set to either *accepted* or *accepted-unknown-config*, depending on the information received from the Agent.

This proposed extension does not change the specification of the current protocol. Authentication information can be exchanged using existing types of message, by adding new definitions to the ISO/IEEE 11073-10101 – Nomenclature (such as *rejected-authentication-required*) and using already defined optional fields for storing the challenges and responses.

The proposed extension of the protocol is optional. If a Manager does not support the *mutual challenge-response authentication*, Agents can associate using the protocol depicted in Figure 3.4. This procedure allows much flexibility to the protocol and enables backward compatibility.

3.3.2 The extended ASN.1 specification of OEP

This section focuses on the extension of the ASN.1 definitions with the proposed authentication features. Before detailing the extension, the following terms have to be defined:

- a new type of outcome for the *Association Response;*
- a new *attribute-id* for identifying the challenge;

- a new *attribute-id* for identifying the response.

The first term that should be defined is the new type of association outcome: *rejected-authentication-required*. The outcome of the association procedure is stored in the field *result* of the *AareApdu* message. *Result* is of type *AssociateResult*, which defines nine possible results, such as *accepted*, *accepted-unknown-config*, or *rejected-unauthorized*. *Rejected-unauthorized* is used with different purposes, and it cannot be used in the *mutual challenge-response authentication* process. Thus, the addition of a new outcome is proposed: *rejected-authentication-required*. The extended ASN.1 definition of the *AssociateResult* is depicted in Figure 3.6.

```
674 AssociateResult ::= INTEGER {
675     accepted(0),
676     rejected-permanent(1),
677     rejected-transient(2),
678     accepted-unknown-config(3),
679     rejected-no-common-protocol(4),
680     rejected-no-common-parameter(5),
681     rejected-unknown(6),
682     rejected-unauthorized(7),
683     rejected-unsupported-assoc-version(8),
684     rejected-authentication-required(9) }  -- See 9.7.3.2 for a usage description.
685                         -- All unassigned " AssociateResult " values are reserved
686                         -- for future expansion and shall not be used.
```

Figure 3.6. Extended ASN.1 definition of *AssociateResult*

When the Agent initiates an *Association Request*, it sends an *AarqApdu* message that contains a list of supported protocols. This information is stored in field *DataProtoList*, which is a list of *DataProto*. *DataProto* objects contain two fields: *data-proto-id* and *data-proto-info*. The former is a protocol identifier, while the latter defines the protocol's structure. If *data-proto-id* is set to *data-proto-id-20601*, which means that the Agent is ISO/IEEE 11073-20601 compliant, then *data-proto-info* stores a structure of type *PhdAssociationInformation*. In short, if the Agent initiates the association and is OEP compliant, than the association information is stored in a structure of type *PhdAssociationInformation*.

Similarly, the Manager's response, i.e. the *AareApdu* message, contains a *DataProto* field that holds information about the protocol chosen by the Manager, which is one of the protocols supported by the Agent. If the protocol is OEP, then the *AareApdu* message is also a container of a *PhdAssociationInformation* structure.

Note: The usage of a protocol different from OEP is not the subject of this research.

Every association message exchanged by the two entities contains a *PhdAssociationInformation* structure, which provides means for storing custom information in an optional field. For this

reason, the *PhdAssociationInformation* structure should be used as a container for the challenge and response.

PhdAssociationInformation contains a field of type *AttributeList*, which will store the exchanged authentication information. *AttributeList* is a list of *AVA-Type*, which is a data type that specifies the attribute of an object by its attribute ID and its value. Two different *AVA-Types* are used, one for the challenge and one for the response.

In order to define the two attributes, the ISO/IEEE 11073-10101 – Nomenclature [NOM04] needs to be extended by adding the definition of the IDs that correspond to the challenge and response. The definitions should be added in the "Partition: ATTR/GROUP" section of the Nomenclature, as shown in Figure 3.7.

```
[…]
#define MDC_ATTR_CMPLX_STATIC_ATTR                2622 /* */
#define MDC_ATTR_CMPLX_RECURSION_DEPTH            2623 /* */
#define MDC_ATTR_RANGE_CURR                       2624 /* */
#define MDC_ATTR_RANGE_OP_TEXT_STRING             2625 /* */
#define MDC_ATTR_AUTH_CHALLENGE                   2626 /* */
#define MDC_ATTR_AUTH_RESPONSE                    2627 /* */
[…]
```

Figure 3.7. ASN.1 definition of challenge and response attribute-ids

Once *MDC_ATTR_AUTH_CHALLENGE* and *MDC_ATTR_AUTH_RESPONSE* IDs are defined, *AVA-Type* attributes can be created and used for storing authentication information.

3.3.3 Illustration of the mutual challenge-response authentication

This section highlights the contents of the messages exchanged in the authentication process.

The Agent initiates the association procedure

The Agent initiates the communication requesting to establish an association with the Manager. At this point, the Agent has no knowledge whether the Manager employs an authentication protocol or not. The Agent sends an *AarqApdu* message, identical to the one defined in OEP. The *Association Request* corresponds to the first message depicted in Figure 3.5.

The Manager requests authentication

After the Agent sends the *Association Request*, the Manager replies with an *AareApdu* message that informs the Agent that the association is rejected and authentication is needed. This message, which corresponds to the message 2 depicted in Figure 3.5, is detailed in Figure 3.8.

0xE3 0x00	APDU CHOICE Type (AareApdu)
0x00 0x36	CHOICE.length = 54
0x00 0x09	**result = rejected-authentication-required**
0x50 0x79	data-proto-id = 20601
0x00 0x30	data-proto-info length = 48
0x80 0x00 0x00 0x00	protocolVersion
0x80 0x00	encoding rules = MDER
0x80 0x00 0x00 0x00	nomenclatureVersion
0x00 0x00 0x00 0x00	functionalUnits
0x80 0x00 0x00 0x00	systemType = sys-type-manager
0x00 0x08	system-id length = 8
0x11 0x22 0x33 0x44	system-id value
0x55 0x66 0x77 0x88	
0x00 0x00	manager's response to config-id is always 0
0x00 0x00	manager's response to data-req-mode-capab is always 0
0x00 0x01	optionList.count = 1
0x00 0x0C	optionList.length = 12
0x0A 0x42	**attribute-id = MDC_ATTR_AUTH_CHALLENGE**
0x00 0x08	attribute-value.length = 8
0x00 0x11 0x22 0x33	**manager's challenge**
0x44 0x55 0x66 0x77	

Figure 3.8. Example of an *AareApdu* message containing the authentication challenge

The Manager's response to the *Association Request* contains two important pieces of information related to authentication:

- the notification that the Agent must authenticate; this association outcome is stored in field *result*, which is set to *rejected-authentication-required;*
- the challenge sent to the Agent, which is stored as the value of an attribute contained in *OptionList*; the id value of this attribute is set to *MDC_ATTR_AUTH_CHALLENGE.*

The Agent initiates the authentication procedure

The *AarqApdu* message sent by the Agent contains the response to the Manager's challenge and a new challenge needed for assuring the mutual authentication protocol. This message corresponds to the third message depicted in Figure 3.5 and is illustrated in Figure 3.9.

0xE2 0x00	APDU CHOICE Type (AarqApdu)
0x00 0x52	CHOICE.length = 82
0x80 0x00 0x00 0x00	assoc-version
0x00 0x01	data-proto-list.count = 1
0x00 0x4A	data-proto-list.length = 74
0x50 0x79	data-proto-id = 20601
0x00 0x46	data-proto-info length = 70
0x80 0x00 0x00 0x00	protocolVersion
0xA0 0x00	encoding rules = MDER or PER
0x80 0x00 0x00 0x00	nomenclatureVersion
0x00 0x00 0x00 0x00	functionalUnits – e.g., flag unsolicited event reporting capability
0x00 0x80 0x00 0x00	systemType = sys-type-agent
0x00 0x08	system-id length = 8
0x88 0x77 0x66 0x55	system-id value
0x44 0x33 0x22 0x11	
0x40 0x00	dev-config-id
0x00 0x81 0x01 0x01	data-req-mode-capab
0x00 0x02	optionList.count = 2
0x00 0x20	optionList.length = 32
0x0A 0x43	**attribute-id = MDC_ATTR_AUTH_RESPONSE**
0x00 0x10	attribute-value.length = 16
0xFF 0xEE 0xDD 0xCC	**agent's response**
0xBB 0xAA 0x99 0x88	
0x77 0x66 0x55 0x44	
0x33 0x22 0x11 0x00	
0x0A 0x42	**attribute-id = MDC_ATTR_AUTH_CHALLENGE**
0x00 0x08	attribute-value.length = 8
0x00 0x11 0x22 0x33	**agent's challenge**
0x44 0x55 0x66 0x78	

Figure 3.9. Example of an *AarqApdu* message containing the authentication response

This message is similar to the first one sent by the Agent when initiating the association procedure. In addition to the former, it contains the response to the Manager's challenge and a new challenge for the Manager. The message encapsulates:

- an attribute having the *attribute-id* set to *MDC_ATTR_AUTH_RESPONSE* and the *attribute-value* set to the value of the authentication response;
- an attribute having the *attribute-id* set to *MDC_ATTR_AUTH_CHALLENGE* and the *attribute-value* set to the value of the authentication challenge sent to the Manager.

At this point, the Manager already knows the list of the Agent's supported protocols, which was provided in the association request message. The Agent has two choices:

1. Send the list again, as a form of redundancy check;
2. Send a message without the list of supported protocols. Although this reduces the message length, it leads to inconsistencies in the way the Manager handles association requests.

For interoperability reasons, it is suggested that there should be no change in the way the Manager handles association requests. It is proposed that the Agent resends the list of supported protocols to the Manager when sending the authentication message.

The Manager responds to the authentication request

Since the authentication is mutual, the Manager needs to authenticate to the Agent, as well. The fourth message depicted in Figure 3.5 contains the association response, the authentication verdict and the response to the Agent's challenge. The message is constructed in a similar way to the other messages presented in this section.

3.4 Enhancing OEP with Bidirectional Association Initiation

In the ISO/IEEE version of the protocol, the Agent is the only entity that is able to initiate the association procedure. The bidirectional association initiation means that both the Agent and the Manager can initiate the association procedure, an important feature that the communication protocol lacks. This represents a limitation of the communication capabilities, which affects the exchange of medical data and the way medical devices are configured and managed [EGN13a].

For instance, if the Agent is configured as a dialysis device, the correct configuration of this device is vital, and it should be done whenever the care personnel, represented by the Manager, needs to. The Manager should be enabled to change the Agent's configuration parameters, without having to wait for an association request from it. The bidirectional association initiation is also needed when the Manager needs to reestablish an association. This may happen when the Manager recovers from a crash and needs to re-associate to the Agents, if the authentication procedure is changed and the Agents have to re-authenticate using another protocol, or if configurations such as the session length are changed.

The ISO/IEEE version of OEP does not support authentication. Previous sections discussed about the necessity of an authentication mechanism and provided a solution to address this need. The proposed authentication phase was designed as optional, in order to support interoperability with existing implementations. Thereby, two different scenarios can be distinguished, in terms of the bidirectional association initiation:

- extending the association procedure for implementations that allow the authentication procedure;
- extending the association procedure for implementations that do not allow authentication (existing implementations of the ISO/IEEE version of the protocol).

3.4.1 Bidirectional association when authentication is required

This section proposes the extension of the association procedure for systems that employ authentication mechanisms [EGN13a]. The method of authentication considered here is the one presented in Section 2.3.1, i.e. *mutual challenge-response*. Figure 3.10 describes the flow of messages exchanged between the Agent and the Manager when the Manager initiates the association procedure.

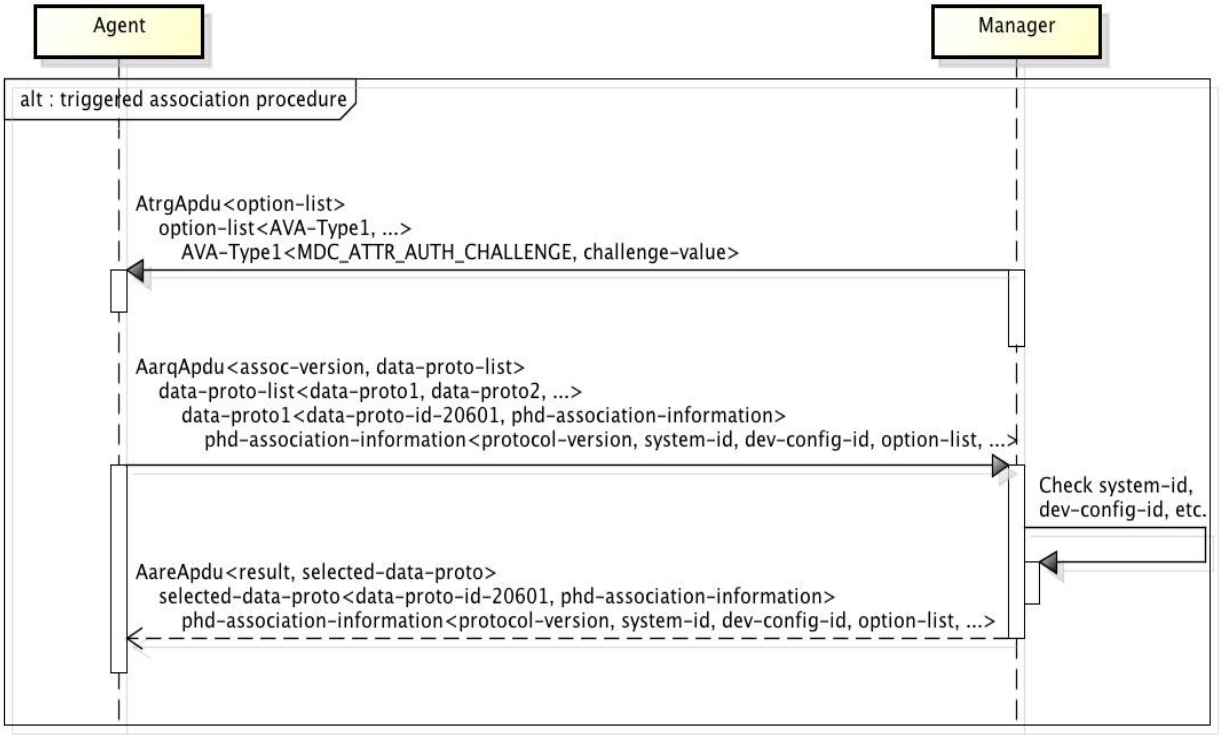

Figure 3.10. The message flow of the association procedure initiated by the Manager

The Manager initiates communication by triggering the Agent to request for association. The first message in Figure 3.10 represents the trigger message. Existing association message types were used for describing the exchange of authentication information. In this case, however, existing association message types cannot be used to trigger the association. These types can only be used when the Agent and the Manager are in the *Associating* state. Since the Manager is in the *Unassociated* state when initiating the association, new type of message should be defined, to avoid ambiguous use of these messages.

Therefore, a new type of message named *AtrgApdu* was defined, which is detailed in subsection 3.4.3. This message is used by the Manager to trigger the Agent to request for association. To optimize the traffic flow, the *AtrgApdu* sent by the Manager should also contain the challenge, stored as the value of an *AVA-Type* attribute contained in the *optionList* field.

The second message represents the Agent's response to the triggered event received from the Manager. The *AarqApdu* message contains three important pieces of information:

- the *Association Request* triggered by the *AtrgApdu* message;
- the response to the Manager's challenge;
- the challenge for the Manager.

The response to the Manager's challenge is the output of an irreversible function that takes two inputs: the challenge and the cryptographic key. A new *AVA-Type* attribute is created having the id set to *MDC_ATTR_AUTH_RESPONSE* and value set to the computed response. This attribute is stored in the *optionList* field contained in the *AarqApdu* message sent to the Manager.

The challenge sent by the Agent ensures that the authentication procedure is mutual. The Agent's challenge is generated identically to the one received from the Manager in the *AtrgApdu* message. This challenge is stored as the value of a new *AVA-Type* attribute that has the id set to *MDC_ATTR_AUTH_CHALLENGE*. This attribute is also added to *optionList*.

The third message represents the Manager's response to the *Association Request*. It also contains the authentication verdict and the response to the Agent's challenge. If the authentication succeeds, the outcome stored in the field *result* of the *AareApdu* message is usually set to either *accepted* or *accepted-unknown-config*, depending on the association information received from the Agent. If the authentication fails, the *result* is set to *rejected-authentication-required* and a new challenge is sent to the Agent.

The response to the Agent's challenge is determined identically to the one calculated by the Agent. A new *AVA-Type* attribute is created having the id set to *MDC_ATTR_AUTH_RESPONSE* and value set to the determined response. This attribute is stored in the *optionList* field of the *AareApdu* message.

If the *response* of the *AareApdu* message is *accepted*, the Agent enters in the *Operating* state. In this state the association is considered established, the configuration of the device is known, authentication is ensured and medical data can be transmitted.

3.4.2 Bidirectional association when authentication is not required

This section proposes the extension of the association procedure for systems that don't employ any authentication mechanism [EGN13a]. The mechanism is similar to the one used when authentication is required. Figure 3.10 depicts the association message flow.

In this case, the trigger message sent from the Manager to the Agent does not encapsulate any authentication information, such as the challenge. The field *optionList* of the *AtrgApdu* message is empty.

The other two messages, i.e. *AarqApdu* and *AareApdu*, are identical to the ones used by the systems that implement the current specification of the protocol. The only extension of the protocol in this scenario is the *AtrgApdu* message, which represents just a trigger for the Agent to start the association phase.

3.4.3 The extended ASN.1 specification of OEP

This section presents the ASN.1 definitions of the terms introduced to support the bidirectional association initiation.

The first message depicted in Figure 3.10, i.e. *AtrgApdu*, is not defined in the ISO/IEEE version of the standard. A new ASN.1 message definition should be added to the specification, as shown in Figure 3.11.

```
604 ApduType ::= CHOICE {
605     aarq [57856] AarqApdu,  --Association Request [0xE200]
606     aare [58112] AareApdu,  --Association Response [0xE300]
607     rlrq [58368] RlrqApdu,  --Association Release Request [0xE400]
608     rlre [58624] RlreApdu,  --Association Release Response [0xE500]
609     abrt [58880] AbrtApdu, --Association Abort [0xE600]
610     prst [59136] PrstApdu, --Presentation PDU [0xE700]
611     atrg [59392] AtrgApdu -- Association Trigger [0xE800]
612 }
```

Figure 3.11. Extended ASN.1 definition of *ApduType* with support for *AtrgApdu*

The *AtrgApdu* message has a clearly defined purpose, i.e. to determine the Agent to start the association procedure. This message is not intended to be a container for medical data, or configuration information. The only piece of information stored in the *AtrgApdu* message is the authentication challenge, which is used when systems employ an authentication procedure.

Therefore, we propose that the *AtrgApdu* message type contains one optional field of type *AttributeList*, named *optionList*. *AttributeList* is a list of *AVA-Type* attributes, which are suitable

containers for storing authentication information such as the challenge. *AVA-Type* attributes are adaptable to any changes regarding the challenge, such as its length, or type.

If there is no authentication procedure employed, the *optionList* should be left empty. Figure 3.12 shows the ASN.1 definition of the *AtrgApdu* message.

```
792 AtrgApdu ::= SEQUENCE {
793     optionList AttributeList
794 }
```

Figure 3.12. ASN.1 definition of *AtrgApdu*

AVA-Type is a data type that specifies attribute objects characterized by *attribute-ids* and *attribute-values*. When instantiating new *AVA-Type* attributes, their *attribute-ids* must be set to values defined in the ISO/IEEE 11073-10101 – Nomenclature. Therefore, attributes used for storing authentication information should use two *attribute-ids*, i.e. *MDC_ATTR_AUTH_CHALLENGE* and *MDC_ATTR_AUTH_RESPONSE*. The definition of the two ids are added in the "Partition: ATTR/GROUP" section of the Nomenclature, as shown in Figure 3.7.

If no authentication is required, the *AtrgApdu* message contains an empty list of attributes. If authentication is required, the *AtrgApdu* contains a list that includes an *AVA-Type* attribute that has the *attribute-id* set to *MDC_ATTR_AUTH_CHALLENGE* and the *attribute-value* set to the value of the actual challenge.

The extension of the protocol for allowing the Manager to instantiate communication does not imply major changes in the standard. The behavior of the applications that implement the ISO/IEEE version of the protocol does not change and the new functionality can be easily implemented.

3.4.4 Illustration of association initiated by the Manager

The section highlights the contents of the messages exchanged in the association process. We consider the case where an authentication mechanism is used. In this scenario, the challenge's length is 64 bits, and the response is 128 bits.

The association trigger

Figure 3.13 describes the message sent by the Manager to initiate communication. This message corresponds to the first message depicted in Figure 3.10.

0xE8 0x00	APDU CHOICE Type (AtrgApdu)
0x00 0x10	CHOICE.length = 16
0x00 0x01	optionList.count = 1
0x00 0x0C	optionList.length = 12
0x0A 0x42	**attribute-id = MDC_ATTR_AUTH_CHALLENGE**
0x00 0x08	attribute-value.length = 8
0x00 0x11 0x22 0x33	**manager's challenge**
0x44 0x55 0x66 0x77	

Figure 3.13. Example of an *AtrgApdu* message

The association request

The *Association Request* contains the response to the Manager's challenge and its own challenge. This message, which corresponds to the second message presented in Figure 3.10, is detailed in Figure 3.14.

0xE2 0x00	APDU CHOICE Type (AarqApdu)
[...]	
0x00 0x02	optionList.count = 2
0x00 0x20	optionList.length = 32
0x0A 0x43	**attribute-id = MDC_ATTR_AUTH_RESPONSE**
0x00 0x10	attribute-value.length = 16
0xFF 0xEE 0xDD 0xCC	**agent's response**
0xBB 0xAA 0x99 0x88	
0x77 0x66 0x55 0x44	
0x33 0x22 0x11 0x00	
0x0A 0x42	**attribute-id = MDC_ATTR_AUTH_CHALLENGE**
0x00 0x08	attribute-value.length = 8
0x00 0x11 0x22 0x33	**agent's challenge**
0x44 0x55 0x66 0x78	

Figure 3.14. Example of an *AarqApdu* message triggered by the *AtrgApdu*

The association response

Since the authentication is mutual, the Manager must authenticate to the Agent, as well. The third message described in Figure 3.10 represents the last step of the association process. This message contains the association outcome, the verdict to the Agent's authentication attempt, and the response to the Agent's challenge, if authentication succeeded. This message is constructed in a similar way to the messages presented in this section.

3.5 Case Study

The OEP enhancements were implemented [EGN12c] as extensions to the OpenHealth Project from LibreSoft [LOH], which is an open source implementation of the ISO/IEEE 11073-20601 protocol. OpenHealth Project is developed in Java.

In order to draw conclusions about the implementation of the extensions, an analyzer was needed for capturing the OEP messages exchanged. However, the research has revealed no available analyzer. Manufacturers of OEP compliant devices, such as Bluegiga [BLU], have developed internal tools for the message analysis that are not made public. Thus, a Wireshark plugin was developed for dissecting OEP messages.

Wireshark plugin for OEP

Wireshark [WIR] is a popular network protocol analyzer that captures and analyzes data packets over a communication channel. Data coming over different types of networks (e.g. Loopback, Ethernet) can be captured and analyzed in real-time, or saved locally for later examination. The tool provides an intuitive user interface where the structure of the packets and the values encapsulated within the fields can be studied.

Wireshark provides dissectors covering a varied range of popular protocols, such as TCP, or PPPoE. In addition to this, it is possible to create plug-ins for the analysis of new protocols. In this case study, the Wireshark plugin was developed for the Windows operating system and used for intercepting the OEP messages exchanged in a TCP-based communication between the OpenHealth implementations of the Agent and Manager.

In order to create a plugin dissector for the OEP data packets, the following steps were followed:

Step 1. Create a new directory for the dissector, within the plugins directory: *<Wireshark_installation_dir>\plugins\oep* (on Windows platforms)

Step 2. Add *packet-oep.c* to the plugins directory

This is the main file, which contains the source code for the dissector. Among other things, this file defines the port on which the OEP packets are expected to arrive, functions for registering the dissector and for dissecting the data packets. Every plug-in directory has to contain a file named *packet-<protocol_name>.x* where the actual implementation is found

Step 3. A set of additional files are added to the OEP directory

The content of these files is similar to the one found in most of the plug-ins. There are situations where the only difference is represented by the values for the parameters, which indicate the name of the protocol. These files are:

- Makefile.nmake, the Windows makefile for the plug-in;
- Makefile.common, which contains the definition of the parameters that can be ported between different operating systems;
- moduleinfo.h, which specifies the plug-in version info; this can be helpful in case of redistributing the plugin;
- moduleinfo.nmake, which contains the DLL version info for Windows;
- plugin.rc.in, which contains the DLL resource template for Windows.

Step 4. The source file *packet-oep.c* is compiled. If the build is successful, an *oep.dll* file is generated in the OEP directory

Step 5. *oep.dll* is copied in the Wireshark's plugins directory

Figure 3.15 depicts the implementation for displaying the *Association Response*'s outcome in the Wireshark user interface. The array *aare_result_values* maps the possible values of *result*, as defined by the OEP protocol, to the text information that is displayed by Wireshark. For example, when an *Association Response* message with *result* set to 0 is received, Wireshark displays that the outcome of the received *Association Response* is *accepted*.

The plugin was designed for the proposed extended version of OEP. The definition also contains fields required in the authentication process, such as the *rejected_authentication_required* outcome. The value *rejected-authentication-required* is displayed when an *Association Response* message having *result* set to 9 is received. This is performed in the function *dissect_oep*, where the actual dissemination is implemented and where the packet details are set for display.

```
static const value_string aare_result_values[] = {
        { 0, "accepted" },
        { 1, "rejected-permanent" },
        { 2, "rejected-transient" },
        { 3, "accepted-unknown-config" },
        { 4, "rejected-no-common-protocol" },
        { 5, "rejected-no-common-parameter" },
        { 6, "rejected-unknown" },
        { 7, "rejected-unauthorized" },
        { 8, "rejected-unsupported-assoc-version" },
        { 9, "rejected-authentication-required"},
        { 0, NULL }
};

static hf_register_info hf_aare[] = {
        { &hf_aare_result,
            { "Result", "oep.aare.result",
            FT_UINT16, BASE_DEC,
            VALS(aare_result_values), 0x0,
            NULL, HFILL }
        },
        ...
}

static void dissect_oep(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)
{
    ...
    switch (packet_type) {
        case OEP_TYPE_AARQ:
            ...
            break;
        case OEP_TYPE_AARE:
            offset = 4; /* increase with length of packet_type field */

            /* add to the tree the value of the association response result */
            proto_tree_add_item(pdu_tree, hf_aare_result, tvb, offset, 2, ENC_BIG_ENDIAN);

            offset +=2; /* increase with length of association response result */
            ...
            break;

        case OEP_TYPE_RLRQ:
            proto_tree_add_item(pdu_tree, hf_rlrq_reason, tvb, 4, 2, ENC_BIG_ENDIAN);
            break;

        case OEP_TYPE_RLRE:
        ...
    }
}
```

Figure 3.15. Wireshark plugin – example of mapping *Association Response* outcomes to displayed texts

Figure 3.16 shows the displayed information when the outcome of the *Association Response* is set to *rejected_authentication_required*.

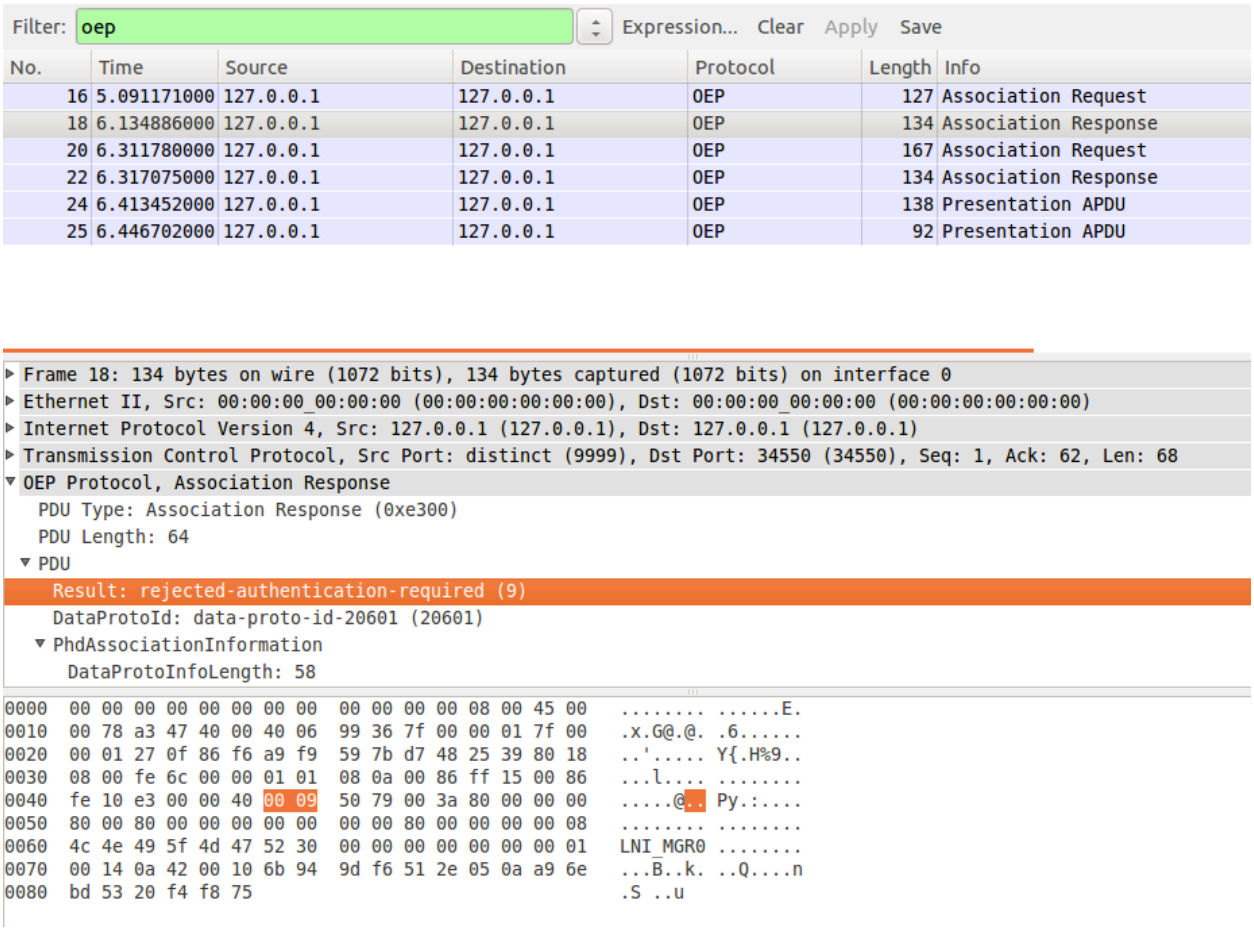

Figure 3.16. Example of Association Response in Wireshark

The results of the case study were discussed in the LLP Erasmus Intensive Programme *Secure Web Applications: Best Practices for Protection and Development.* During the programme, students were able to analyze the protocol in two states: the current state, as defined by the ISO/IEEE 11073-20601 standard and the enhanced state, which contains the implemented authentication protocol. They were able to perform well-known attacks, such as the man in the middle, and observe the need for security and patient identification. Several conclusions drawn in the programme are mentioned below:

- OEP cannot be used in shared environments, such as laboratories or hospitals, as it does not provide any patient identification mechanism;
- OEP should not be used without security features implemented on top of it; OEP is open to many attacks;
- the Manager should be able to initiate the *Association* procedure;
- the authentication protocol should be mutual;

- an authentication protocol included in the standard, or proposed as a guideline in the standard specification, would enable interoperability between the mobile medical devices.

4 FUNDAMENTALS OF CONFORMANCE TESTING FOR HL7-BASED APPLICATIONS

4.1 Health Level 7

Health Level 7 (HL7) is an American National Standards Institute (ANSI) accredited Standards Developing Organization (SDO) that operates in the healthcare domain [HL7]. HL7 is unique among SDOs in that it is focused solely on the development of healthcare informatics interoperability standards. HL7's declared mission is to improve care delivery, optimize workflow, reduce ambiguity and enhance knowledge transfer between the eHealth stakeholders.

According to HL7, the organization consists of more than 2300 members from which approximately 500 are corporate members that represent more than 90% of the worldwide healthcare information systems vendors.

4.1.1 HL7 standards

HL7 provides a wide range of eHealth standards, which are classified in the following categories:

- messaging and data interchange standards (e.g. HL7 V2.x and V3);
- document standards (e.g. CDA);
- conceptual standards (e.g. HL7 RIM);
- application standards (e.g. CCOW).

HL7 version 2

HL7 version 2 is a standardized protocol that facilitates the exchange of clinical information in a message-based communication. The second version of the HL7 messaging standard was released in 1988, just a year after the release of first version. Since then, HL7 specified several

incremental releases of V2, grouped under the name of HL7 V2.x. The final draft of the latest release of HL7 V2, namely V2.8, was announced in September 2011.

HL7 V2.x defines the communication protocol and the structure of the messages. It does not specify the communication channel and its properties. Even though it sometimes offers guidance with respect to the transport protocols, these are not mandatory requirements.

HL7 V2.x is organized into chapters. With most of the V2 releases, the first chapter introduces the standard, Chapter 2 defines the encoding and decoding of the V2 messages and the rest of the chapters provide information about a custom area or domain of healthcare information addressed by a specific work group.

The most common notation used for encoding HL7 V2.x messages uses ASCII text strings delimited by separators. Messages encoded in this way contain the following elements:

- segments – logical grouping of data fields;
- fields – strings that correspond to HL7 data types;
- components – elements composing a complex field;
- delimiter characters – separators that delimitate segments, fields, components, subcomponents, etc.

In addition to this encoding, HL7 also adopted a specification for encoding HL7 V2.x messages in XML, i.e. HL7 V2X.xml.

Currently, every major medical information system vendor in the United States supports the HL7 V2.x messaging standard. It is supported by over 90% of the hospital information systems in US, widely supported by the healthcare service suppliers worldwide and, according to Benson [BEN09], it is the most used healthcare interoperability standard in the world.

HL7 version 3

Even though version 2 proved to be very successful, HL7 realized that a different methodology was needed to support interoperability between healthcare systems. As a result, version 3 of the standard was released.

The declared objectives of HL7 V3, as summarized by Hinchley [HIN05], were:

- reflect trends in system interoperability;
- remove optionality as far as possible;
- include international requirements from earliest stages;
- plug-and-play capability.

In contrast to the second version, V3 introduces a different approach for defining the communication model and message semantics. Object modeling is one of the most important changes V3 proposed. Moreover, HL7 Reference Information Model (HL7 RIM), which is the fundamental model from which all HL7 messages are derived, was introduced.

HL7 V3 was designed forward compatible, but not backward compatible. This is probably the main cause why V2.x is still more used worldwide. Although V3 covers all the information content defined in V2.x, and many mapping tools between the two versions were designed and implemented, full compatibility was not yet achieved.

HL7 Clinical Document Architecture

HL7 Clinical Document Architecture (CDA) is an HL7 standard that defines the structure and semantics of electronic clinical documents such as consultation notes, discharge letters, or laboratory reports.

CDA is closely connected to HL7 V3. Like HL7 V3, CDA was developed using HL7 Development Framework (HDF). Moreover, it uses the same information model, i.e. HL7 RIM, and the same data types defined in version 3.

HL7 V3 messages are usually used when the information exchanged between healthcare systems is related to specific events that occur. Messages may not require human interaction. As opposed, CDA documents are usually the result of a note written by a clinician. CDA documents allow free form narrative speech, are persistent in nature, can be signed and carry information as a whole, i.e. are not divisible.

The content of a CDA document consists of a mandatory section that contains text written in natural language, which enables human interpretation of the content, and optional structured sections, which rely on coding systems such as Systematized Nomenclature of Medicine (SNOMED) or Logical Observation Identifiers Names and Codes (LOINC).

HL7 Clinical Context Object Workgroup

Clinical Context Object Workgroup (CCOW) is a standard protocol designed to allow healthcare systems synchronize clinical information in real time in a unified way. CCOW is the primary healthcare framework to support Context Management [SEL01]. CCOW enables stakeholders to create contexts that allow it to share clinical information and have access to different healthcare services simultaneously in the same interface. As part of the Context Management, the Single Sign On process enables users to acquire access to several healthcare services from different vendors through the use of a single username and password authentication.

Arden Syntax

Arden syntax is a language for encoding procedural medical knowledge. Originally not an HL7 standard, Arden syntax was adopted by HL7 beginning with version 2.0. It is used for creating Medical Logic Modules (MLM) that generate alerts, diagnoses, clinical interpretations and also administrative support.

4.1.2 HL7 Reference Information Model

HL7 V2.x does not define an information model. This means that HL7 V2.x-based systems have to agree upon a common set of rules that describe message contents. In other words, in order to achieve functional interoperability, systems have to generate a localized standard that allows them to exchange clinical information. This characteristic of the standard leads to serious interoperability issues when interconnecting different healthcare systems.

The goal of HL7 Reference Information Model (RIM) is to provide the same information model for any healthcare application. As opposed to the method in which HL7 V2.x-based systems establish the common localized information model, which is bound to a specific domain, RIM was designed on a top-down approach, meaning that it defines the healthcare domain as a whole in order to improve communication between any eHealth application, regardless of its specific domain.

The foundation classes, which represent the core of HL7 RIM, can be used to create any object from the eHealth world. There are six foundation classes, namely Act, Participation, Entity, Role, ActRelationship and RoleLink. In addition to the foundation classes, RIM introduces two infrastructure classes to represent messages and documents, i.e. MessageControl and CDA, respectively.

The foundation classes and their associates are referred to as the RIM Backbone. The Unified Modeling Language (UML) diagram representing the RIM Backbone is shown in Figure 4.1.

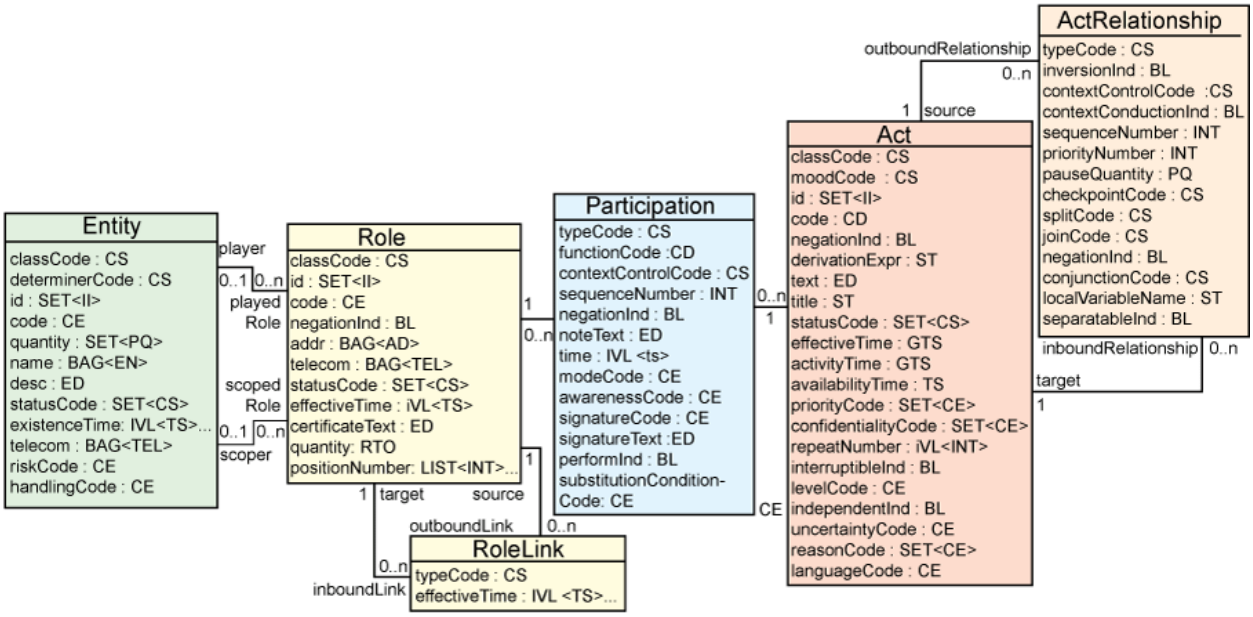

Figure 4.1. HL7 RIM Backbone (source: IBM [SHA10])

HL7 V3 messages are the result of a combination of three processes called model refinement, model constraining and localization. Through model refinement the model is restrained to a specific domain. This is achieved by selecting only the classes and attributes specific to that domain. Constraining implies restricting attributes to a specific vocabulary. Localization refers to the adaptation of the model to the particularities of the eHealth systems in a specific geographical area.

A Domain Message Information Model (D-MIM) is a model derived from the HL7 RIM that represents all the concepts needed to support the communication requirements of a particular eHealth domain. D-MIM is comprised from clones of RIM classes and is not intended to be serialized. D-MIM may be replaced in the future with Domain Information Model (DIM), which is defined in the HL7 Development Framework [HDF].

A Refined Message Information Model (R-MIM) is obtained by constraining a D-MIM. As opposed to the D-MIM, it has a single entry point and can be serialized. The R-MIM type of model is defined in the Message Development Framework (MDF), the predecessor of HDF. R-MIM may be replaced in the future with Serializable Information Model (SIM), which is defined in the HDF.

Implementable Technology Specification (ITS) describes a method of encoding HL7 artefacts. ITSs specify how abstract models can be serialized and transmitted.

Figure 4.2 (adaptation of the figure by Spronk [SPR], slide 9) depicts the process of deriving models from the RIM.

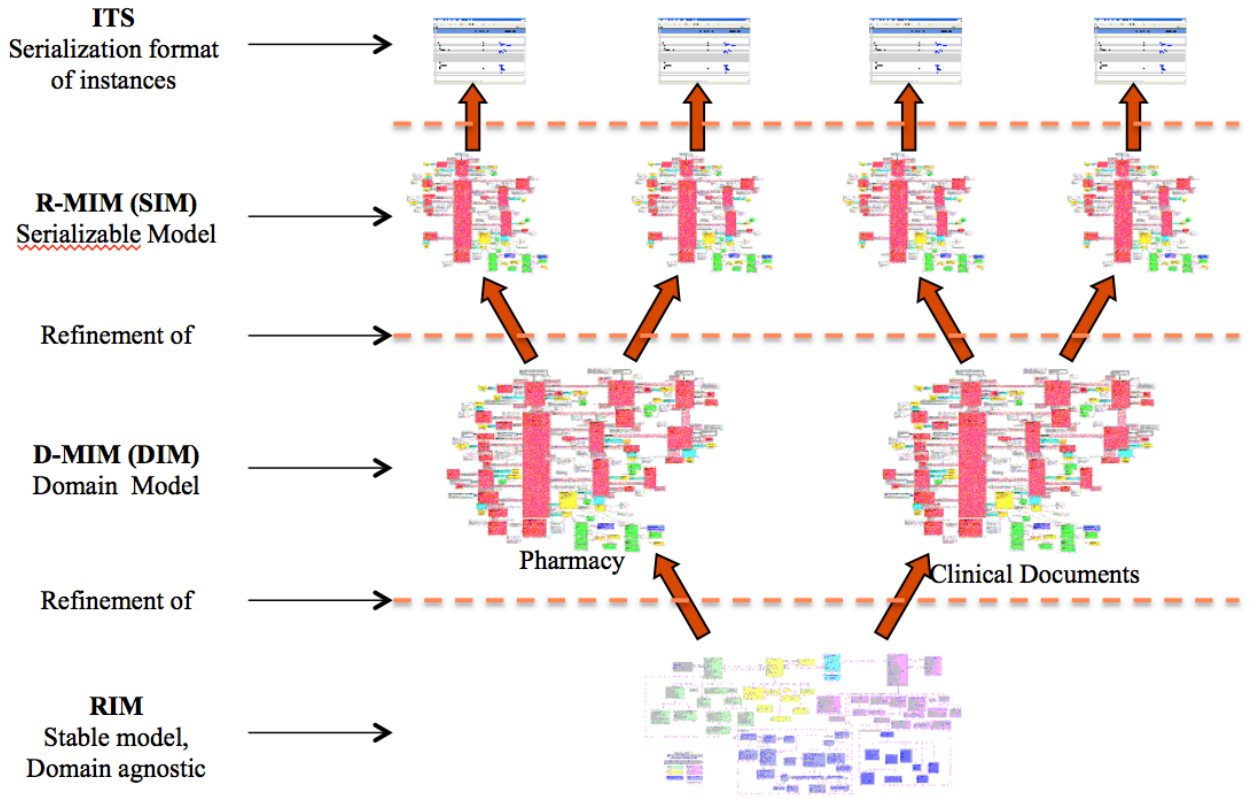

Figure 4.2. Refinement of the HL7 RIM (adaptation of figure by Spronk [SPR], slide 9)

4.1.3 HL7 version 3

The development of HL7 V3 started in 1996. The standard was developed in parallel with V2.x versions. HL7 V2.x proved successful. However, as the number of eHealth systems adopting the V2.x standard worldwide was constantly growing, significant interoperability problems emerged. A new methodology was required to address the problems of V2.x such as, for instance:

- the lack of a documented methodology for developing V2.x messages;
- the lack of an information model (applications supporting V2.x establish ad-hoc protocols);
- the lack of a formal definition of content and relationships (fields and events are described in natural language in V2.x);
- the degree of optionality and ambiguity;
- vocabularies that define coding terms are negociated ad-hoc by the V2.x-based systems.

In essence, HL7 V3 differs from V2.x in that it is based on a set of principles that provide a common philosophy for V3 standard developers. This is backed by the release of the HL7 Reference Information Model (RIM) and HL7 Development Framework (HDF).

HL7 V3 development methodology

HL7 uses a formal object oriented design methodology for the development of V3 standards. As opposed to V2.x, which defines just a messaging artefact, the development methodology allows V3 to introduce a family of interoperability artefacts, such as messages, documents, rules and templates.

The initial development methodology was called Message Development Framework (MDF). As the Unified Modeling Language (UML) developed, a new methodology was conceived: HL7 Development Framework (HDF). Most of the concepts defined in HDF are modeled with UML technologies.

Documentation for the V3 messaging can be created using HDF. The documentation can be classified in two different categories:

- documentation for the dynamic flow of messages (dynamic models);
- documentation for the static structures of the messages (static models).

The dynamic models include storyboards, interaction diagrams and activity diagrams, while static models include the HL7 Reference Information Model (RIM), the Domain Information Model (DIM), which is the successor of D-MIM, and the Serializable Information Model (SIM), the successor of R-MIM.

The HL7 metamodel

The Model Interchange Format (MIF) is used for creating the formal definition of the HL7 metamodel. MIF consists of a set of XML-based formats that support defining and exchanging HL7 artefacts. The HL7 metamodel represents all existing HL7 artefacts in a consistent way. Even though MIF is very complex, and designed to be machine-interpreted exclusively, there is currently no better alternative for representing the HL7 metamodel.

The MIF-based HL7 metamodel can be derived or translated into other representation formats, such as Unified Modeling Language XML Metadata Interchange (UML/XMI), Web Ontology Language (OWL), XML Schema (XSD), or Docbook. However, the levels of detail supported by the mentioned alternatives do not match MIF's level, which means that translating the MIF metamodel to another representation format may result in losing V3-specific details.

Implementation Technology Specification (ITS)

All HL7 V3 Messages are derived from the RIM, as shown in Figure 4.2. The Serializable Information Model (SIM) defines messages in an abstract manner. In order to be transmitted "on-the-wire", they are serialized in a format known by the communication parties. This format is generically known as Implementation Technology Specification (ITS). ITSs specify how instances of SIM should be transmitted between eHealth systems. Currently, the most commonly used ITS is XML ITS.

HL7 V3 integration profiles

Integration profiles provide solutions for a specific integration problem, in a particular healthcare domain. Profiles provide details for implementers to develop systems that cooperate to address the integration problem. Integrating the Healthcare Enterprise (IHE) is an initiative that aims to improve the way eHealth systems exchange and share clinical information through the use of the integration profiles [IHE].

IHE defines integration profiles for specific domains such as Laboratory (LAB), Patient Care Coordination (PCC), Patient Care Device (PCD), etc. IHE PCC domain handles general clinical care aspects such as document exchange and order processing. IHE Patient Care Coordination Technical Framework (PCC TF) provides guidelines for specific implementations of healthcare standards to promote appropriate exchange of medical information between eHealth systems.

Integration profiles specify interactions in terms of a set of standards-based transactions between entities named actors. An example of an IHE integration profile is Query for Existing Data (QED) [PCC08], which is defined by PCC TF in the season 2007-2008. Query for Existing Data is the integration profile used by the Healthcare Information System (HIS) that was used as the case study for the testing framework proposed in Chapter 5.

4.2 Conformance testing in eHealth domain

National E-Health Transition Authority (NEHTA) defines *conformance* as the evaluation, or assessment, of the adherence of an implementation to a specification or a standard [NEH]. *Conformance assessment* ensures that applications accurately implement specifications and standards. Conformance assessment can be performed by a first-party (self-assessment), a second- or a third-party. Third-party conformance assessment provides increased rigor.

Conformance requirements are technical requirements in a specification that have to be supported by the implementation. One of the main requirements of a conformance evaluation is having a specification that contains assessment criteria, such as conformance requirements, against which applications can be objectively evaluated. Another requirement is to have a set of assessment methods and procedures that assessors can use during conformance evaluations in a repeatable and tester-independent fashion.

Conformance assessment can be performed through observation or testing. *Conformance testing* provides software developers and users assurance and confidence that the conforming application behaves as expected, performs functions in a know manner, or possesses a prescribed interface or format. *Validation* is the process of testing applications for conformance. The validation process consists of all the steps necessary to perform the conformance testing. In this thesis, the term *validation* is used interchangeably with *conformance testing*.

As a result of conformance testing, the following can be achieved:

- increased probability that applications are implemented correctly;
- increased probability of portability and interoperability;
- buyer's increased confidence in a vendor's product.

4.3 Conformance test approaches: technologies and tools

Section 4.1.3 introduces the most important characteristics that differentiate HL7 V2.x of V3. Since the two versions are not fully compatible, existing tools are specialized for testing either V2.x or V3 applications. Given the success of V2.x, it is no surprise the fact that most of the existing technologies and tools are used for testing V2.x-based applications. This section presents the most important testing methodologies, tools, or utilities available.

4.3.1 Tools for testing HL7 V2-based applications

1. HAPI

HL7 Application Programming Interface (HAPI) is an open source project designed to help HL7 V2 developers and testers [HAP]. HAPI is currently probably the most used HL7 V2 library available. HAPI provides the following functionalities:

- HL7 V2 message parser (used in message validation);
- HL7 V2 message creator;
- transport operations (i.e. send/receive HL7 V2 messages, handle encodings, etc.);
- etc.

HAPI TestPanel [HTP] is a rich testing application based on HAPI. It consists of a fully feature HL7 V2 message editor, a sender and receiver that handle communications with Systems Under Test (SUTs) and a validation module, which uses DefaultValidation, the HAPI built-in validation mechanism. TestPanel supports validating messages using HL7 Conformance Profiles, special XML-based files that constrain V2 message definitions. Conformance profiles are generally created using the Messaging Workbench, which is presented below.

2. The Messaging Workbench

The Messaging Workbench (MWB) is a multipurpose productivity tool for HL7 V2.x developers [BON]. It is a mature tool, which is, however, available only on Windows platforms. It incorporates an online message validation service and a message generator that can be used in testing.

3. DVTk

DICOM Validation Toolkit (DVTk) is another open source project used for testing communication protocols in medical environments [POT07]. Although it initially offered support

solely for Digital Imaging and Communications in Medicine (DICOM), it now supports also HL7 and IHE integration profiles. The project started in 1997, when the first version of the Validation Test Suite (VTS) was released. In the subsequent years, VTS evolved and changed its name several times. It was completely redesigned in 2005, when the final version was released. In 2008 ICT Automatisering released a wide range of services for DVTk.

DVTk's advantages were its flexibility to different eHealth standards and that it was released as an open source project, thus benefiting from the interest shown by different categories of stakeholders from healthcare communities. Despite that, DVTk did not attract the hoped amount of interest from the HL7 community.

4. HL7 V2 Testing Toolkit from NIST

The National Institute of Standards and Technology (NIST) has accorded much attention to the subject of testing the conformance of HL7 V2-based applications. NIST HL7 V2 Testing Toolkit provides means of testing HL7 V2 applications using message profiles [NIS11]. The core of this toolkit is a set of Java APIs that provide functionalities such as message generation and message validation. NIST developed a set of tools based on these APIs, such as the MessageMaker, a testing tool that can be used for defining suites of test message instances for a given profile [NIS04]. NIST is tightly collaborating with members of the HL7 V2 testing community, such as DVTk [POT07], IHE Gazelle [GAZ], Cypress [CYP] and CCHIT [CCH].

5. Test framework for automated interoperability testing based on TTCN-3

The test framework proposed by Vega et al. in [VEG08] and [VEG10] describes a methodology for testing Healthcare Information Systems (HIS) using Testing and Test Control Notation version 3 (TTCN-3). The test framework was validated on an IHE HL7 V2 integration profile, i.e. IHE Patient Care Device (PCD)[PCD12]. The solution was also presented at IHE Connectathon [CON]. The thesis considers this solution the most appropriate approach to match the goals of testing the conformance of eHealth applications. Even though this solution was designed for V2 only, and cannot be adapted to test V3 applications, it proves that TTCN-3 is powerful enough to be used as the basis for a generic HL7 V3 conformance testing framework.

4.3.2 Tools for testing HL7 V3-based applications

1. v3Generator

HL7 GForge [HGF] is a platform that helps HL7 developers manage their projects. Tools available on the GForge site support the development of HL7 V3 messaging standard. One of the tools available on GForge is v3Generator [MCK], a powerful utility that can be used for generating a variety of V3 artefacts, which include:

- schemas for data types, vocabulary, Hierarchical Message Definitions (HMDs) and interactions;
- HTML table views for HMDs;

- serializable static models (R-MIM/SIM, HMD, etc.).

V3Generator is useful when testing HL7v3 applications that don't adhere to a particular IHE integration profile. Generation of the models and schemas may automate much of the testing process.

2. HL7 V3 Test Harness

HL7 V3 Test Harness (TH) is a project that targets the conformance testing of HL7 V3-based applications [DEL10]. TH is a free tool that provides a Graphical User Interface (GUI) only for Windows platforms. TH has not reached maturity and several bugs have been reported and not fixed yet. Moreover, it seems that the support is discontinued currently. TH is limited in terms of flexibility provided to the testing team. It has few configurations available, is dependent on the platform and technologies of the SUT and is not modular enough to cover extensions to different integration profiles. Moreover, integration of other automation tools is not facile.

3. SOA Testing and Simulation of HL7 V3 messages using Schematrons

Another interesting method for testing the conformance of HL7 V3 is using Schematrons [VLI07]. Schematron rules provide a flexible way of capturing business domain specific rules for HL7 V3 messages. These rules can be used to improve interoperability between the eHealth applications. An attempt has been made to follow this approach and a Service Oriented Architecture (SOA) for testing and simulation of HL7 V3 messages using Schematron was developed. It was validated against the Dutch government's Health Information Broker (HIB), which provides a variety of web service-based healthcare services. Schematron was used for checking the correctness and validity of the V3 requests and responses.

4.4 Testing and Test Control Notation version 3

This section introduces Testing and Test Control Notation version 3 (TTCN-3), which is the underlying technology used in the new conformance testing approach presented in the thesis. TTCN-3 is an internationally standardized language designed specifically for testing. The TTCN acronym was initially derived from the name Tree and Tabular Combined Notation. The initial name of the language was changed with the release of the third version, when the language underwent a series of important improvements, which made it essentially a different language.

TTCN-3 denominates two different concepts, which have to be clearly distinguished: a) the testing language and b) the test system. The TTCN-3 testing language is used for defining an abstract testing behavior, which is not linked to the System Under Test (SUT). In addition to the testing language, TTCN-3 also denominates a test system, which is responsible for handling the testing execution and managing the communication with the SUT.

4.4.1 TTCN-3 testing language

TTCN-3 testing language is a European Telecommunications Standards Institute (ETSI) standardized scripting language, with the look and feel of a programming language but extended to include additional testing concepts such as, for instance, built-in data matching mechanism, distributed test system architecture and support for concurrent execution of test components.

The fact that the TTCN-3 testing language is standardized means that a test specification written in TTCN-3 is unambiguous. This is important to the testing team because it provides independence in terms of the testing tools developed by different vendors. The tools should execute a test suite in exactly the same way, which means that the testing team can use testing tools from different vendors in parallel, in the same testing environment.

The TTCN-3 testing language provides several presentation formats that help test designers define test scenarios. The most common is the textual format, referred to as the TTCN-3 *core notation* [MTS08], which is similar to conventional procedural programming languages. In addition to the TTCN-3 core notation, other formats such as *tabular format*, or *Message Sequence Chart (MSC) format* are supported. Figure 4.3 depicts different TTCN-3 presentation formats, as shown in [WIL05] – Figure 1.1, pg. 3.

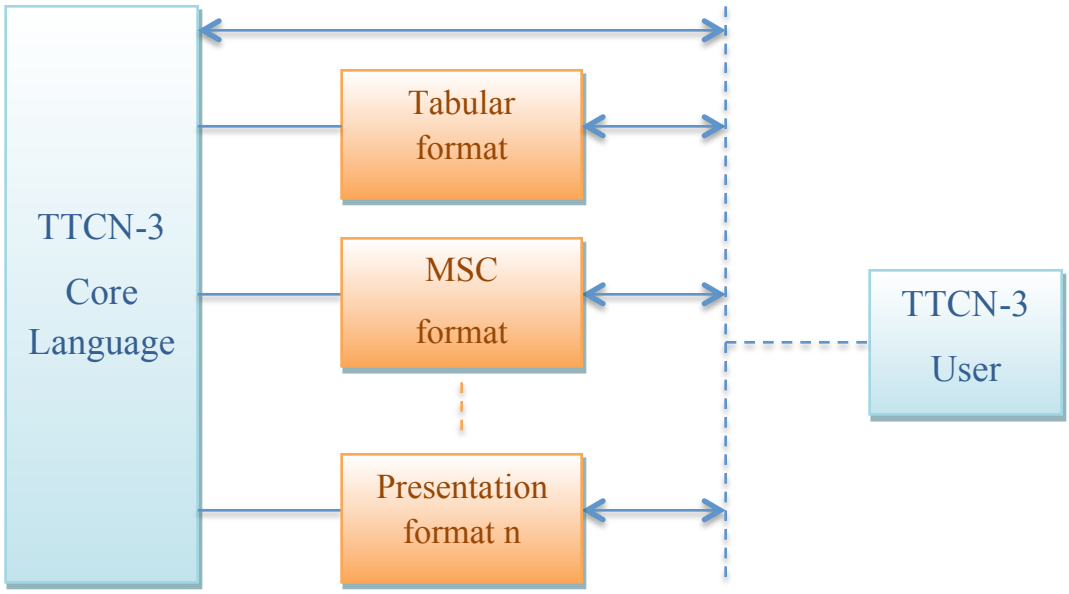

Figure 4.3. TTCN-3 presentation formats (adaptation of figure by Willcock et al. [WIL05])

As opposed to TTCN-2, the TTCN-3 testing language provides support for both procedure-based and message-based communication. The message-based communication mechanism is the appropriate test approach for many applications, however there are situations where testing cannot be adequately modeled using this paradigm. Therefore, version 3 of TTCN introduced the

procedure-based mechanism. In the message-based approach only the message semantics differentiate the entities involved in the communication. In contrast, the procedure-based approach makes a clear distinction between the roles of each involved entity. Another distinction between the two paradigms is that while the message-based approach is asynchronous, the procedure-based approach is commonly synchronous, meaning that the caller of a remote procedure is usually blocked until it receives a response or an exception from the called party.

4.4.2 TTCN-3 type system

The type system is one of the most important components of a test system. The correlation between application data or Protocol Data Units (PDUs) and the type system is a vital aspect of the test system design. TTCN-3 provides the means of mapping data types used by Systems Under Test (SUTs) to their representation in the TTCN-3 type system.

The TTCN-3 type system extends the basic constructs that usually have correspondents in programming languages to include additional testing specific concepts. The range of TTCN-3 data types is much larger than the one found in typical programming languages. The TTCN-3 type system is, thus, very complex.

TTCN-3 data types can be conceptually grouped in two categories:

- built-in data types;
- user-defined data types.

TTCN-3 type system possesses a large number of built-in data types, many of which (e.g. *boolean*, *integer*, *float*, or *charstring*) having close correspondence in other programming languages. However, TTCN-3 type system also includes a number of additional unique data types, such as *verdicttype*, *hexstring*, *objid*, or *default*, that distinguish it as a scripting language with a strong focus on protocol testing.

The TTCN-3 user-defined data types provide test designers much flexibility when modeling SUT's specific data types. They can be further divided into two categories: structured types (e.g. *record*, *enumerated*, *set*, or *union*) and list types (e.g. *array*, *record of*, or *set of*). The TTCN-3 user-defined data types are used for defining the TTCN-3 representation of the data model used by the SUT.

The core components of the TTCN-3 type system are the records and enumerated types. These two data types were used for defining the equivalent of the HL7 V3 data types in TTCN-3 specific format.

TTCN-3 records are constructs used for grouping related fields in a single type. TTCN-3 records are used to store data in a structured way. Field names within a record must be unique and their types may be either built-in or a user-defined. TTCN-3 records are arguably the most used types of the TTCN-3 type system.

TTCN-3 enumerated types are ideal for representing types that have small, finite sets of values. They are used for modeling types that take only a distinct named set of values called enumerations. Each enumeration has an identifier that is unique within the enumerated type. Operations on TTCN-3 enumerated types use only these identifiers and are restricted to the assignment, equivalence and ordering operators. TTCN-3 enumerated types are especially suited for encoding vocabulary terms.

Even though the definition of TTCN-3 data types is not a difficult process, the manual definition is unsuited when modeling a totally different type system. In the case of HL7 V3, manual definition of all the data types can be excluded from the beginning. Moreover, TTCN-3 does not provide generation tools for automating this process. Therefore, a generation tool was designed and implemented to address this issue. This code generating tool is detailed in Section 5.3.

4.4.3 TTCN-3 templates

TTCN-3 templates are used for defining information exchanged between the test system and the SUT. They are predominantly used in the message-based communication, where they define the actual messages exchanged. They are strongly linked to the type system. TTCN-3 data types such as records or enumerated types define logical structures for storing information, whereas TTCN-3 templates specify the information content.

TTCN-3 templates, in their simplest form, define a single value, which is usually used as an argument of a sending operation. However, their true power lies in the fact that they can group multiple values of different types, or even other templates, in a single flexible definition. As their name suggests, they are used for defining templates of messages, allowing test designers to concentrate on the relevant piece of information that needs to be transmitted and omit the information that is not relevant to that specific scenario.

TTCN-3 templates are tightly coupled to one of the most powerful features of the TTCN-3 testing language, the built-in matching mechanism. The matching mechanism is handled by the TTCN-3 test system component that manages the testing execution. It is responsible for interpreting two templates and analyzing the similarity between the two. Based on the similarity between the two, a verdict is set to the test case, generally indicating if the test failed or passed.

When defining a TTCN-3 test case for testing the conformance of HL7 V3-based applications, test designers create two TTCN-3 templates. The first one represents the input, which is designed to trigger a specific SUT behavior. The second template represents the expected output, which is derived from the SUT's software requirements specification. During the test case execution, the TTCN-3 matching mechanism verifies if the expected output matches the one received from the SUT, establishing if the SUT responded properly to the stimulus.

An important concern when testing HL7 V3-based applications using TTCN-3 is that templates are difficult to create. Their hierarchical structure can span on many levels, usually reaching more than twenty levels, which makes manual definition and maintenance cumbersome. Section

5.4 presents a generation tool and an editor that facilitate the creation and editing of TTCN-3 templates.

4.4.4 TTCN-3 test system

Using the TTCN-3 testing language, test designers can define test cases and the order in which they are executed. However, for the actual execution of the test cases, a test system is needed. The TTCN-3 test system can be defined as a set of interacting entities that implement specific test system functionalities [EGN10]. Figure 4.4 depicts the general architecture of a TTCN-3 test system, highlighting the main components and the relationship between them.

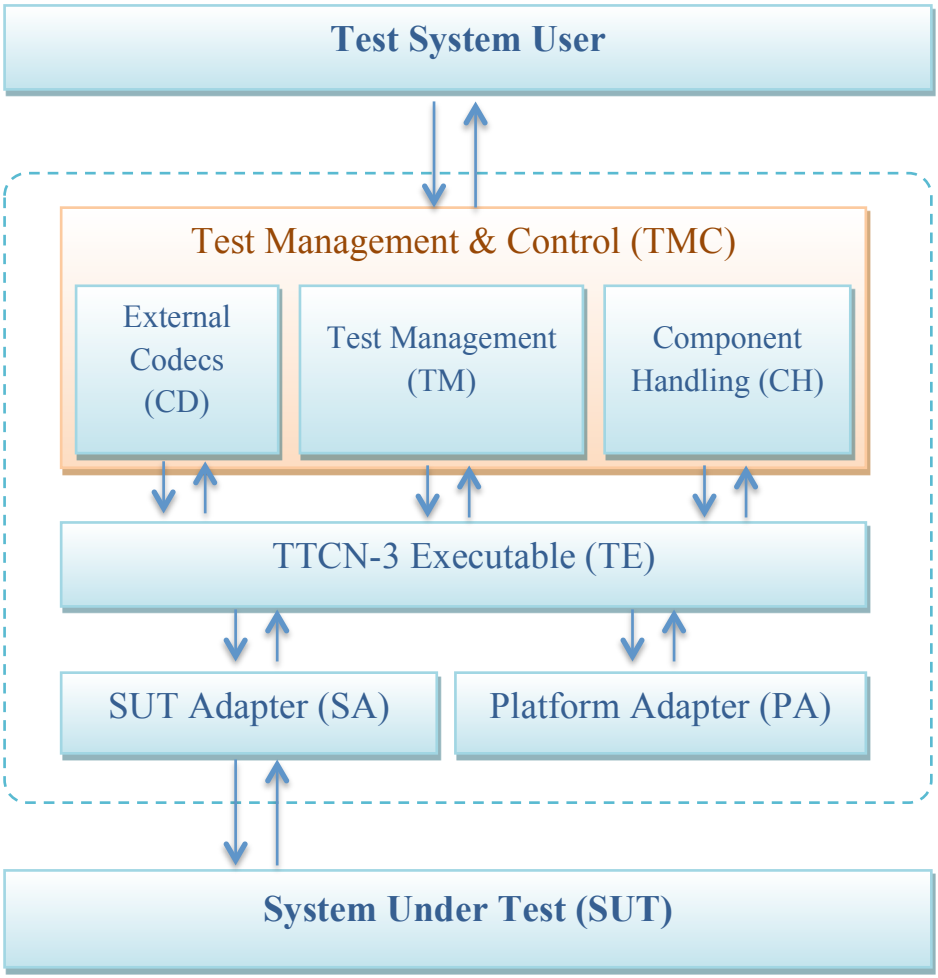

Figure 4.4. General architecture of the TTCN-3 test system

(adaptation of figure by Willcock et al. [WIL05])

The TTCN-3 test system architecture can be decomposed in three different layers. The Test Management Control (TMC) represents the first layer. TMC consists out of three components:

- External Codecs (CD) is responsible for encoding and decoding data between the TTCN-3 and SUT specific formats;
- Test Management (TM) represents the interface with the Test System User; it enables test designers to define the Abstract Test Specification (ATS);
- Component Handling (CH) handles the distributed execution of the test cases.

The core of the TTCN-3 test system, the TTCN-3 Executable (TE) component, can be found in the second layer. TE manages the execution of TTCN-3 statements. It depends on the services provided by the other two layers, i.e. TMC and the Adapters. TE is responsible for interpreting the behavior of the test scripts defined on the TTCN-3 testing language level and converting it to a suitable execution form. TE also includes a subcomponent named Runtime System (RS), which implements the TTCN-3 operational semantics.

The Platform Adapter (PA) implements TTCN-3 external functions and provides timing mechanisms. The SUT Adapter (SA) adapts the message or procedure-based communication between the TTCN-3 test system and the SUT to the particular execution platform of the test system.

The available TTCN-3 test systems usually provide default implementation of TM and CH. This is not the case of CD, SA or PA, since they cover aspects of the test system that are either test suite or SUT specific. The responsibility of implementing these components is left to the testing team. Throughout the thesis, the CD component is referred to as the Codec, while the SA component is referred to as the Adapter.

4.4.5 TTworkbench

TTworkbench is an integrated test development and execution environment. TTworkbench is developed in Java and is based on the Eclipse platform. TTworkbench provides implementation for the TTCN-3 test system components and for the standardized communication interfaces that link them. In addition to the TTCN-3 test system, TTworkbench contains several modules that aid test designers to define and execute test suites. TTworkbench includes the following important five components or features:

1. TTman

This component is responsible for the management, execution and analysis of the TTCN-3 test suites. It provides a variety of logging choices, from plain text to graphical layout on different levels. TTman is the component that handles the selection and configuration of the Adapter component and runtime plugins.

2. CLEditor

CLEditor is the built-in editor for designing test scenarios in TTCN-3 language. It features text formatting, syntax highlighting, text annotations and error reporting. CLEditor also offers the option of automatic generation of HTML formatted documentation.

3. TTthree

TTthree compiles TTCN-3 modules into test executables. It provides support for two of the most important components of the TTCN-3 test system, i.e. the Codec and the Adapter, via the two standardized interfaces, TTCN-3 Control Interface (TCI) and TTCN-3 Runtime Interface (TRI), respectively.

4. GFT Editor

The GFT Editor can be used for specifying and documenting tests in graphical mode. It provides support for the TTCN-3 presentation formats shown in Figure 4.3. In addition, it can be used for automatic generation or exporting of testing reports as documentation items.

5. TTDebug

Testing execution can be analyzed with this debugger. It includes a GUI for defining breakpoints and watchpoints and can be used for suspending or resuming running tests, analyzing stack traces for different components, and viewing and manipulating statuses of timers, ports, or message queues.

TTworkbench is in the author's opinion the best professional suite that integrates the TTCN-3 test system components, which provides plug-and-play capabilities for setting up the Codecs and the Adapters. It offers many other features that help testers design test scenarios, automate the testing solution and analyze results during runtime or as a documented report.

5 TTCN-3 FRAMEWORK FOR CONFORMANCE TESTING OF HL7 V3-BASED APPLICATIONS

5.1 Testing methodology

This section presents the methodology for defining the Abstract Test Specification and the conformance testing approach. It also describes the testing workflow and the roles of various TTCN-3 test system components in the testing process [EGN12d]. External components, which are used in the testing process but are not part of the TTCN-3 test system, are also presented in this section.

Properties of the System Under Test

The System Under Test (SUT) is the tested system. The SUT chosen as the case study is a Patient Management Information System, i.e. a medical application that enables the management of clinical data for different patients. The detailed particularities of this SUT are presented in Section 5.8. Briefly, it has two important characteristics that influence the testing process:

- it's based on a HL7 V3 integration profile – IHE QED;
- it is deployed as a web service.

The first characteristic influences the definition of testing inputs and outputs. Inputs should be structured as *QED Get Care Record Profile Queries* and outputs should be structured as *QED Get Care Record Profile Responses*. For readability reasons, the two types of message will be henceforth referred to as QED Query and QED Response, respectively.

The type of communication between the test system and the SUT is influenced by the SUT's second characteristic. The communication is based on the Simple Object Access Protocol (SOAP) [BOX00]. Therefore, the IHE QED messages need to be encapsulated into SOAP messages.

Conformance testing overview

As detailed in Chapter 4, testing the conformance of a system with the HL7 V3 messaging standard means verifying if the system's behavior is compliant with the standard specification. Conformance testing is a type of black-box testing, i.e. the test system tests the functionality of the system, without considering its internal structure. Briefly, a simple conformance testing procedure implies the following steps:

Step 1. Defining an input message
Step 2. Defining the expected response of the SUT; the expected response is derived from the standard's specification
Step 3. Defining the test case for a specific scenario, setting up the test system and starting the execution of the test case
Step 4. Triggering the test system to send the input message to the SUT
Step 5. Intercepting the response message from the SUT
Step 6. Comparing the expected response with the actual response received from the SUT
Step 7. Setting the verdict of the test case

The testing environment

The testing process can be logically divided into two sub-processes, namely the definition of the Abstract Test Specification (steps 1-2) and the management and execution of the test cases (steps 3-7). A test system that provides support for the two sub-processes can be implemented based on the following TTCN-3 technologies:

- the TTCN-3 testing language can be used when defining the ATS;
- TTCN-3 test system components can be used for setting up the test environment, executing test cases and establishing verdicts.

The selected TTCN-3 test system is part of the TTworkbench testing framework. TTworkbench [TTW] was preferred because it integrates:

- TTCN-3 test system components;
- *TTthree* – a TTCN-3 compiler;
- *CL Editor* – a TTCN-3 specific editor;
- *TTdebug* – a TTCN-3 source code level debugger;
- *GFT Editor* – a visualization tool.

TTworkbench also supports automation capabilities and other interesting features, such as the possibility to change test system components at runtime. TTworkbench is based on the Eclipse platform [RIV04] and is developed using Java technologies. This means that the test system components and the communication interfaces between them, the mapping rules and the internal structures are developed in Java.

Abstract Test Specification

The Abstract Test Specification, or Abstract Test Specification (ATS) represents a collection of simple tests defined using the TTCN-3 testing language. ATS defines the test behavior independently of the SUT's properties. When defining the ATS, test designers do not take into account any system-specific information, such as message encodings, communication channels, etc.

The definition of ATS depends on the TTCN-3 type system. Before defining test cases for HL7 V3-based applications using TTCN-3, test designers must ensure that the type system provides support for specific data types. In this particular case, the TTCN-3 type system should include all the HL7 V3 data type definitions referenced by the IHE QED integration profile. Figure 5.1 shows the dependency between the test case specifications and the type system.

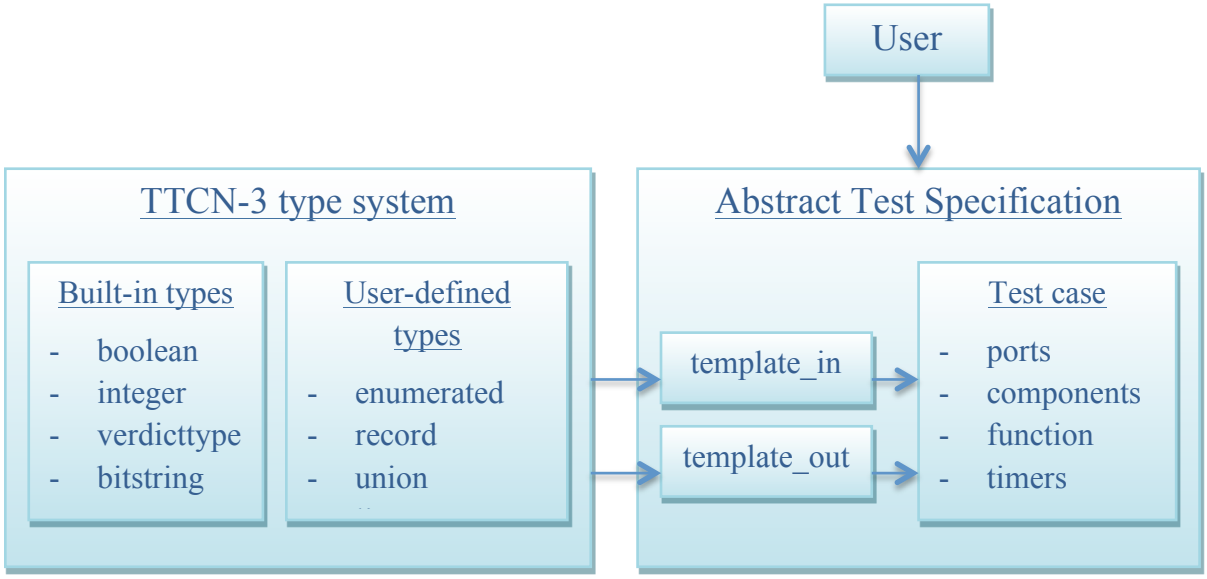

Figure 5.1. Defining the Abstract Test Specification

The task of manually creating user-defined TTCN-3 data types is difficult and prone to errors. Section 5.3 discusses about this problem and describes a methodology for automatic generation of user-defined data types from the XML Schema definitions of the HL7 V3 data types.

Testing workflow

The TTCN-3 source code defined in the ATS is not executable by itself and is only conceptually linked to the SUT. A testing system contains a set of components that manage the execution of the test cases in a real scenario, with a real SUT. Figure 5.2 presents the message flow for a simple test case, highlighting the TTCN-3 test system components involved in the testing process.

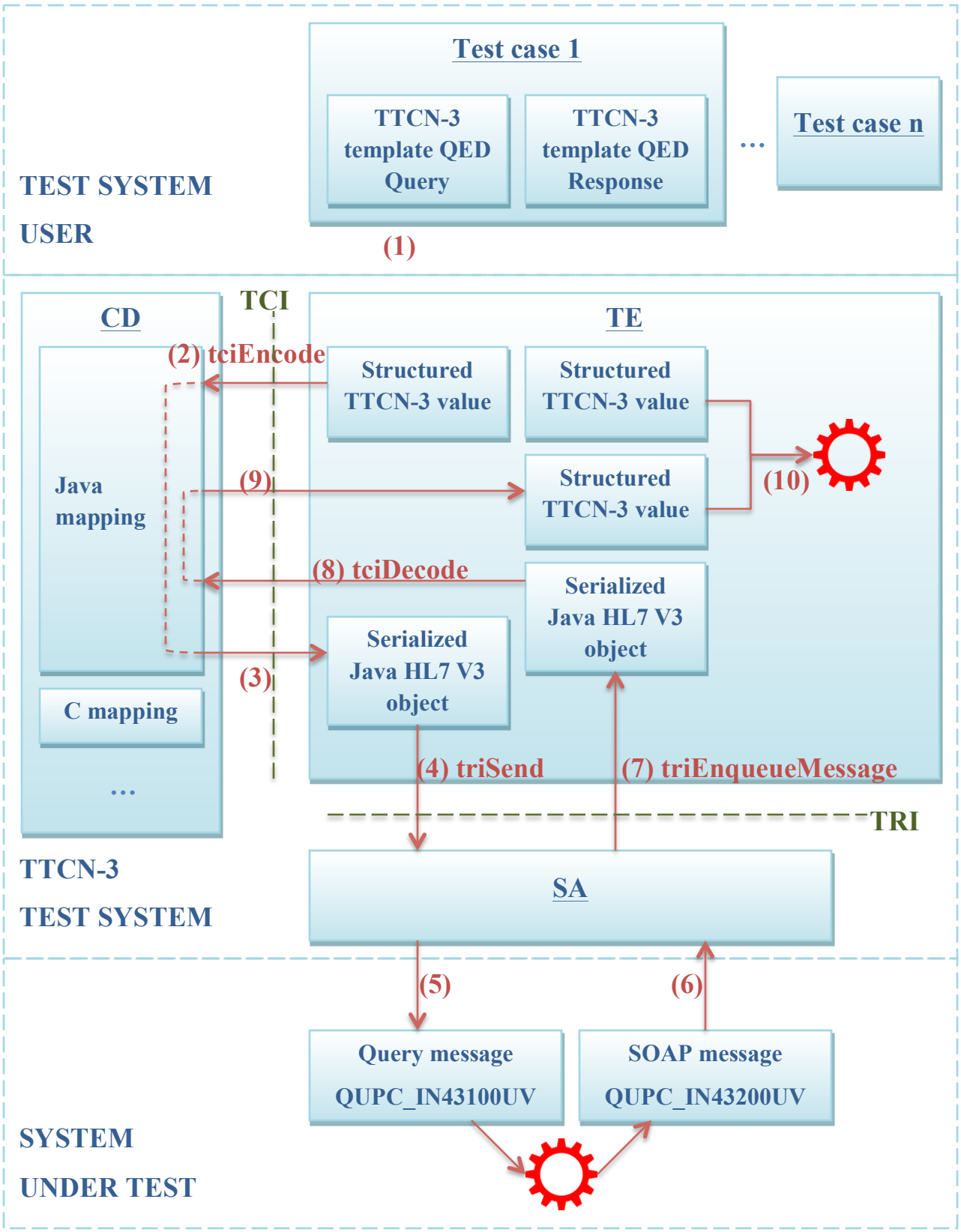

Figure 5.2. Message flow of the testing process

The execution of a simple test case is described in the steps below. The initialization phase of the test system is not detailed here.

Step 1. The TTCN-3 Executable (TE) starts the execution of the test case
Step 2. TE interprets the TTCN-3 templates that define the QED Query and the QED Response and creates one *structured TTCN-3 value* for each of them (1)
Step 3. The *structured TTCN-3 value* corresponding to the QED Query is passed to the Codec (CD) through the TTCN-3 Control Interface (TCI). This is performed by invoking the method *tciEncode* (2)
Step 4. CD encodes the *structured TTCN-3 value* into a HL7 V3 Java object, which is serialized to a byte stream
Step 5. TE receives the *tciEncode*'s returned value, i.e. the byte stream (3)
Step 6. TE forwards the serialized HL7 V3 Java object to the SUT Adapter (SA), through the TTCN-3 Runtime Interface (TRI). This is performed by invoking the method *triSend* (4)
Step 7. SA encapsulates the Java object into a SOAP message
Step 8. SA sends the SOAP message to the SUT (5)
Step 9. The SUT receives the SOAP message encapsulating the QED Query, i.e. *QUPC_IN43100UV* and replies with another SOAP message containing the QED Response, i.e. *QUPC_IN43200UV* (6)
Step 10. SA decapsulates the Java object from the SOAP message
Step 11. SA sends the Java object to TE, through the TRI interface. This is performed by invoking the method *triEnqueueMessage* (7)
Step 12. TE forwards the Java object to the Codec, through the TCI interface. This is performed by invoking the method *tciDecode* (8)
Step 13. CD decodes the *structured TTCN-3 value* from the HL7 V3 Java object
Step 14. CD sends the *structured TTCN-3 value* to TE (9)
Step 15. TE compares the *structured TTCN-3 value* received from the SUT with the one defined in the test case and sets a verdict (10)

Testing prerequisites

An important prerequisite of the proposed testing process is the existence of a TTCN-3 type system that contains the HL7 V3 data types definitions. More specifically, it is sufficient that the TTCN-3 type system contains the definitions of the data types and messages required in a specific healthcare domain where the SUT operates. For the chosen case study, the TTCN-3 type system should contain HL7 V3 data types that are referenced by IHE QED.

Another important prerequisite is that the TTCN-3 type system is extended to the programming language that was used for implementing the TTCN-3 test system. In other words, the same data types defined in the TTCN-3 type system should also be defined in the programming language chosen for the development of the TTCN-3 test system components. This is preferred in order to

support an easy integration of automation mechanisms in the TTCN-3 test system. Therefore, the HL7 V3 data types and messages should also be defined in Java.

It is problematic to meet both of the two mentioned preconditions. Even though TTCN-3 is a standardized language, it is difficult to find TTCN-3 data type definitions even for common HL7 V3 integration profiles. Moreover, it is ambitious to synchronize them with the Java definitions, too. For this reasons, the two prerequisites are bound into one single precondition: the existence of a set of XML Schemas (XSDs) that define the data types.

This research shows that the XML Schemas are sufficient for generating both the TTCN-3 data types and the Java classes. Section 5.2 analyzes the decision of choosing the XSDs over the other metamodel formats and describes the generation process for the set of Java classes. Section 5.3 presents the automatic generation of the TTCN-3 data types.

5.2 Automated generation of Java classes defining HL7 V3 data types

Automation of the testing process hinges on the existence of a properly defined set of Java classes that define HL7 V3 data types. The existence of these classes is an important requirement in the following situations:

- the tester performs an automatic generation of TTCN-3 data types (see Section 5.3);
- the Codec performs a TTCN-3 to Java encoding, or a Java to TTCN-3 decoding;
- the Adapter encapsulates/decapsulates data into/from an SUT supported format;
- the tester implements a Test Data Converter based on the Codec (see Section 5.7.2.2).

There are two primary types of Java data types generation:

- Model Interchange Format (MIF) driven generation;
- MIF-derived model driven generation (e.g. UML/XMI, XML ITS Schema, RIM ITS Schema, etc.).

Table 5.1 shows the comparison between the MIF model and MIF-derived models, highlighting each one's advantages and disadvantages. The table also covers the existing tools that can be used for class/code generation.

Table 5.1. Comparison between MIF and MIF-derived models

Source	Advantages	Disadvantages	Tools & technologies
MIF	• Normative (full) specification of a V3 model, inclusive of documentation • Full validation support	• No cross-industry tools for class generation • MIF structure is more complex • Includes testing-irrelevant structures (publication, requirements-analysis, etc.)	• MDHT • Everest Framework • JavaSIG
Derived model	• Wide availability of cross-industry tools for class generation, parsing, etc. • Simplicity • Readability	• MIF-derived models cannot express all model requirements (less expressive than MIF) • HL7 V3 template validation capabilities (HL7 V3 templates are defined in MIF)	• JAXB • XML Beans • Everest API Generator (XML-based ITSs) • UML/EMF

The analysis comprised in Table 5.1 shows that MIF model, the formal definition of the HL7 metamodel is more complex and has a lower support from the developers' community. However, the first attempt on generating the Java classes was based on this approach. MDHT [OHT10] concentrates exclusively on the HL7 CDA, while the Everest Framework [FYF12] was primarily designed for .NET and is currently expected to extend to Java. However, the third option, i.e. the HL7 Java SIG Project [JSI07], proved impractical.

Java SIG provides an API for generating Java classes from MIF formatted HL7 definitions. However, the API is still very low-level and most of the times the generated classes resemble the structure of the MIF, as opposed to higher-level concepts such as classes, associations, attributes, etc. Moreover, the generated classes don't have a unified format. For instance, the primary issue is that not all the class fields have getter and setter methods, which makes them unsuitable for runtime manipulations.

For this reasons, another approach was considered. The second attempt consisted of generating the Java data type definitions using a MIF-derived model [EGN11]. The options considered here were:

- schemas for XML-based ITSs;
- other type of metamodel formats, such as UML/XMI, Eclipse Modeling Framework (EMF), etc.

Because of the high developers' support and wide availability of tools, the first alternative was chosen, i.e. generation of data types based on Schemas for XML-based ITSs. The XML-based ITSs can furthermore divided into different alternatives. The most common XML-based ITSs are XML ITS and RIM ITS, which are compared in Table 5.2.

Table 5.2. Comparison between XML-based ITSs

	XML ITS	RIM ITS
Availability	Since the beginning of HL7 V3	2010
Based on	The serialization of SIM (R-MIM)	The serialization of RIM
Schemas	Separate schema for each SIM (R-MIM)	Single schema for RIM
XML elements' names	Based on clone names	Based on RIM core class names

The final choice was to use schemas for XML ITS. The XML Schemas (XSD) are provided by HL7 as part of the HL7 Development Framework (HDF). Therefore, the approach consisted of generating Java classes from the XSDs defining the HL7 V3 data types using Java Architecture for XML Binding (JAXB) [VAJ04]. The JAXB generated classes have an uniform structure and provide instantiation and object manipulation during runtime.

An example of Java class generated with JAXB is presented below. Figure 5.3 shows a snippet from an XSD file that defines *COCT_MT010000UV01.AccomodationEvent*. Figure 5.4 shows the equivalent Java definition generated with JAXB.

```
<xs:complexType name="COCT_MT010000UV01.AccomodationEvent">
    </xs:sequence>
        <xs:group ref="InfrastructureRootElements"/>
        <xs:element name="code" type="CD" minOccurs="0" maxOccurs="1"/>
        <xs:element name="effectiveTime" type="SXCM_TS" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="reasonCode" type="CE" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="InfrastructureRootAttributes"/>
    <xs:attribute name="nullFlavor" type="NullFlavor" use="optional"/>
    <xs:attribute name="classCode" type="ActClass" use="required" fixed="ACCM"/>
    <xs:attribute name="moodCode" type="ActMood" use="required" fixed="EVN"/>
</xs:complexType>
```

Figure 5.3. Definition of HL7 complex type in XML Schema

```
public class COCTMT010000UV01AccomodationEvent {
    protected List<CS> realmCode;
    protected II typeId;
    protected List<II> templateId;
    protected CD code;
    protected List<SXCMTS> effectiveTime;
    protected List<CE> reasonCode;
    @XmlAttribute
    protected List<String> nullFlavor;
    @XmlAttribute(required = true)
    protected List<String> classCode;
    @XmlAttribute(required = true)
    protected List<String> moodCode;

    public List<CS> getRealmCode() {
        if (realmCode == null) {
            realmCode = new ArrayList<CS>();
        }
        return this.realmCode;
    }

    //rest of the getters and setters
}
```

Figure 5.4. Definition of HL7 complex type generated with JAXB

For a full integration in the automatization process, these classes had to be postprocessed. Adjustments were required, because of the particularities of the TTCN-3 test system. For instance, the interface that connects TE with CD, i.e. the TCI interface shown in Figure 5.2, enforces that the objects passed as arguments to *tciEncode* and *tciDecode* are serializable.

The postprocessing procedure includes four adjustments:

Step 1. Adding the "implements Serializable" to the class signature; this includes adding the "*import"* statement

Step 2. Changing the "List<Serializable>" and "ArrayList<Serializable>" to "List<String>"

Step 3. Removing the unnecessary JAXBElement returning methods from *ObjectFactory.java*

The *ObjectFactory* class defines a set of methods that return JAXBElements, as shown in Figure 5.5. These methods generate compilation errors. As they are not needed for the testing solution, they were removed.

```
/**
 * Create an instance of {@link JAXBElement }{@code <}{@link COCTMT910000UVStudent }{@code >}}
 *
 */
@XmlElementDecl(namespace = "urn:hl7-org:v3", name = "student", scope = REPCMT000300UV01Informant.class)
public JAXBElement<COCTMT910000UVStudent> createREPCMT000300UV01InformantStudent(COCTMT910000UVStudent
    value) {
    return new JAXBElement<COCTMT910000UVStudent>(_REPCMT000100UV01Author3Student_QNAME,
        COCTMT910000UVStudent.class, REPCMT000300UV01Informant.class, value);
}
```

Figure 5.5. JAXB generated code that needs post-processing

Step 4. Removing all JAXBElement declarations; for example, "JAXBElement<Type> variable" is converted to "Type variable"

The post-processing scripts were implemented in Python. Regular expressions were used for finding and replacing statement definitions from the Java classes. Figure 5.6 shows snippets from the first post-processing script, i.e. the one that adds the "implements Serializable" clause. The first part depicts the regular expression used for obtaining the location of the class signature. The second method, i.e. "__repl__", implements the actual statement addition.

```
    accessModifiers = "public?"
    otherModifiers = "(abstract|final|strictfp)?"
    whiteSpace = "(\s|\n)*"
    className = "[a-zA-Z0-9_-]+"

    classDefinition = accessModifiers + \
                      "(" + whiteSpace + otherModifiers + ")*" + \
                      whiteSpace + "class" + whiteSpace + className + \
                      "(" + whiteSpace + "extends" + whiteSpace + className + ")?" + \
                      "(" + whiteSpace + "implements" + \
                      "(" + whiteSpace + className + whiteSpace + ",?)*" + ")?" + \
                      whiteSpace + "(?={)"

    self.pattern = re.compile(r"%s" %classDefinition, re.S | re.I)

    importDefinition = "(?=import)"
    self.patternImport = re.compile(r"%s" %importDefinition, re.S | re.I)


def __repl__(self, match):
    if match is None:
        return None

    str = re.sub(r"(\s|\n)+", " ", match.group(0))
    if re.search("\s+implements\s+\w+", str, re.S | re.I):
        str += ", Serializable "
    else:
        str += "implements Serializable "

    return str
```

Figure 5.6. Snippets from post-processing script

5.3 Automated generation of user-defined TTCN-3 data types

As shown in Figure 5.2, the process of defining the Abstract Test Specification depends on the TTCN-3 type system. More specifically, the definition of TTCN-3 templates depends on the user-defined TTCN-3 data types. Defining the TTCN-3 data types manually is difficult, expensive, and susceptible to errors. This section proposes a methodology for generating TTCN-3 data types such as records or enumerated types automatically [EGN11]. Furthermore, a TTCN-

3 template generator is proposed to fully automate the definition of the Abstract Test Specification (ATS). The generators are implemented for the IHE QED integration profile. Yet, the methodology is generic and does not depend on the properties of any HL7 V3 integration profile.

5.3.1 The architecture of the TTCN-3 records and enumerated types generator

The generator was designed with the clear separation of functionalities in two different modules that can be easily substituted when test designers choose to replace the HL7 V3 integration profile. The architectural overview of the TTCN-3 records and enumerated types generator is presented in Figure 5.7.

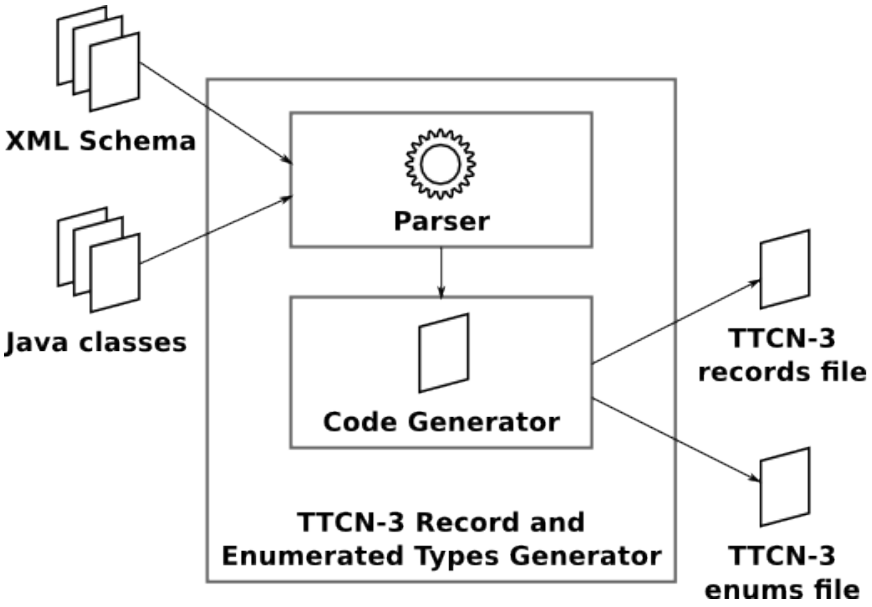

Figure 5.7. Architectural design of the records and enumerated types generator

The core components of the generator are the *Parser* and the *Code Generator*. The *Parser* is responsible for extracting the definitions of the HL7 V3 data types and messages. The *Code Generator* uses the information extracted by the *Parser* to generate two TTCN-3 modules that contain the records and enumerated types.

The *Parser* takes one of the two inputs: either the XML Schemas that contain the definitions for the HL7 V3 data types and messages for a specific HL7 V3 integration profile, or a set of classes that define the same data types and messages in Java language. In the first case, the *Parser* automatically generates the Java classes using the generation methodology presented in Section 5.2. The *Parser*'s output is a set of object instances that are passed to the *Code Generator*. The

Code Generator evaluates the fields of the Java objects and converts them into TTCN-3 records or enumerated types.

5.3.2 The implementation of the TTCN-3 records and enumerated types generator

Java was chosen for implementing the TTCN-3 enumerated types and records generator. Since the adopted TTCN-3 test system is developed in Java, it was natural to select the Java technology to enable a good integration of the generator into the testing process.

In this implementation, the *Parser* uses the Java classes generated with JAXB as input. This approach is backed by the Java mechanisms that enable full automation of the generation process. Algorithm 1 describes the process of generating the TTCN-3 records and enumerated types.

Algorithm 1. TTCN-3 records and enumerated types generation process

Inputs
- *hl7 V3_cls_dir* – path pointing to the directory that contains the Java classes that define the HL7 V3 data types and messages

Outputs
- *records.ttcn3* – the TTCN-3 module containing the records' definitions
- *enums.ttcn3* – the TTCN-3 module containing the enums' definitions

```
1.  for each file in hl7_cls_dir do
2.      class_name ← normalize file name
3.      cls ← use classloader to load the class with class_name
4.      if cls is Enum then
5.          str ← convert_to_enum(cls)
6.          update enums.ttcn3 with str
7.      else
8.          str ← convert_to_record(cls)
9.          update records.ttcn3 with str
10.     end if
11. end for
12.
13. // other function definitions defined in Code Generator
14. function convert_to_record(cls)
15.     str ← empty string
16.     for each field in cls.declaredFields do
17.         type ← field.genericType
18.         if type is parameterized type then
19.             if type is list then
20.                 raw ← type.rawType
21.                 case raw of
22.                     // for each conversion call proper CodeGenerator method
```

```
23.                 List of JAXBElement: record_field ← convert raw to TTCN-3 record field
24.                 List of Serializable: record_field ← convert raw to TTCN-3 record field
25.                 List of String: record_field ← convert raw to TTCN-3 record field
26.                 List of integer: record_field ← convert raw to TTCN-3 record field
27.                 List of non primitive: record_field ← convert raw to TTCN-3 record field
28.             end case
29.           else if type is JAXBElement then
30.             record_field ← convert type to TTCN-3 record field
31.           end if
32.         else
33.           // handle cases for Array, String, Boolean, BigInteger, Double and non-primitive
34.         end if
35.         update str with record_field
36.       end for
37.       return str
38. end function
39.
40. function convert_to_enum(cls)
41.       str ← empty string
42.       for each constant in cls.enumConstants do
43.         enum_field ← convert constant to TTCN-3 enum field //call CodeGenerator method
44.         update str with enum_field
45.       end for
46.       return str
47. end function
```

Briefly, the generation process is composed of the following steps:

- the generator is fed an input consisting of the path to the directory where the Java classes are located. A classloader is created for this specific directory;
- the Java files from the input directory are successively loaded and inspected to obtain the Java types. The type, which can be either Enum or Class, has a direct correspondent to the TTCN-3 data types. Java Enums correspond to TTCN-3 enumerated types, while Java Classes correspond to TTCN-3 records.

Relevant information for the conversion to TTCN-3 enumerated types or records can be obtained at runtime, with the help of Java Reflection. The translation of Java classes to TTCN-3 records proceeds with the following steps:

Step 1. The class name is obtained from the file name

Step 2. The class is instantiated to get access to important members of the class, such as fields or methods

Step 3. An equivalent TTCN-3 record is generated with the same name

Step 4. The record's fields are generated based on the information extracted from the Java object. For flexibility reasons, the fields of the equivalent record are generated as *optional* by default. This means that the fields are generated with no constraints, which provides test designers the flexibility to create different types of TTCN-3

Following the generation process, two files are created:

- *records.ttcn3*, the TTCN-3 module containing the TTCN-3 records;
- *enums.ttcn3*, the TTCN-3 module containing the TTCN-3 enumerated types.

Test designers are not required to perform any post-processing operations on the two output files. After the generation process ends, the two modules can be directly imported in other TTCN-3 modules.

Even though the TTCN-3 records and enumerated types generator was implemented for a specific HL7 integration profile, i.e. IHE QED, its modular design enables extensions to other HL7 V3 integration profiles, as well. The extension is conditioned by the existence of either a set of Java classes that define the HL7 V3 data types and messages for the particular HL7 V3 integration profile or a set of XSDs from which the Java definitions can be automatically generated.

5.4 The TTCN-3 templates editor and generator

The TTCN-3 template generator is a simple, but powerful tool, which enables further automation for the testing process. In the absence of this tool, TTCN-3 templates have to be manually defined, preferably using a specialized TTCN-3 editor, such as *CL Editor*, which is the TTworkbench's integrated default editor. However, defining TTCN-3 templates for HL7 V3 messages is difficult even for experienced test designers. TTCN-3 templates have a tree-like structure that commonly spans on more than 20 levels. This kind of structure is difficult to define and maintain.

An alternative to the TTCN-3 format is the XML format, which is more human-friendly and more adaptable to adjustments. To avoid natural human errors, it is recommended that the TTCN-3 template's content be defined in the XML format. The proposed tool enables automatic conversion from XML to TTCN-3 and enables test designers to edit and define TTCN-3 template content in any one of the two formats.

The TTCN-3 template generator is developed in Java. It provides an intuitive and easy to use Graphical User Interface (GUI). Figure 5.8 depicts the TTCN-3 template generator's user interface. The implementation follows Model View Controller (MVC), a design pattern that provides guidelines for dividing the application into three components, in order to separate the user interaction from the application data and business rules [KRA88].

As shown in Figure 5.8, there are two editing areas, namely *xml* and *ttcn-3*. The test designer can switch between the two areas by changing the tab in the upper part of the GUI. The two areas integrate two powerful editors that have similar capabilities with stand-alone editors.

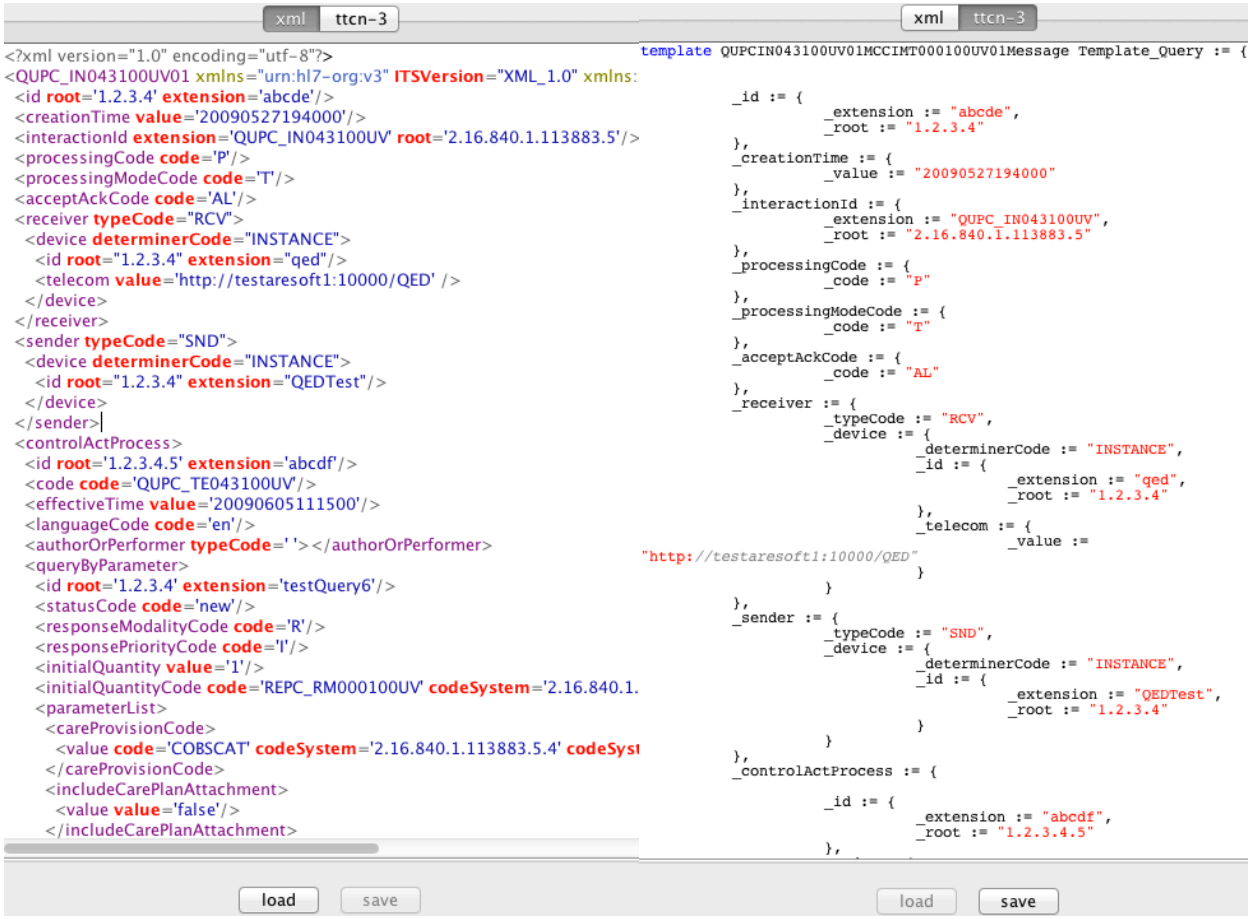

Figure 5.8. Interface of the TTCN-3 template generator

The editors provide features such as auto-completion, highlight of keywords, or automatic indentation. Language specific terms, such as strings, attributes, or namespaces, are highlighted as well. These features reduce the time required for defining HL7 V3 messages.

The automated conversion between the two formats is done when switching between the tabs. When changing to the *ttcn3* tab, the text area is updated and the editing environment is changed. Editing is also available in *ttcn3* mode and the test designer is able to make further changes to the TTCN-3 template.

The test designers are responsible for adding the final statements to the generated TTCN-3 template. This happens because there is no reference to the TTCN-3 records file, or to other TTCN-3 modules that define TTCN-3 templates. The TTCN-3 template generator has no

information about the TTCN-3 type system that the test designer intends to use. Thus, it is the test designer's responsibility to add the necessary imports after the finalization of the TTCN-3 template generating process.

The *Model* component of the MVC describes three types of actions:

- load an XML file that contains a previously defined message content;
- convert XML to TTCN-3;
- save the TTCN-3 template in a new TTCN-3 module.

The steps of the XML to TTCN-3 conversion algorithm are presented below:

Step 1. Parse the XML file
Step 2. Traverse the tree-like structure
Step 3. Recursively translate each node defined in the XML file into the corresponding TTCN-3 record
Step 4. Handle XML attributes
Step 5. Handle each child of the current node defined in the XML file
Step 6. Indent the final result

5.5 Overview of the particularities of defining input test data for testing HL7 V3-based applications

Independently of a test specification language, the test data adequacy criterion remains among the most important factors for the effectiveness of a test. Approaches to study the relevance of the selected test data include the notion of distance between programs, where programs are the tested systems. A test data set is considered to be adequate if it distinguishes the tested program from other programs that are sufficiently far from it, i.e. produce different input-output behavior. Close programs producing same results are considered identical [DAV88]. A similar concept of Adaptive Random Testing is provided in [CIU06], where random test inputs rely on the notion of distance between the test values.

It is widely accepted in the literature that dividing the input domain in equivalence classes reduces testing costs, without affecting the quality of the testing process. Many researchers address the problem of creating equivalence classes for a given input data domain of an SUT.

Data clustering is a powerful method of grouping input data based on their properties. It is commonly used when data sampling and processing is needed [JAI99]. Test designers can define clusters in such way that input data from the same clusters trigger identical responses from the SUT. Instead of testing the behavior of SUT exhaustively, it is sufficient to choose one or more representatives from each cluster as testing input, which eliminates redundant testing and reduces testing costs.

Cluster analysis is a method for finding groups of clusters in a population of objects. The goal of cluster analysis is to divide a population in such way that objects with similar attribute values are placed in the same cluster, the reunion of all clusters form the entire input domain, and clusters are disjunctive. The similarity is measured by dissimilarity metrics, such as the Euclidean Distance [GOW82], or the Google distance suggested in [CIL07], which is a semantic distance function between pairs of words or terms.

There are many researchers that study test data variance as a measure of test data dispersion over the input domain. Given the very large number of possible inputs that could be used when testing eHealth applications, the goal is to minimize the number of test cases in a test suite while maintaining test effectiveness as high as possible. Data variance is investigated by inspecting the proximity of objects. The survey of Jain et al. analyzes how clustering process facilitates grouping of elements from a given collection into meaningful similar data points [JAI99]. Measurement of the proximity (similarity) between data points is accomplished through well-defined partition clustering algorithms.

While clustering has many advantages, it also has some drawbacks. In some cases the process of clustering is too expensive and is not worth it. Creating clusters of HL7 V3 messages is especially cumbersome, because there is no customizable HL7 V3 message generator available. In absence of a message generator, clusters can be derived from repositories of existing messages. However, finding such repositories is difficult, because of the way HL7 V3 messages are structured and used in different healthcare domain.

The HL7 V3 standard for messages defines the Reference Information Model (RIM), the fundamental model from which all HL7 V3 messages are derived. To avoid overloading the RIM with a huge number of classes, HL7 defines generic healthcare information classes at the RIM level. The RIM classes can be refined into more specific classes for a particular domain. The refinement process leads to the development of a huge number of classes for different domains. These classes may not share common features.

Many HL7 V3 integration profiles have been developed from the need to formalize the messages, so that applications from a specific domain can better interoperate. Even though some of these profiles have been standardized, there are still many companies that, for different reasons, do not adhere to these standards when developing eHealth products. For this reasons, finding repositories of similar messages is complicated.

Considering the difficulties of finding such repositories of specific type of HL7 V3 messages that can be used as input test data, the present work proposes a different approach. Test designers directly generate clusters of equivalent messages, instead of collecting similar HL7 V3 messages that adhere to the same message structure in a test data repository and applying clustering algorithms to group the messages.

Section 5.6 proposes a new approach for a test data generator tool that addresses the issue of creating input data for testing eHealth applications. Test designers can use this test data generator

for creating custom types of input data. It can be used to generate messages that adhere to any type of HL7 V3 compliant structure. The generator enables the possibility of choosing the target testing language and its correspondent test system, adapting the format of the generated test data to the language dependent format.

One of the most important characteristics of this generator is that it offers the possibility to create clusters of test data. Due to the high level of customization it allows, specific messages can be created as part of a certain cluster. Generating clusters of HL7 V3 messages is an important feature, useful in many testing contexts.

5.6 Architectural design of the input test data generator – the first HL7 V3 message generator

As presented in the previous section, it is difficult to find or create a suitable set of input test data for testing HL7 V3-based applications that operate in different healthcare domains and structure their messages differently. This section proposes an architectural design for a test data generation tool that can be used to generate sets of HL7 V3 messages in different testing language formats [EGN13b].

5.6.1 Requirements

The following requirements were taken into account when designing the test data generator:

R1. *The generator enables any type of HL7 V3 message generation, regardless of the message representation format required by the SUT*

The test data generator should provide methods for creating input test data suitable for any HL7 V3 interface. The generating tool should not depend on a specific message representation format, or on the particularities of a specific HL7 V3 integration profile.

R2. *The generator provides methods for engendering test data that do not depend on a specific testing environment*

The generator is designed for creating input test data that can be used within different test systems. The generated message format should coincide with the testing language specific format. To achieve this, there should be a clear separation between the process of generating HL7 V3 messages in a base representation format, which is language independent, and the adaptation of the generated messages from the base format to a language specific format. This differentiation between the two components shows its advantages when testing team often change the testing technologies. The generated messages can be reused in a new testing environment if a simple format converter is implemented.

R3. *A distance-based generation model is used*

A distance is a numerical descriptor used for measuring the similarity between two objects. Distances can be used in generating processes to ensure that the inputs and outputs of such processes have a specific degree of similarity. The testing team can customize a distance-based generator to generate outputs that satisfy specific requirements.

R4. *The generated HL7 V3 messages can be grouped to form equivalence partitions (helpful for Equivalence Class Testing)*

One of the advantages of using a distance-based generator is that the test designer can generate input for different testing methods. For example, in Equivalence Class Testing the whole input domain is divided in equivalence partitions and one or more representatives are chosen from each partition as testing stimuli. This method significantly reduces the number of test cases. A test data generator that satisfies R3 can be used to generate the input domain in such way that is already partitioned in equivalence classes.

R5. *The generator is highly customizable*

The test data generator should provide several levels of customization regarding:

- the configuration of the distances used in the generating process;
- the selection of the testing language format converters;
- the operating modes – single message vs. batch process generation;
- the interface – GUI for single message generation vs. command-line scripts for multiple message generation.

R6. *The generator automates the testing process*

The test data generator should optimize the creation process of the test data. Generating messages using this tool should decrease costs by eliminating human errors and reducing the time required for defining input test data. The generator should be easy to configure, and should offer enough flexibility to the test designer in terms of the intended output. Moreover, it is required to have a small learning curve.

5.6.2 Generation methodology

The test data generator uses an existing HL7 V3 message, i.e. an instance of a HL7 type, as input and generates messages that have a custom level of similarity with the original one. The input of the test data generator can be a message that adheres to a HL7 V3 integration profile, or one that has a custom defined structure based on specific RIM refined classes.

The generation process follows a distance-based approach. Distances are used for creating messages with custom levels of similarity. Distances are not used on the message-level. Instead, they produce localized effect, at the field-level. Every field of the message is assessed.

Depending on its data type, a specific type of distance is chosen and used for generating the new value. This methodology allows test designers to define the most significant fields whose values should vary.

The generating process can be conceptually divided in two operations:

- the *raw test data generation* is used for creating HL7 V3 messages in a generic, simple format named *base format*, which is not designed for a specific testing technology; the *base format* chosen is Extensible Markup Language (XML); the generated HL7 V3 messages that adhere to the XML format are grouped under the name *raw test data;*
- the *test data conversion* is used for translating the *raw test data* from the *base format* to the specific format required by the testing solution; this adaptation is performed by a set of *mapping rules*, as described in Figure 5.9.

This differentiation between the two operations implies the existence of an intermediate layer, which consists of a set of modules that convert the *raw test data* to specific testing language formats. However, the advantage of having a flexible and reusable solution often overcomes the costs of this approach.

5.6.3 Components of the test data generator

The *Test Data Generator* (TDG) consists of two components:

- Raw Test Data Generator (RTDG);
- Test Data Converter (TDC).

Figure 5.9 depicts the overview of the TDG's architectural design. The two layers are designed to allow test designers flexibility when choosing the target testing language and its correspondent test system.

The *Raw Test Data Generator* (RTDG) represents the first level of this architecture. It is the main component of the TDG and it is responsible for generating *raw test data*. It provides two levels of customization that enable test designers to build specific types of messages. RTDG can be used for generating single messages, or as a batch process, in case large repositories of test data are needed.

The second level of the architecture consists of the *Test Data Converter* (TDC). TDC is composed of several independent modules whose responsibility is to convert *raw test data* from XML format to any test language specific format. This level represents the link between the generated *raw test data* and the test languages used by different test systems.

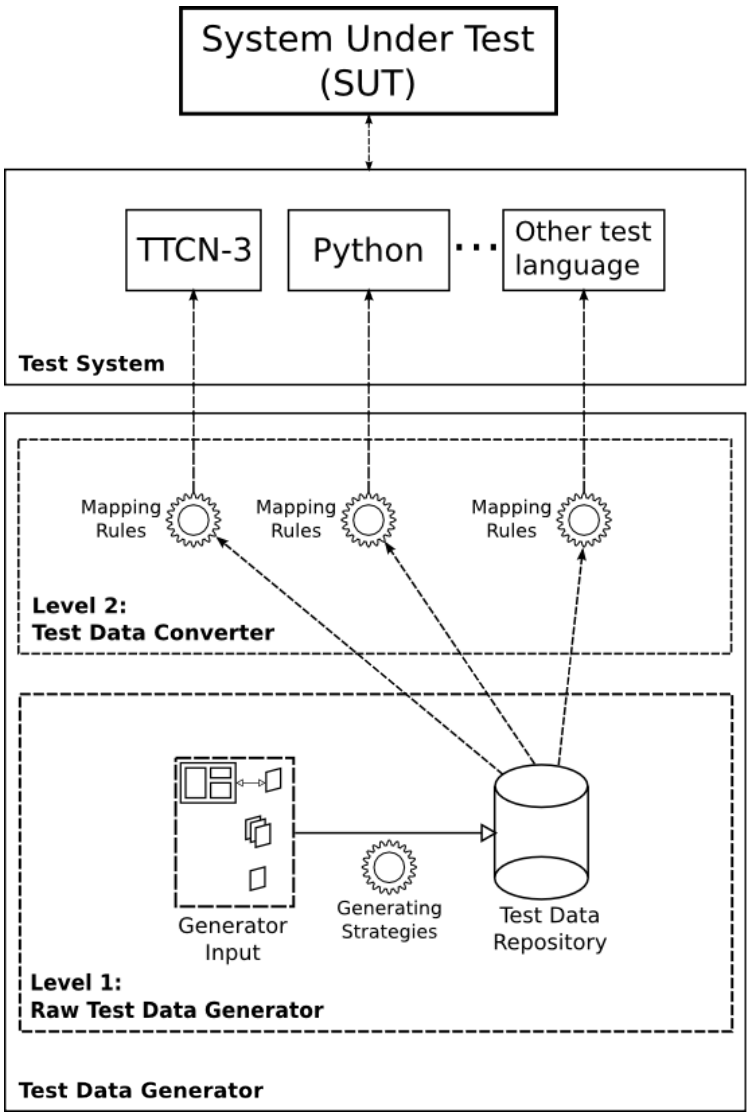

Figure 5.9. Architectural overview of the test data generator

5.6.3.1 The Raw Test Data Generator (RTDG)

RTDG uses three input files in the generation process, as shown in Figure 5.10:

- a valid XML file that contains a compliant HL7 V3 message;
- an XSD file, i.e. the XML Schema, which defines the HL7 V3 data types and messages; the XML file containing the base message should be validated against this XML Schema;
- a configuration file that contains a set of rules and constrains (such as the use of mandatory field values or semantic contexts) that control the generation process.

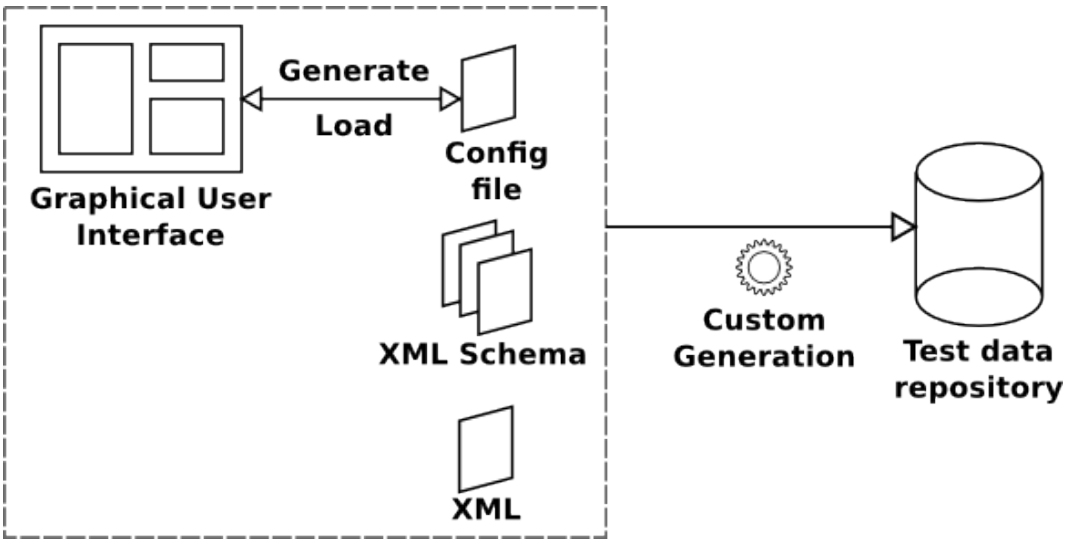

Figure 5.10. Overview of the Raw Test Data Generator

The output of RTDG is another XML file that contains the new HL7 V3 message. The new message is generated by applying changes to the base HL7 V3 message.

There are two types of changes: changes that affect local message values (*low-level generation*) and changes that affect the entire message (*high-level generation*).

Low-level generation

The modules responsible for changing the local values, named *low-level generators*, are the cores of the RTDG. There are two types of *low-level generators*.

1. Generators that create new values independently of the ones contained in the original message

There are situations where the process of generating a new value depends only on the data type and not on the original value. This is the case for data types such as Boolean, or enumerated. The *low-level generators* that fall in this category cannot be customized. They are used for creating values for clearly defined, constrained or fixed data types. A consequence of the generation process in this case is that all the possible values of a data type have uniformly distributed probabilities.

2. Generators that apply a set of transformations to the values contained in the original message

The *low-level generators* that fall into this category are named *distance-based generators*, because they generate new values using several types of distances. The distances used in the generation process can be customized to determine a particular level of similarity between values. For example, strings with no constraints defined in the input XML Schema file are translated into new strings of customizable Levenshtein distance [YUJ07] and integers are

translated into new integers of customizable Euclidean distance. The *distance-based generators* are the only *low-level generators* that can be customized.

Both types of generators create new values according to a field's data type. Although they behave differently, their main scope is to generate field values, not whole messages. This is the reason why the process they are involved in is called a *low-level generation*. The *low-level customization* refers to the ability to customize the distances used by the *distance-based generators*.

The customization consists of the possibility of adjusting distances used in the generation process. Each distance can be adjusted independently. Lower values should be assigned to the *distance-based generators* to obtain outputs that are similar to the inputs. By varying the distance below a fixed threshold, test designers can generate messages that are part of the same cluster. Higher values assigned to the *distance-based generators* determine significant differences between input and output messages. The test designer may want to use high values for generating messages belonging to different clusters.

Figure 5.11 shows how new messages are generated by the *low-level generators*. The three examples chosen represent the three most used *low-level generators*:

- the first two are *distance-based generators*; the distances they use can be customized according to the intended output message;
- the third one, which handles strings with constraints, is not customizable and generates new values independently of the original ones.

Figure 5.11 also shows that the original message structure is preserved.

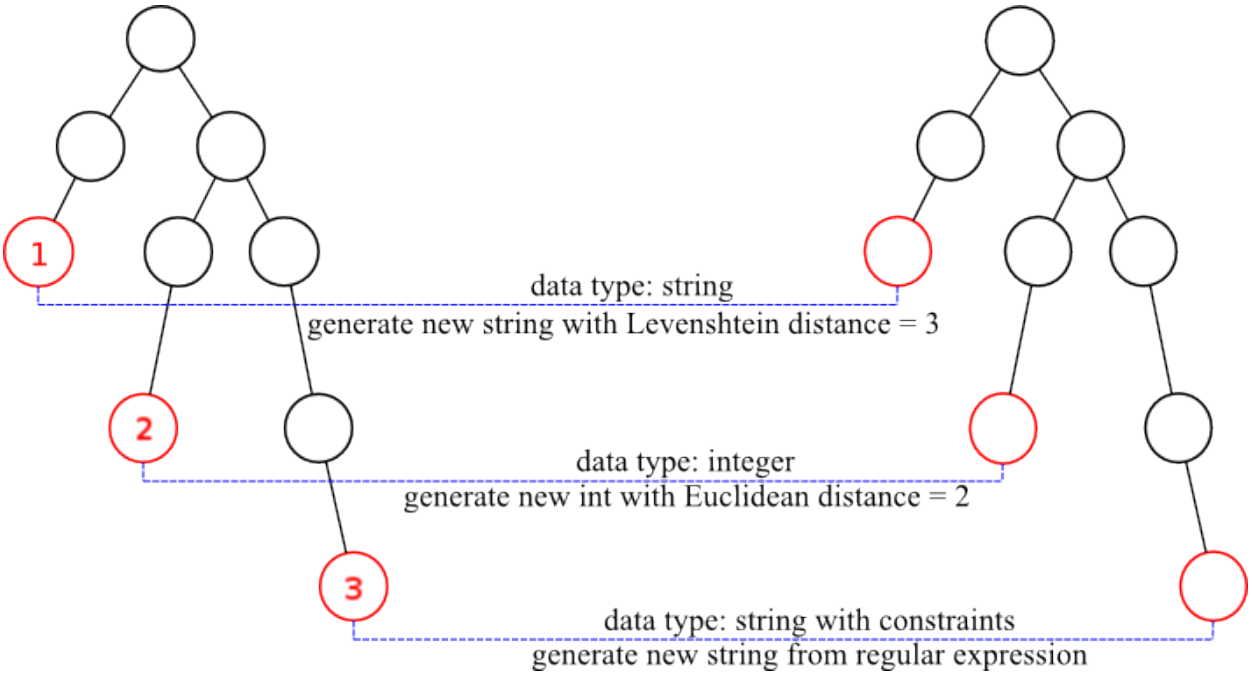

Figure 5.11. Low-level generators

High-level generation

The *distance-based generators* are powerful tools for generating messages with localized differences. RTDG provides test designers methods for customizing entire messages, as well. This process is called *high-level customization*.

There are various situations where the *high-level customization* is useful, as for instance:

- defining supplementary constraints for certain fields, in addition to the ones already defined in the XML Schema; a plain string with no constraints defined in the XML Schema file, for example, can be restricted to match a particular regular expression; in the same manner, the possible values of an integer can be restricted to a predefined set;
- adjusting the default behavior of the *low-level generators* in different scenarios; the distance used by a *distance-based generator,* for instance, can be changed for a particular field;
- obstructing the default behavior of the *low-level customization* for specific fields of a message; the obstruction of a *low-level generator* is equivalent to its exclusion from the generation process.

It can be considered that the *high-level customization* overrules *low-level customizations*, modifying the default behavior of the *low-level generators*.

Figure 5.12 presents the general architecture of the Raw Test Data Generator, highlighting the two types of customization and the way they influence the test data generating process.

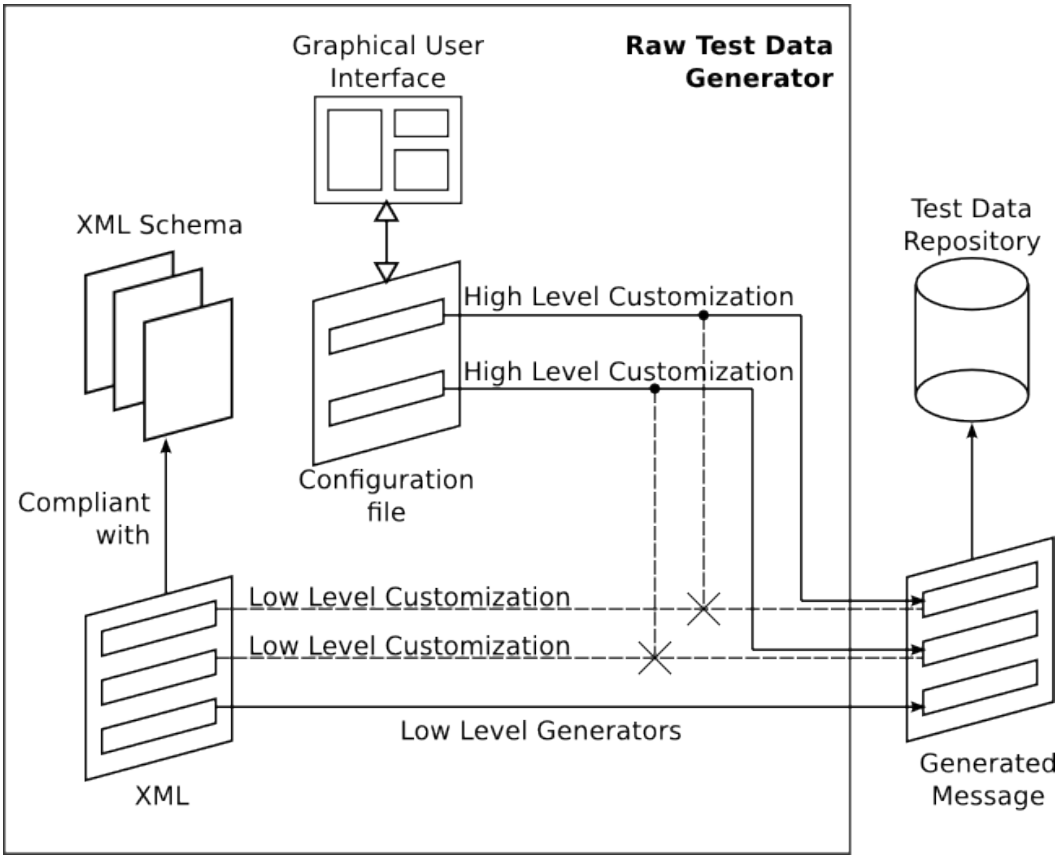

Figure 5.12. Architectural design of the Raw Test Data Generator

As presented in Figure 5.11, the generation process consists of modifying the values of an existing message according to certain rules. These rules are defined on two different levels: low-level rules and high-level rules.

The low-level rules can be differentiated in two categories:

- the first category comprises the rules that cannot be customized or overruled; the non-customizable *low-level generators* use these rules; this type of rules is represented as the lowest of the three lines of the *low-level generators* in Figure 5.12;
- the second category of rules aggregates the rules that can be customized and overridden; the customizable *low-level generators*, i.e. the *distance-based generators*, use these rules; they are shown in the upper two lines of the *low-level generators* in Figure 5.12.

The high-level rules include rules that can be defined in the configuration file, or with the help of the Graphical User Interface (GUI). The GUI is designed to provide visual representation of the configuration file. The rules defined in the GUI are automatically synchronized with the ones existing in the configuration file. These rules override the ones governing the low level customization.

5.6.3.2 The Test Data Converter (TDC)

TDC is an aggregation of several converter modules that translate the *raw test data* from the *base format*, i.e. XML, to the test system specific format. These converters depend on the structure definitions of the data types in various testing languages. It is difficult to design a generic architecture for this component, because of the differences between the possible target languages.

Section 5.7.2 describes in detail two examples of TTCN-3 converters and provides several implementation guidelines. The implementations show that many approaches can be considered when developing the converter. One of the approaches shows how specific characteristics of the target language can be used as advantages in the conversion process.

5.7 Technical realization of the Test Data Generator

This section describes the methodology used for implementing the *Test Data Generator* (TDG). TDG consists of two components: the *Raw Test Data Generator* and the *Test Data Converter*. Their implementation, together with the employed tools and applied technologies and algorithms are detailed in this section. Guidelines for different implementations are also presented.

5.7.1 Level 1: the *Raw Test Data Generator* (RTDG)

RTDG is responsible for generating custom HL7 V3 messages in a simple, generic format. Ensuring the generated test data's independence of the testing system provides test designers with a great level of flexibility, but leads to the addition of an intermediate layer responsible for adapting input data to the test system. This adaptation, however, is not difficult to realize.

RTDG provides two ways for customizing the generation process. The first one applies to the message's field values. This type of customization is named *low-level customization* because it does not provide means to customize the message as a whole. Instead, it only affects small parts of it. As opposed, the other type of customization, named *high-level customization*, enables the generation of certain type of messages, by constraining them to match a specific message template.

5.7.1.1 Low-level customizations

This section analyzes the *low-level generation* process and the means of customizing it using the *low-level customizations*. The procedure of generating new messages using *low-level generators* is described in Algorithm 2 presented below.

The first step is parsing the input XML file representing the base HL7 V3 message and converting it into a document, using Document Object Model (DOM). DOM is an interface that allows dynamic access and update of the content, structure and style of a document [WOO00].

DOM was preferred because it provides an in-memory tree structure of the XML document, which can be easily updated.

The resulting tree is recursively parsed and for each leaf node a list of ancestors is saved. An XML Path Language (XPath) expression is created for the list of ancestors and used for identifying the corresponding node in the XML Schema [CLA99]. The data type is then extracted from the node identified in the XSD. Depending on the data type, certain decisions are made and the leaf is given a new value. After all leaves have been assigned new values, the modified DOM document, which represents the newly generated HL7 V3 message, is serialized.

Algorithm 2. Distance-based generation of HL7 V3 messages

Inputs
- *hl7 V3_in* – the original HL7 V3 message (XML format)
- *xml_schema* – the XSD that validates *hl7 V3_in*

Outputs
- *hl7 V3_out* – the new HL7 V3 message (XML format)

```
1.  tree ← parse hl7 V3_in and obtain DOM tree
2.  root ← get the first element of tree
3.  new_doc ← modify (root)
4.  hl7 V3_out ← serialize new_doc
5.
6.  // other function definitions
7.  function modify (root)
8.      for each child in root.children do
9.          if child is leaf then
10.             ancestors ← get list of child's ancestors
11.             query ← create XPath expression using ancestors
12.             xsd_node ← identify node in xml_schema using the query XPath expression
13.             xsd_node_type ← extract data type for xsd_node from xml_schema
14.             child.value ← calculate a new value based on the old child.value
15.         else
16.             modify (child)
17.         end if
18.     end for
19. end function
```

The decisions made when assigning new values to the leaf nodes are an important aspect of the generation algorithm. A comparison between snippets from the original message and the generated one is shown in Figure 5.13. This example illustrates how *low-level generators* are used for creating new values. This figure shows different types of situations encountered when generating these values.

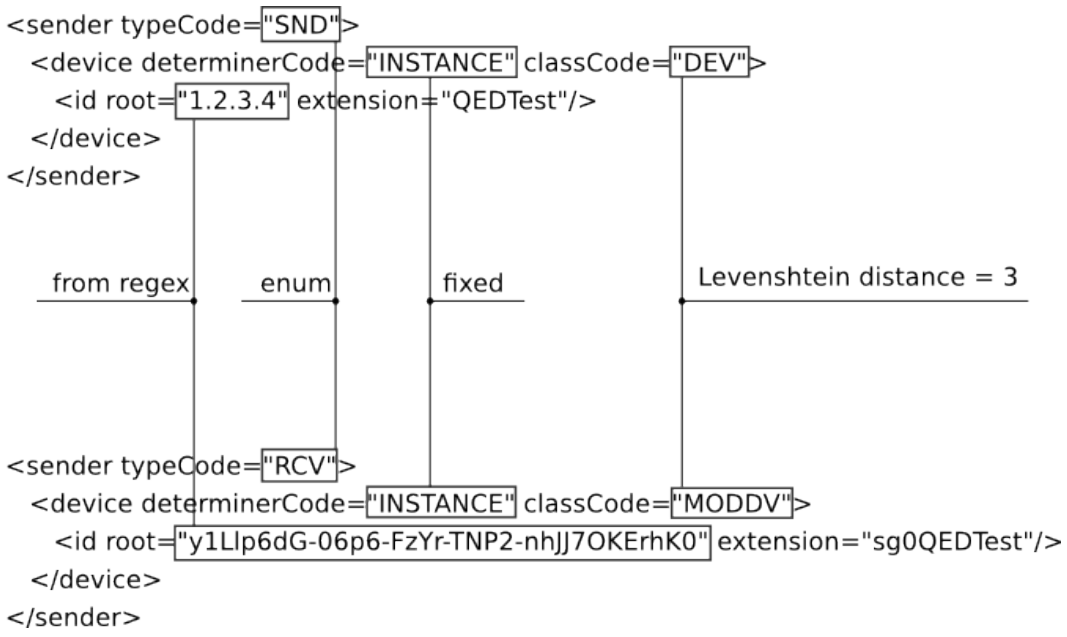

Figure 5.13. Example of values generated with the low-level generators

This example illustrates three common situations that appear during the generation process.

1. The first case is represented by the fixed values, exemplified by the value *INSTANCE* of the *determinerCode*. The generation process cannot modify the fixed values defined in the XML Schema.
2. The second case presents values such as Booleans, enumerations or strings that match a specific regular expression. These are not fixed values, but they are constrained by the XSD. The non-customizable *low-level generators* handle these values.

For example, a list of possible values is created according to the constraints specified in the XML Schema file for the enumerated type. A value is then chosen from that list without taking into account the original one. This way the generator ensures that the possible values have uniformly distributed probability of appearance.

For strings with constraints defined in the XSD, an external Java library, Xeger [SPR09] is used. This library enables generation of strings that match a particular regular expression. There are many libraries and tools available for string generation from which the implementer can choose. Both of these two examples show how to generate values taking into account various constraints defined in the XML Schema. It is important to mention again that these types of generators are not customizable.

3. The third case presented in Figure 5.13 illustrates values that are not constrained. The *distance-based generators* handle these values. These are the only situations where new values are generated based on the original ones. The example shows how a *distance-*

based generator for strings, which uses the Levenshtein distance, generates new values. In this example, the Levenshtein distance was set to 3 and the value of the *classCode* becomes *MODDV*.

RTDG was designed taking into account requirement R4 from Section 5.6.1, i.e. the generator should be able to generate groups of similar messages, named clusters. It is the test designer to define the clusters and the thresholds of these clusters.

One of the fundamental concepts used in clustering theory is the distance. Distances are metrics used for determining the similarity of two values. In the clustering context, distances can be used for validating the membership of a value to a certain cluster. The same concept is applied to the TDG. There are several different distances used by the TDG for customizing the generation process. By gradually changing the distances, messages with various levels of similarity can be created.

A test designer can profit from the advantages of the *low-level customization* to shape the output of the generation level. For instance, the test designer can opt to generate messages grouped in clusters called *equivalence partitions*. This is useful when following a testing approach based on Equivalence Class Testing. Before generating the clusters the test designer should establish the cluster boundaries. He can define a distance threshold so that every message generated based on a distance lower than the fixed threshold is part of the same cluster. Distances greater than the threshold can be used for generating other message clusters.

There are several distances that can be used in the *low-level customization*. The test designer can set a correspondence between data types and distances. For instance, integers can be mapped to Euclidean distance and strings can be mapped to Levenshtein distance. However, custom distances can also be defined. For instance, new distances can be defined for evaluating dates of birth, or addresses.

Two of the distances that are used for basic data types and can be adjusted by the *low-level customization* are Levenshtein distance and Euclidean distance.

The Levenshtein distance [GUS97], [YUJ07] is a metric for measuring the amount of differences between two sequences. It is also called edit distance. In terms of strings, the Levenshtein distance is defined as the minimum number of edits needed to transform a string into another one. As opposed to the Hamming distance, the Levenshtein distance can also be used for measuring similarity between strings of different lengths.

Levenshtein distance defines three types of string operations: substitution, insertion and deletion of a character. The cost of each operation is described in the following formulas:

- $Sub(a, a) = 0, \forall a \epsilon \Sigma$; Σ is defined as the set of all characters;
- $Sub(a, b) = 1, \forall a, b \epsilon \Sigma \; with \; a \neq b;$
- $Del(a) = 1, \forall a \epsilon \Sigma$;
- $Ins(a) = 1, \forall a \epsilon \Sigma$.

Damerau defines a new operation in addition to the ones defined by Levenshtein: transportation of two adjacent characters. The Damerau-Levenshtein distance is stated to correct 80% of the usual misspellings [GUS97]. In our implementation, Levenshtein distance is the default distance used by the *low-level generators* for strings.

The Euclidean distance is a metric for measuring the distance between two different points [GOW82]. The distance between points $X = (x_1, x_2 \dots x_n)$ and $Y = (y_1, y_2 \dots y_n)$ belonging to the Euclidean space $\mathbb{R}^n$ is given by $d = |X - Y| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

In terms of integer values, the Euclidean distance means the absolute value of the subtraction of the two numbers: $d = |X - Y|, \forall X, Y \in \mathbb{Z}$. In the present implementation, Euclidean distance is the default distance used by the *low-level generators* for integers.

In addition to the distances that are used for basic data types, the test designer can define new custom distances. Let's consider the case of the HL7 V3 data type TS. The timestamp data type can be used in applications to store birthdates, for instance. The test designer can define a custom distance that expresses time differences. The test designer may consider measuring the difference in days, months, or years. A distance of 1 may be assigned to 10 days difference, or two months difference, or three years difference. Patients' birthdates may be grouped, using these distances, based on the intended criterion.

Other example of HL7 V3 data type is AD, which is used for storing addresses of patients. A distance can be defined here to evaluate the location differences. In this implementation, the following were considered:

- $d = 0$, for patients who live in the same city;
- $d = 1$, for people that live in the same county, but not in the same city;
- $d = 2$, for people that live in the same geographical area that spans on several counties, but not in the same county;
- $d = 3$, for people that live in the same country, but not in the same geographical area.

The distance, as defined here, is usable in a country where this geographical separation exists. However, as shown here, distances are easy to define for simple HL7 V3 data types and can be easily adapted by test designers to ensure localization.

When defining distances for complex HL7 V3 data types, the test designer has to understand and define the semantic context. It is difficult to offer guidelines for defining distances for complex HL7 V3 data types, because it depends on what the test designer aims to test when defining the messages. In addition, these types of distances are rarely used. It's more practical to define HL7 V3 message templates then to define complex distances and generate messages based on these distances.

5.7.1.2 High-level customizations

The capability of TRDG to create test data repositories of HL7 V3 messages with different structures depends on the level of customization provided by the generation process. The *low-level customization* provides means of personalizing the *distance-based generators*. This section reveals another layer of components that enables further customization, i.e. the *high-level customization*.

As shown in Figure 5.12, there are two components that handle the *high-level customization*, which are tightly linked: the configuration file and the Graphical User Interface (GUI).

The *high-level customizations* are defined in the configuration file. As opposed to the *low-level customizations*, they define generation rules that affect entire messages. The configuration file allows test designers to create specific message templates suitable for particular test objectives. The configuration file consists of a set of constraints that apply to fields of the message in the course of the generating process. Several types of constraints that can be defined in the configuration file have been identified, such as:

- fixed values for particular fields within the message;
- values that are part of a set;
- values that match a certain regular expression;
- values that are system dependent.

The constraints defined in the configuration file override the ones used in the *low-level customization*. If no constraint is defined in the configuration file for a specific message field, its value is generated with the default behavior of the *low-level generators*. The *high-level customization* is useful when test designers want to define clusters of messages. A snippet of how the configuration file may look like is shown in Figure 5.14.

```
<extension>
    <ancestors value="QUPC_IN043100UV01 receiver device id"/>
    <pattern value="[a-zA-Z][a-zA-Z0-9-_]+"/>
</extension>
```

Figure 5.14. Example of string constraint definition: regular expression

Figure 5.14 shows an example of string constraint definition in the configuration file. The example shows the constraints defined for node *id*. Several nodes with the name *id* may exist in the original XML file. A list of ancestors of that node is used for creating the XPath query that uniquely identifies the node. This list contains all the nodes in the tree path, starting from the root node. The next line shows the constraint definition. In this case, the constraint is a regular expression that the new value must match. The new value of the node *id* must start with a letter and continue with one or more letters, digits, hyphens, or underscores.

```
<extension>
    <ancestors value="QUPC_IN043100UV01 receiver device id"/>
    <original value="qed"/>
</extension>
```

Figure 5.15. Example of string constraint definition: original value

Figure 5.15 shows another example of string constraint definition. In this case, the original value is defined. In terms of the configuration file, the original value has no relationship with the node's value from the XML. The original value defined in the configuration file provides another starting point to the *distance-based generators*. It is used, in other words, for overwriting the value used as input by the *distance-based generators*. The original value defined in the configuration file affects the generating process only when the default behavior of the *distance-based generators* is maintained, i.e. a specific Levenshtein distance is used for generating a new string considering the initial value.

The GUI component is the visual equivalent of the configuration file. It provides two functionalities:

- visualization – the GUI can be used for displaying HL7 V3 messages and information about the constraints defined in the XML Schema file;
- capability of defining high-level constraints similar to the ones defined in the configuration file.

In terms of generating test data, the configuration file has a similar role to the GUI. Their objective is the same, but they differ mainly in one important aspect. The GUI is useful when test designers prefer defining restrictions in a visual way while viewing the structure of the input test data. The drawback is that the GUI handles a single *high-level* configuration at a time. Instead, in a batch generation process the test designer can use interchangeable configuration files, which leads to the definition of a larger and more diversified input test data set.

For test designers who need both, implementation of other components, such as converters between the two formats, or GUI generators based on the configuration file, the XML and the XML Schema, may prove useful.

5.7.2 Level 2: the *Test Data Converter* (TDC)

The second layer of TDG is the *Test Data Converter*, i.e. the logical layer above RTDG, as shown in Figure 5.9. This component is responsible for linking the *raw test data* with test systems. The target testing language chosen for analysis is TTCN-3. Two different approaches were implemented and compared:

- TDC specifically designed for a HL7 V3 integration profile;
- TDC based on TTCN-3 test system components.

5.7.2.1 Stand-alone TDC

This implementation of TDC is independent of the HL7 V3 integration profile chosen and requires no external libraries, or configuration files. It is a stand-alone application, which provides a simple GUI that interfaces the transition between the XML format, i.e. the *base format* of the generated HL7 V3 messages, and the TTCN-3 template format. Its purpose is to provide means for the test data to be used in the TTCN-3 environment. It is not meant for complete TTCN-3 test system solutions. It is helpful for the test designers who define the Abstract Test Specification.

The conversion of the test data from XML to TTCN-3 is not a straightforward process. Since TTCN-3 has no reflection mechanism, TTCN-3 templates' fields cannot be instantiated and assigned values at runtime. For this reason, the information stored in the XML is extracted, converted into a string, formatted as a TTCN-3 template and then written in a new TTCN-3 module.

The process of converting test data between the two formats follows the steps below. A detailed algorithm is presented in Algorithm 3.

Step 1. The XML file containing the HL7 V3 message is loaded
Step 2. The HL7 V3 message is validated
Step 3. The XML is parsed and converted into a DOM document
Step 4. The document tree is parsed and each node is interpreted. Information is extracted from the node's attributes, text value and children
Step 5. The information is serialized in the TTCN-3 template format
Step 6. The final output is a TTCN-3 file containing all the information that was stored in the XML file.

Algorithm 3: Test data conversion from base format to TTCN-3 template

Inputs
- *hl7 V3_msg_def* – XML file containing the HL7 V3 message definition
- *xml_schema* – the XSD that validates *hl7 V3_msg_def*

Outputs
- *ttcn3_template* – file containing the TTCN-3 template

1. *str* ← empty string
2. *ok* ← validate *hl7 V3_msg_def* against *xml_schema*
3. **if** *ok* **then**
4. *tree* ← parse *hl7 V3_msg_def* and obtain the DOM tree
5. *root_xml* ← get the first element of *tree*
6. *obj_type* ← locate class in *classpath* with the name *root_xml.name*
7. *obj_instance* ← instantiate *obj_type*
8. *str* ← *convert* (*root_xml*, *obj_instance*)
9. *serialize*(*str*, *ttcn3_template*)

```
10. end if
11.
12. function convert (root, obj)
13.     str_attr ← convertAttributes(root, obj)
14.     update str with str_attr
15.     update str with root.textValue
16.     for each child in root.children do
17.         child_type ← parse obj's fields and extract child's data type
18.         child_instance ← instantiate class_type
19.         convert (child, child_instance)
20. end function
21.
22. function convertAttributes(root, obj)
23.     str_attr ← empty string
24.     if root has attributes then
25.         for each attribute in root.attributeList do
26.             update str_attr with attribute.name and attribute.value
27.         end for
28.     end if
29.     return str_attr
30. end function
```

The implementation of this TDC concluded with several observations.

On the one hand, any change in the environment, such as the substitution of the target testing language, determines the implementation of a completely new TDC. This approach is not recommended when constant changes are expected.

On the other hand, the implementation of this TDC is not difficult. It does not require advanced knowledge of the TTCN-3 test system architecture, or ample experience handling the TTCN-3 type system. Moreover, the implementation of this type of TDC for a different scripting language may prove even easier. Generic scripting languages, such as Python, or specialized testing languages, such as Tool Command Language (Tcl) [OUS89], provide data manipulation during runtime. The serialization of the TTCN-3 template would be straightforward in these cases.

5.7.2.2 TDC based on TTCN-3 test system components

This section describes a different implementation approach for converting HL7 V3 messages from the *base format* to the TTCN-3 format. This implementation is specifically designed for TTCN-3 and relies on the TTCN-3 test system components. This solution requires that the test designer is accustomed with the architecture of the TTCN-3 test system and its components. Figure 5.16 shows the architecture of a TTCN-3 test system.

The Encoder/Decoder component, i.e. the Codec, is the TTCN-3 test system component that interfaces the TTCN-3 type system with the TTCN-3 test system. A generic Codec defines a set of operations that convert TTCN-3 data types into high-level abstract types. Depending on the Codec implementation, the high-level abstract types are mapped to a specific programming language, such as ANSI C, or Java. This implementation is based on a Codec that uses Java mapping rules. Therefore, this Codec has two functionalities:

Step 1. Encoding data, i.e. the process of converting TTCN-3 values into Java objects
Step 2. Decoding data, i.e. the process of converting Java objects into TTCN-3 values

The objective of this TDC is the data conversion from the *base format* into TTCN-3. This conversion is performed with the help of the Codec's decoding functionality.

There are tools, such as TTworkbench [TTW], which can be used for automatic generation of Codecs based on XML Schemas.

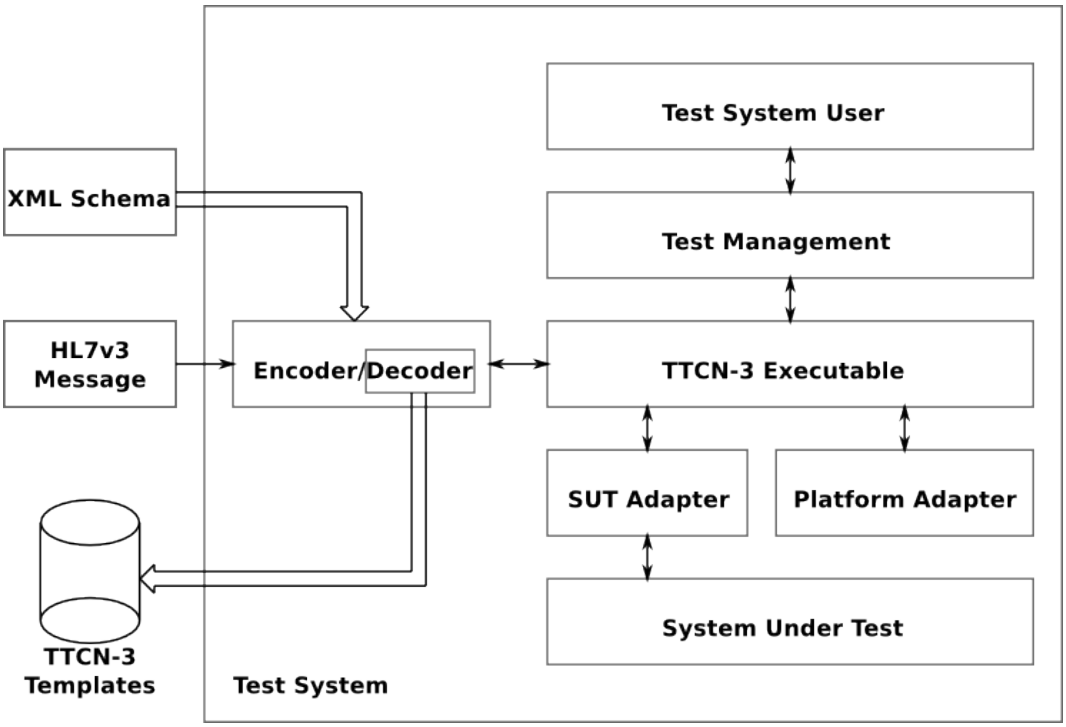

Figure 5.16. TTCN-3 template generator

The process of converting the generated *raw test data* into TTCN-3 templates follows the next steps:

Step 1. Validate the HL7 V3 message against the XML Schema
Step 2. Parse the XML file that defines the HL7 V3 message
Step 3. Convert the HL7 V3 message to an intermediate Java object

Step 4. Call the Codec's *decode* method with the intermediate Java object
Step 5. Serialize the resulting TTCN-3 template in a TTCN-3 module

With this implementation approach, the test designer should be familiar with the architecture of the TTCN-3 test system. A set of Java classes that define the HL7 V3 data types and messages is required and the test designer should be able to implement a converter between XML and Java.

On the other hand, this approach is more adequate for testing environments based on TTCN-3. This TDC can be easily integrated in a testing solution based on the TTCN-3 test system.

5.7.2.3 Comparison between TDCs

The proof-of-concepts shown illustrate two different approaches for converting the generated *raw test data* into TTCN-3 templates. The two methods show that the conversion process is not difficult to implement and that solutions may be adapted to other testing language, as well.

Table 5.3. Features of different TDC implementations

Feature	Stand-alone TDC	TDC based on the Codec
Adaptable to other HL7 V3 integration profiles	Yes	Yes
Adaptable to other testing languages	Yes	No
Designed as an integrant component of the TTCN-3 test system	No	Yes
Requires knowledge of TTCN-3 test system architecture	No	Yes
Requires external files (libraries/configuration files/schemas)	No	Yes
Requires license	No	Yes

Table 5.3 shows that, in general, a stand-alone TDC is preferable. It is flexible enough and its implementation costs are reduced. The TDC based on the Codec is, however, a better solution when the test system is based on TTCN-3.

5.8 Case study and results

The testing framework and the adjacent generators were implemented and validated in a real environment. The research was part of the Reliability Testing of Medical Systems (ReTeMeS) Project [EUR10]. The methodology was validated on Hospital Manager 3.0 (HM3), a Patient Information Management System developed by one of the consortium partners. HM3 is IHE QED compliant. During the solution validation, HM3 was deployed as a web service, in order to

facilitate remote testing. TTCN-3 allows both message and procedure-based communication. In this case, however, a message-based communication was implemented and analyzed.

Step 1. Generation of HL7 V3 data types in Java

Defining the HL7 V3 data types in Java language was the first task. It was clear from the beginning that their existence was an absolute requirement and that they couldn't be manually defined. The generator described in Section 5.2 was used for creating 1423 Java files containing both basic and complex HL7 V3 definitions. From these definitions:

- **789** were Java Enums, corresponding to TTCN-3 enumerated types;
- **634** were Java classes, corresponding to TTCN-3 records.

The Java Enums definitions didn't need post-processing. For the Java Class definitions a light post-processing tool was designed and implemented in Python. The post-processing procedure involves adding or modifying some statements in the class definition. The scripts that form the post-processing tool, which are described in Section 5.2, don't depend on the HL7 V3 integration profile, as they primarily intend to ensure compatibility with the TTCN-3 test system components and interfaces.

An example of a generated QED Query message in Java format is presented in Figure 5.17. The getter and setter methods are omitted, for readability reasons.

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "QUPC_IN043100UV01.MCCI_MT000100UV01.Message", propOrder = {
    "realmCode",
    "typeId",
    "templateId",
    "id",
    "creationTime",
    "securityText",
    "versionCode",
    "interactionId",
    "profileId",
    "processingCode",
    "processingModeCode",
    "acceptAckCode",
    "sequenceNumber",
    "attachmentText",
    "receiver",
    "respondTo",
    "sender",
    "attentionLine",
    "controlActProcess"
})
@XmlSeeAlso({
    QUPCIN043100UV01 .class
})
public class QUPCIN043100UV01MCCIMT000100UV01Message implements Serializable{

    protected List<CS> realmCode;
    protected II typeId;
    protected List<II> templateId;
    @XmlElement(required = true)
    protected II id;
    @XmlElement(required = true)
    protected TS creationTime;
    protected ST securityText;
    protected CS versionCode;
    @XmlElement(required = true)
    protected II interactionId;
    protected List<II> profileId;
    @XmlElement(required = true)
    protected CS processingCode;
    @XmlElement(required = true)
    protected CS processingModeCode;
    @XmlElement(required = true)
    protected CS acceptAckCode;
    protected INT sequenceNumber;
    protected List<ED> attachmentText;
    @XmlElement(required = true)
    protected List<MCCIMT000100UV01Receiver> receiver;
    @XmlElement(nillable = true)
    protected List<MCCIMT000100UV01RespondTo> respondTo;
    @XmlElement(required = true)
    protected MCCIMT000100UV01Sender sender;
    @XmlElement(nillable = true)
    protected List<MCCIMT000100UV01AttentionLine> attentionLine;
    @XmlElement(required = true)
    protected QUPCIN043100UV01QUQIMT020001UV01ControlActProcess
        controlActProcess;
    @XmlAttribute
    protected List<String> nullFlavor;

    //Getters and setters
```

Figure 5.17. Generated Java class defining a QED Query

Steps 3-5 describe the usage of this message in a practical example.

Step 2. Generation of user-defined TTCN-3 data types

The ReTeMeS project was the perfect environment for validating the user-defined TTCN-3 data type generators presented in Section 5.3. The analysis shows that the use of these generators

considerably reduced the time spent for defining the TTCN-3 data types, allowing test designers to concentrate their efforts more on defining the testing behavior.

There are a couple of numbers that demonstrate the use of these generators:

- **850** TTCN-3 records were generated;
- the TTCN-3 module that contains records definitions has **8500** lines of code;
- **789** TTCN-3 enumerated types were generated;
- the TTCN-3 module that contains enumerated types definitions has **6800** lines of code.

These numbers substantiate the importance of these generators. The generators demonstrate their utility firstly by reducing the test designer's time spent on creating the testing context and secondly by eliminating the possible human errors caused by such laborious definition processes.

The Java class shown in Figure 5.17 is used for generating the TTCN-3 record that defines the QED Query message. The equivalent TTCN-3 record is depicted in Figure 5.18. Test designers can use this record for defining TTCN-3 templates that act as testing inputs.

```
type record QUPCIN043100UV01MCCIMT000100UV01Message {
    record of CS _realmCode optional,
    II _typeId optional,
    record of II _templateId optional,
    II _id optional,
    TS _creationTime optional,
    ST _securityText optional,
    CS _versionCode optional,
    II _interactionId optional,
    record of II _profileId optional,
    CS _processingCode optional,
    CS _processingModeCode optional,
    CS _acceptAckCode optional,
    INT _sequenceNumber optional,
    record of ED _attachmentText optional,
    record of MCCIMT000100UV01Receiver _receiver optional,
    record of MCCIMT000100UV01RespondTo _respondTo optional,
    MCCIMT000100UV01Sender _sender optional,
    record of MCCIMT000100UV01AttentionLine _attentionLine optional,
    QUPCIN043100UV01QUQIMT020001UV01ControlActProcess _controlActProcess
        optional,
    record of charstring _nullFlavor optional
}
```

Figure 5.18. Generated TTCN-3 record defining a QED Query

Step 3. Test case definition

This step describes the process of defining a simple test case. The definition of the first test case is the difficult part. After manually defining the first QED Query, test designers can use the Test Data Generator for generating other input test data. Since the definition of this message is difficult and an incorrect definition is a poor starting point for the generation process, it is recommended that the QED Query be defined in XML format, instead of TTCN-3. The message

can be then translated into the TTCN-3 template format with the help of the TTCN-3 templates editor and generator presented in Section 5.4.

The test case presented here verifies the SUT's behavior when queried about the list of allergies of a patient. The parameter that determines the type of the query is stored in an HL7 V3 CD data type, i.e. *ControlActProcess.ParameterList.careProvisionCode*. In this scenario, the value is set to *INTOLIST*. The target patient is registered in the SUT's database with the id *57f8844f-5904-49c4-8f42-7266859a48f5*. The final TTCN-3 template that resulted from the automatic conversion from XML to TTCN-3 format is shown in Figure 5.19.

xml ttcn-3

```
template QUPCIN043100UV01MCCIMT000100UV01Message T2_Open := {

          _realmCode := -,
          _typeId := -,
          _templateId := -,
          _id := Template_II("1.2.3.4", "abcde", -, -),
          _creationTime := Template_TS("20100520194000"),
          _securityText := -,
          _versionCode := -,
          _interactionId := Template_II("2.16.840.1.113883.5",
"QUPC_IN043100UV", -, -),
          _profileId := -,
          _processingCode := Template_CS("P", -, -, -, -),
          _processingModeCode := Template_CS("T", -, -, -, -),
          _acceptAckCode := Template_CS("AL", -, -, -, -),
          _sequenceNumber := -,
          _attachmentText := -,
          _receiver := {
                    {
                              _typeCode := RCV,
                              _device := {
                                        _determinerCode := "INSTANCE",
                                        _id := {Template_II("1.2.3.4", "qed", -, -)},
                                        _telecom := {Template_TEL}
                              }
                    }
          },
          _respondTo := -,
          _sender := {
                    _typeCode := SND,
                    _device := {
                              _determinerCode := "INSTANCE",
                              _id := {Template_II("1.2.3.4", "QEDTest", -, -)},
                              _name := -
}
          },
          _attentionLine := -,
          _controlActProcess :=
Template_ControlActProcess(Template_ParameterList(Template_CD("INTOLIST",
"2.16.840.1.113883.5.4", "ActCode", -,
-),Template_PatientId(Template_II("InfoWorld","57f8844f-5904-49c4-8f42-7266859a48f5" ,
-, -)))),
          _nullFlavor := -

}
```

Figure 5.19. TTCN-3 template that defines a QED Query

The TTCN-3 template's fields that are not relevant to the query are set to "-". There are fields that are assigned values of other TTCN-3 template. For instance, *receiver.device.telecom* is set to *Template_TEL*, a TTCN-3 template that defines the actual address of the web service provided by the SUT.

The second part of this step represents the definition of the expected response TTCN-3 template. The process is similar to the one followed for defining the input TTCN-3 template. As opposed to the QED Query, the information about the sender or the URL of the web service are in this case irrelevant. To simplify the definition of the expected response, it is sufficient to define a TTCN-3 template for the *QUPCIN043200UV01MFMIMT700712UV01Subject1*. This template is depicted in Figure 5.20.

xml ttcn-3

```
template QUPCIN043200UV01MFMIMT700712UV01Subject1 QueryResponse2 := {
          |_contextConductionInd := ?,
          _registrationEvent := {
                    _statusCode := {_code := "active"},
                    _custodian := ?,
                    _subject2 := {
                              _contextConductionInd := ?,
                              _careProvisionEvent := {
                                        _recordTarget := {
                                                  _name := "recordTarget",
                                                  _value := {
                                                            REPCMT004000UV01RecordTarget := {
                                                                      _contextControlCode := ?,
                                                                      _patient := Template_Patient(
Template_II("InfoWorld",                    "57f8844f-5904-49c4-8f42-7266859a48f5" , -, -),
Template_CD("normal", -, -, -, -), Template_EN("IONESCU", "VASILE"),
Template_CE("M","2.16.840.1.113883.5.1", "AdministrativeGender",-, -),
Template_TS("19771010"))
                                                                                }
                                                            }
                                        },
                                        _pertinentInformation3 := {
                                                  {
                                                            _contextConductionInd := ?,
                                                            _observation := {
                                                                      Template_Observation(Template_II("
allergyID", "2890aa2a-13c7-408c-a8d8-fd1ca733d079",-, -),"OBS",EVN,
                                                                      Template_CD("600", "3.3.3.3.2",-,-,
                                                            "Dermita alergica de contact"),
Template_SourceOf("REFR", "InfoWorld", "5cc914b1-138b-4c6b-9565-f6a048ac346e"), -),
                                                                      Template_Observation(Template_II(
"allergyID", "fcf5dbd8-8343-42d6-a16b-8dbdd85788d3",-, - ), "OBS", EVN, Template_CD("514","3.3.3.3.2", -,
-,"Rinita alergica si vazomotorie"),
Template_SourceOf("REFR", "InfoWorld", "5cc914b1-138b-4c6b-9565-f6a048ac346e"), -)
                                                            }
                                                  }
                                        } // end of pertinentInformation3
                              } // end of careProvisionEvent
                    } // end of subject2
          } // end of registrationEvent
}//end templete 2
```

Figure 5.20. TTCN-3 template that defines a QED Response

The expected observations are defined in the *pertinentInformation3* field. In this scenario, the patient has two types of allergies, which are defined using the *Template_Observation* TTCN-3 template.

Step 4. Establishing the communication between the TTCN-3 test system and the SUT

The Adapter is the TTCN-3 test system component responsible for establishing connections and handling the communication with the SUT. The Adapter is the only TTCN-3 test system component that directly interacts with the SUT. The TTCN-3 test system was designed so that test designers can define test suites independently of the SUT. The same test suite can be

executed on different SUTs with different platforms just by replacing the Adapter component. The Adapter mediates the communication between the TTCN-e Executable (TE) and the SUT. It has two different functionalities: a) the encapsulation of the QED Query and b) the extraction of the QED Response from the Simple Object Access Protocol (SOAP) message received from the SUT.

In the encapsulation phase, the Adapter uses the *TriMessage* received from the Codec as input. The byte array containing the QED Query is deserialized. Given the transparency of the test case to the Adapter, the conversion from Java to XML is performed dynamically, at runtime, using Java Reflection. For this translation Java API for XML Processing (JAXP) was used. JAXP [MOR00] provides the capability of validating and parsing XML documents. It offers several parsing interfaces from which Document Object Model (DOM) parsing interface was chosen. DOM was also used for extracting the Java object from the SOAP-formatted response received from the SUT.

The Adapter is also responsible for handling the communication with the SUT. The HM3's web service interface is defined using Web Service Definition Language (WSDL) [CHI06]. In order to establish the connection of the TTCN-3 test system with the SUT, a WSDL Java client was needed. There are several tools that use the WSDL description of the web service for generating stubs and clients, from which Java API for XML Web Services – Reference Implementation (JAX-WS RI) [JAX12] was chosen.

Step 5. Test case execution and result

This step presents the execution of a test case that failed. Figure 5.21 shows a graphical log that displays the overview of the test case result. Figure 5.22 shows a detailed view of the result. Both views are features of the TTworkbench platform that facilitate the analysis of the received response and the test case verdict.

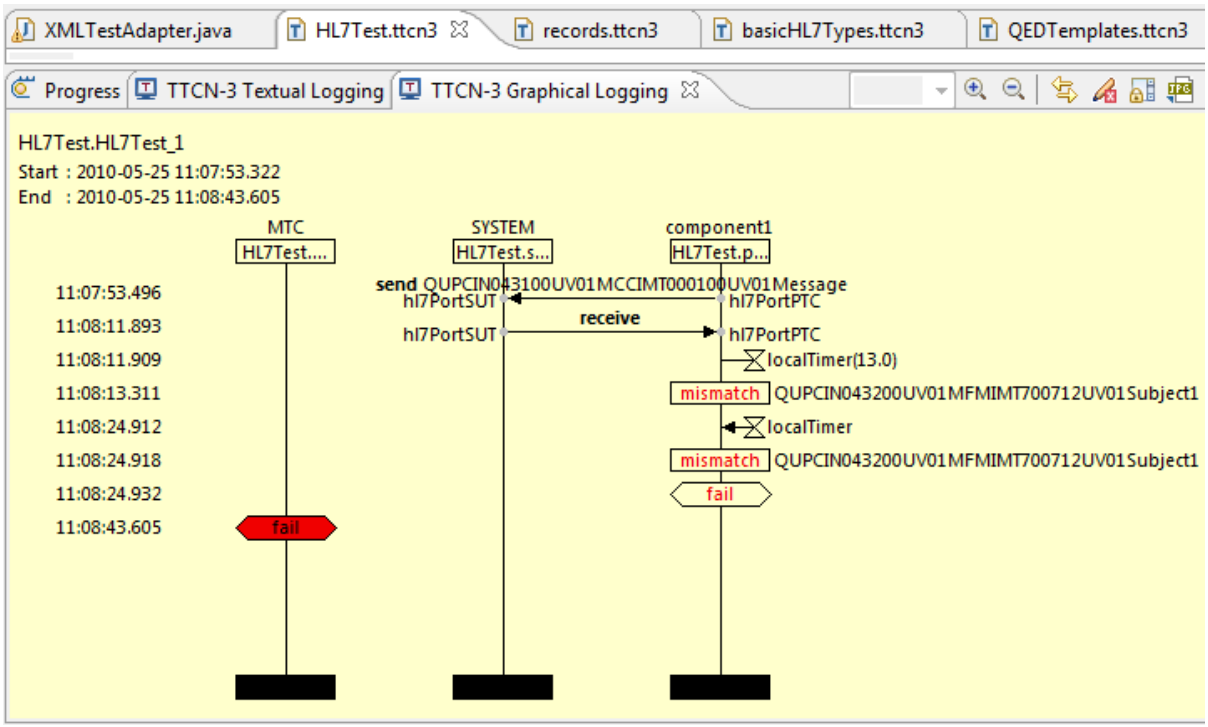

Figure 5.21. Test case result – overview

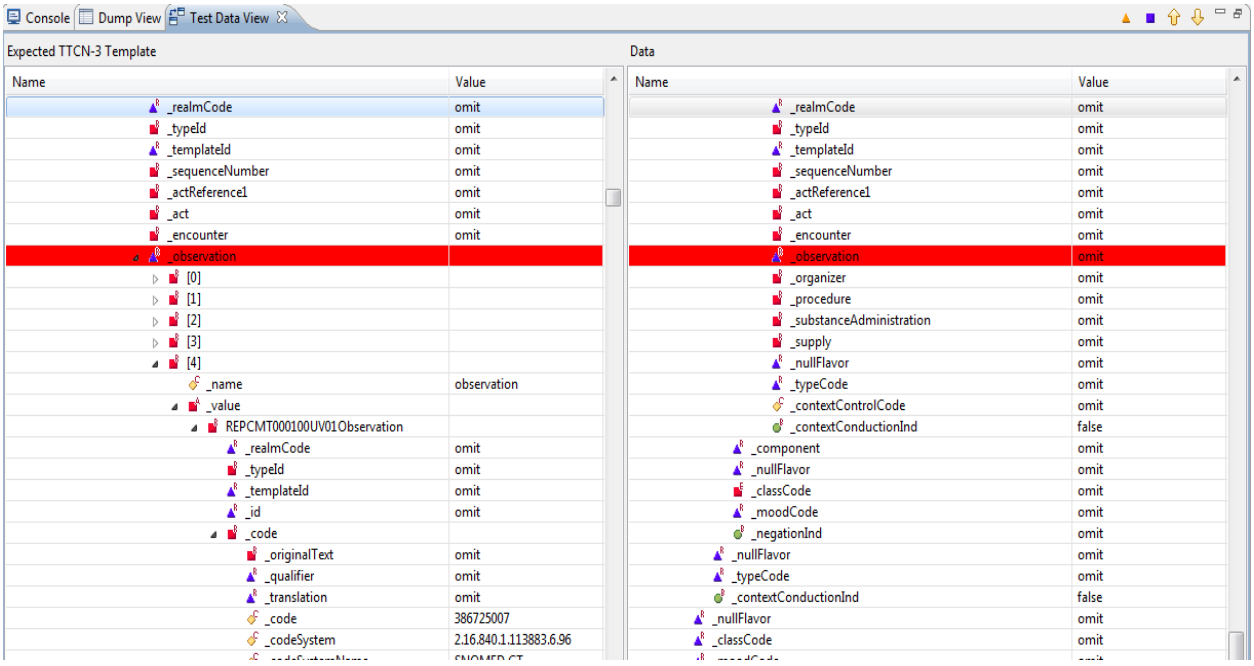

Figure 5.22. Test case result - detailed view

The message displayed in the left panel is the expected response. The actual response is displayed in the right panel. In this scenario, the expected response defines a list of observations that describe vital signs of a patient, such as height, weight, temperature, etc. These expected observations are defined considering the IHE QED specification and the SUT's patient database. The tester can observe that in the actual response received from the SUT the *_observation* value is omitted, which means that the QED Response is not valid. Therefore, the test case is considered failed, as shown in Figure 5.21.

Step 6. Defining input test data for other test cases

The task of defining other input test data is easier, provided that a first reference QED Query is already defined and the tester uses a message generator such as the Test Data Generator described in Sections 5.5 and 5.6. This step shows how to use the Test Data Generator to define a set of test inputs that trigger different responses from the SUT.

The input domain is divided in five subdomains:

- vital signs;
- allergies;
- immunizations;
- problems;
- medications.

Prior to defining the input test data, a reference QED Query message is selected. This message, in this case a QED Query for vital signs defined in a similar way to the one presented in **Step 3**, is used by the Test Data Generator for generating messages that belong either to the same subdomain, or to a different one. *High-level customizations* are used for defining the general structure of the message for each subdomain. For example, a *high-level customization* is used for restricting the value of the *careProvisionCode*, which determines the query type, to a set of predefined values corresponding to the five subdomains.

Figure 5.23 shows a snippet from the reference QED Query that interrogates the SUT about vital signs (*careProvisionCode* value is set to *COBSCAT*). Figure 5.24 shows another snippet from the message generated with TDG, which interrogates the SUT about medications (*careProvisionCode* value is set to *MEDLIST*).

```
<queryByParameter>
  <id root='1.2.3.4' extension='testQuery6'/>
  <statusCode code='new'/>
  <responseModalityCode code='R'/>
  <responsePriorityCode code='I'/>
  <initialQuantity value='1'/>
  <initialQuantityCode code='REPC_RM000100UV' codeSystem='2.16.840.1.113883'/>
  <parameterList>
    <careProvisionCode>
      <value code='COBSCAT' codeSystem='2.16.840.1.113883.5.4' codeSystemName='ActCode'/>
    </careProvisionCode>
```

Figure 5.23. Snippet from the reference IHE QED message (query for vital signs)

```
<queryByParameter>
  <id root='1.2.3.4' extension='testQuery6'/>
  <statusCode code='new'/>
  <responseModalityCode code='R'/>
  <responsePriorityCode code='I'/>
  <initialQuantity value='1'/>
  <initialQuantityCode code='REPC_RM000100UV' codeSystem='2.16.840.1.113883'/>
  <parameterList>
    <careProvisionCode>
      <value code='MEDLIST' codeSystem='2.16.840.1.113883.5.4' codeSystemName='ActCode'/>
    </careProvisionCode>
```

Figure 5.24. Snippet from generated IHE QED message (query for medications list)

After defining the subdomains, the Test Data Generator is used for defining messages in each of these subdomains. *Low-level generators* are used for creating values for patient ids, performers, addresses, interaction ids, etc. Messages defined during this step can be used as inputs in various testing scenarios.

5.9 Summary

This chapter introduced a new testing methodology for testing the conformity of HL7 V3 applications, which materialized into a complete testing framework. The testing framework is based on standardized testing technologies, which makes it very reliable. Moreover, the testing framework is further enhanced with several ingenious automations that help testers throughout the testing process. The methodology and the testing framework were validated in a real environment. The analysis of the results obtained in the chosen case study showed convincing arguments that certify this approach.

Section 5.1 presented an overview on the particularities of conformance testing in healthcare environment, proposed a testing methodology suitable for HL7 V3 applications and discussed about the testing environment and the testing preconditions.

The testing methodology is based on the standardized TTCN-3 testing language. Test cases defined in the TTCN-3 testing language, which form the Abstract Test Specification (ATS), cannot be executed on their own. Their purpose is to define the features that are being tested in a generic way, for an abstract SUT. They do not handle system-specific information, such as communication channels, or message encodings. The execution of the test cases defined in ATS is mediated by a TTCN-3 test system. The chosen TTCN-3 test system is developed in Java and is part of the TTworkbench platform.

The overview of the SUT was considered, with focus on the important characteristics that influence the testing process, such as the fact that it uses an HL7 V3 integration profile, i.e. IHE QED, and that its communication interface is deployed as a web service.

A single testing prerequisite was identified for the proposed methodology. The requirement is that test designers have access to a set of XML Schemas that define HL7 V3 data types and

messages. This prerequisite is not a constraint imposed by the testing methodology and does not limit the flexibility of test designers. A way of validating the testing input is required anyways and the XSD format is a common option.

The testing workflow was presented, highlighting the communication between important components of the TTCN-3 test system and showing various transformations that are applied when exchanging messages during testing execution.

Section 5.2 introduced the first generation tool, which is mandatory to the automation of the testing process. The tool automatically generates Java classes that define HL7 V3 data types and messages. Two alternatives were analyzed:

- generating Java classes directly from MIF, using Java SIG;
- generating Java classes from a MIF-derived model, such as UML/XMI, or XML-based ITS Schemas.

The second alternative was considered more suitable. Thus, a generator of Java classes from XML ITS Schemas was designed and implemented. The generation is based on JAXB. A post-processing tool developed in Python supplemented the code generator.

Section 5.3 presented two other tools, which help test designers create the TTCN-3 type system needed for defining test cases. Defining the TTCN-3 type system manually is not suitable. Thus, a first tool was designed for generating data types needed to define input test data in TTCN-3 test language. The algorithms presented in this section detail the generation procedure. The second tool was tailored to help test designers define the TTCN-3 templates needed when defining the test cases. TTCN-3 templates are complex and difficult to define even by an experienced test designer and with a specialized editor, such as CL Editor from TTworkbench. The current work suggested that the input test data was defined in XML and converted automatically into a TTCN-3 template using the proposed TTCN-3 templates editor and generator.

Section 5.5 overviewed the particularities of defining input test data for HL7 V3 applications. This section presented the advantages of finding redundancy in testing input and discussed about clustering input data into equivalence partitions.

Section 5.6 is strongly linked to the previous one. It introduced a novel test data generator that can be used by test designers to automatically generate clusters of input test data. It can be used as a simple HL7 V3 message generator too, however, its true advantage lies in the capabilities to customize the generation of large sets of input test data and its ability to customize the similarity between the generated data. The complex architecture of TDG was provided in this section and the two types of customizations were presented: one based on various distances, which has localized effect, and one that has a generic effect. The implementation of the TDG and its challenges were described in Section 5.7.

Finally, a test case was considered in order to draw conclusions about the testing methodology and to analyze the advantages gained by automating the process. A full testing scenario was

presented, from the first definition of the test data input to the analysis of the testing results. During the process, the role of each automatic generation tool and their importance were highlighted.

6 CONCLUSIONS AND OUTLOOK

The interoperability of eHealth systems has met with a huge response from the research community in recent years, with healthcare associations and standardization organizations contributing to the advent of common sets of guidelines and communication standards.

One of the most active healthcare organizations is Continua Health Alliance (Continua). One of the key contributions Continua made to the eHealth domain was the proposal of an end-to-end network architecture and implementation guidelines that interconnect Personal Area Network (PAN) devices with different telehealth services – the Continua Reference Architecture.

The main objective of this research was to identify and address the problems that may occur when creating a healthcare environment derived from the Continua model and to validate it against real-life scenarios.

6.1 Personal contributions

The Continua Reference Architecture [CON12] defines four network interfaces: PAN-IF, LAN-IF, WAN-IF and HRN-IF. The research focuses on two of these interfaces, i.e. PAN-IF and HRN-IF, and on the standards designed to ensure communication for these interfaces: ISO/IEEE 11073-20601 Optimized Exchange Protocol (OEP) and Health Level 7 version 3 (HL7 V3), respectively. The analysis of these communication standards was the starting point for this research.

6.1.1 Security enhancements for ISO/IEEE 11073-20601

Optimized Exchange Protocol (OEP) is an ISO/IEEE standard promoted by the Continua Health Alliance and designed to support the transfer of vital signs between mobile medical devices. The

communication protocol was aimed at ensuring both syntactic and semantic interoperability between BAN devices.

OEP lacks important features that limit its use in several common real-life situations. One of the most important features OEP lacks is a mechanism to ensure security and privacy for the medical data exchanged. The standard specifies that the responsibility of ensuring security rests with the implementer. This leads to the development of proprietary solutions that address different security problems, thereby affecting important characteristics of OEP, such as interoperability and plug-and-play capability. In addition, the thesis identified the impossibility of implementing application-layer solutions for medical devices with embedded Agent or Manager software, such as the ones produced by Bluegiga.

The thesis suggests that OEP should not be used without security enhancements. Several real-life scenarios were discussed to demonstrate the importance of security in a healthcare system. The literature describes several approaches for addressing the problem of security. Those approaches do not, however, analyze the implications in terms of interoperability. The thesis proposes a different methodology for ensuring one important aspect of security: authentication. The approach put forward is based on biometric technology.

A two-level authentication process is proposed, consisting of:

- patient-to-device authentication, i.e. Patient to Agent;
- device-to-device authentication, i.e. Agent to Manager.

Authentication mechanism for OEP

Device-to-device authentication can be ensured by various existing authentication protocols. Patient-to-device authentication, on the other hand, is more challenging, considering the fact that Agents have low processing power and only provide a small range of plugin-able interfaces due to their reduced size.

The thesis describes several alternatives for the authentication process. Authentication based on fingerprints is considered the best solution because it has been applied for decades in person identification, is inexpensive, non invasive and can be used for authenticating patients in critical situations when other authentication types are not viable [EGN12a].

Section 2.3 focuses on the enhancement of OEP protocol with a very secure authentication mechanism, namely *mutual challenge-response authentication*. It is a well-known authentication protocol that has been successfully used in practice.

The novelty of the approach is that the keys used in the authentication process are derived from the patients' fingerprint templates. Thus, unique, individual biometric keys are generated for each individual patient from the characteristics of the topology determined by the minutiae points found in their fingerprint templates. As a result, the patient-to-device authentication process is personalized for each individual.

Original biometric key generation process

Although biometric technologies have reached a high level of maturity, generating robust biometric keys is still an open problem due to the inherent characteristics of biometric data. The literature describes several methods that attempt to address this problem. Section 2.2 presents a summary of these methods. The healthcare environment imposes special constraints, however, that make the methods inapplicable and the generation process even more difficult.

Section 2.2 describes an innovative approach to generating biometric keys. The methodology relies on the fact that even though there are differences between distinctive templates taken for the same finger, there are sufficient similarities that can be used for deriving biometric keys.

The method proposes obtaining several templates of the same finger, extracting and then determining the individual minutiae that are found in most of the templates. The templates are aligned and overlapped to form a super-template. The super-template contains all the common minutiae, which tend to group in clusters. A clustering algorithm is applied and a center is calculated for each resulting cluster. The cluster's center represents the class of minutiae that are commonly found in the various fingerprint templates.

Section 2.2 highlights three variations for representing the topology of the clusters' centers:

- static radial division of the feature space [EGN12a];
- Voronoi tessellation [EGN12b];
- Delaunay triangulation.

In each of the three methods the clusters' centers were mapped to values determined by their position relative to the overall distribution. These values are then concatenated into a string that represents the biometric key. The three variations were analyzed, and the results showed that the best topology representation is based on the Delaunay triangulation.

Enhancements of the OEP standards specifications

Chapter 3 details two enhancements proposed for the OEP protocol:

- authentication based on a biometric key [EGN12a];
- mutual initiation of the association procedure [EGN13a].

Chapter 3 discusses the technical details of enhancing OEP and offers several implementation guidelines, as well as examples of messages exchanged. The mutual initiation of the association procedure indicates the fact that the Manager should also be allowed to initiate the communication. Several common scenarios demonstrating the need for this enhancement were discussed.

The OEP extensions are designed to support interoperability between the mobile medical devices. They are designed as optional features, in order to enable backward compatibility with current implementations of the standard.

New message analyzer for OEP

Section 3.5 presents the design and implementation of a message analyzer for OEP protocol [EGN12c]. There is no open source analyzer currently available. Proprietary tools developed by manufacturers exist but are not available for research purposes. This is an important limitation that restricts the expansion of the standard, as developers are required to extend OEP implementations to include features such as security, or identity management, without being able to analyze the results.

The thesis proposes a solution based on a widely known network protocol analyzer: Wireshark. The proposed OEP analyzer is developed as a Wireshark plugin and is currently limited to TCP/IP communication. The analyzer was used for validating the OEP enhancements proposed in Sections 3.3 and 3.4.

6.1.2 New conformance testing method for testing HL7 V3-based applications

Continua Health Alliance promotes HL7 V3 as the messaging standard for transferring medical data between the aggregation manager and the Healthcare Information System (HIS). HL7 V3 is a very complex standard that has not reached mainstream usage yet due to its lack of backward compatibility with V2.x, the previous version of the standard. There have been many attempts to ensure interoperability between the two versions, but full interoperability has not been achieved so far.

The interoperability of HL7 V3-based applications is also influenced by the way HL7 V3 messages are defined. The backbone of the standard is HL7 RIM (Reference Information Model), the fundamental model from which all HL7 V3 messages are derived. The standard is so generic that cannot be used without narrowing it to healthcare domains. HL7 V3 integration profiles have been released to help healthcare systems improve interoperability.

In this context, testing plays an important role in achieving interoperability between HL7 V3-based applications. This research has focused on testing the conformance of applications to various HL7 V3 integration profiles as a first step towards ensuring full interoperability.

The novel testing methodology [EGN12d] presented in the thesis is not designed for a specific HL7 V3 integration profile. It can therefore be extended to virtually any integration profile provided that several requirements are met. The testing methodology is based on standard testing technologies, which makes the process very reliable. Automation tools were also designed and implemented in order to achieve a fully automated testing process.

New testing framework for conformance testing

The first contribution in terms of the conformance testing of HL7 V3-based applications presented in the thesis is the *testing framework* [EGN10]. The testing framework is based on the TTCN-3 test system and TTCN-3 testing language. TTCN-3 technologies have been successfully used for testing HL7 V2.x-based applications as well. The framework was validated on a Patient

Information Management System that uses the IHE QED HL7 V3 integration profile. The components of the TTCN-3 test system can, however, be adapted to other integration profiles as well. Several automation tools were designed for this purpose and integrated into the testing framework.

HL7 V3 data type generator for Java

There are various aspects that make testing difficult regardless of the technology used. The thesis analyzed, for instance, the problem of generating HL7 V3 data types in common programming languages [EGN11]. Several research projects address the problem of generating HL7 V3 data types in Java. The solutions proved to have flaws that hinder the automation process.

Section 5.2 argued the need for Java data types in order to automate the testing process and presented a new modular approach that covers two phases:

- generation of raw data types;
- adjustment of data type definitions.

Both phases depend on the programming language and were designed for Java. The second phase consists of a post-processing tool that can be extended by adding scripts written by test designers.

Original TTCN-3 type system components generator

The automation process also hinges on the existence of the same HL7 V3 data types defined in TTCN-3. Section 5.8 demonstrates that is impossible to define Java and TTCN-3 data types manually. Accordingly, another methodology for automatic generation of user-defined TTCN-3 data types was proposed [EGN11]. Section 5.3 presented the architectural design of this approach. Algorithms and implementation guidelines were also provided for the generation process.

Innovative Input test data generator – the first HL7 V3 message generator

Probably the most significant contribution to automated conformance testing is the input test data generator. An important step in the testing process is the selection of adequate input test data. Researchers acknowledge the need for a generic HL7 V3 message generator that can be used to create input test data independently of the testing technology. This is difficult to achieve due to the complexity of the standard and its message design model.

The thesis proposes a novel methodology for generating input test data [EGN13b]. The novelty of the approach is inherent in the way new messages are created. There are two means of customizing the generated messages. While the input test data generator provides the means for customizing the message as a whole, thereby offering the possibility of creating message templates, the core of the customization lies in the distance-based generators.

Distance-based generators use different distances for generating new values of particular similarity. The integrated pre-defined distances, such as Levenshtein or Euclidean distance, can

be adjusted. Moreover, testers can define new custom distances as well. The generator is customizable in such a way that each individual field of the generated message can be adjusted.

The input test data generator can be used as an HL7 V3 message generator. It is in effect the first HL7 V3 message generator available, which will have a major impact on the HL7 V3 developer community. The generator shows its true potential, however, when customized to create inputs for specific testing methodologies. The case study presented in Section 5.8 shows how the generator can be used to create inputs grouped in equivalence partitions for an Equivalence Class Testing approach.

6.2 Future directions

An important aspect of future explorations is research on more superior methods of representing the minutiae's topology in order to improve the robustness of the biometric key. Another research direction in extension to the ones presented in the thesis could focus on finding better invariants that influence the values approximating the clusters' centers.

The message analyzer developed for OEP currently relies on the static model of the communication protocol, as defined by the standard. The analyzer could be enhanced to generate the model automatically from an ASN.1 definition, and thus become more flexible in respect of standard changes.

Another research aspect not covered in the thesis is the conversion of medical data between ISO/IEEE 11073-20601 and HL7 V3 and the dynamic mapping between the two protocols' data models.

6.3 Summary

The eHealth domain is faced with the challenge of continuous expansion as a result of the intensive research conducted by software and hardware manufacturers. The interest in achieving interoperability is set to intensify in response to the growing support and promotion of remote healthcare. The thesis has identified several sub-areas of the eHealth domain where important needs are waiting to be solved.

The extension of eHealth communication standards with an innovative authentication mechanism based on biometric keys proposed in the thesis can play an important role in ensuring the security and privacy of the patients' medical data. Moreover, the original testing methodology based on standardized technologies, the complex testing framework and the innovative automated tools proposed can significantly improve the process of testing healthcare applications and ultimately improve the interoperability between eHealth systems.

ACRONYMS

ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
ATS	Abstract Test Specification
BAN	Body Area Networks
BIO_KEY	Biometric Key
CCOW	Clinical Context Object Workgroup
CD	Coder/Decoder
CDA	Clinical Document Architecture
CH	Component Handling
D-MIM	Domain Message Information Model
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DICOM	Digital Imaging and Communications in Medicine
DIM	Domain Information Model
DLL	Dynamic-link Library
DOM	Document Object Model
DVTk	DICOM Validation Toolkit
EHC	Electronic Health Card
EHR	Electronic Health Record
EMF	Eclipse Modeling Framework
EPS	Electronic Prescription System
ETSI	European Telecommunications Standards Institute
FAR	False Acceptance Rate
FBI	Federal Bureau of Investigations
FMD	Finger Minutiae Data
FRR	False Rejection Rate
FSM	Finite State Machine
GUI	Graphical User Interface
HAPI	HL7 Application Programming Interface
HDF	HL7 Development Framework
HMD	Hierarchical Message Definition
HDP	Health Data Profile
HIS	Healthcare Information System

HL7	Health Level Seven
HM3	Hospital Manager version 3
HRN	Health Record Network
HTML	HyperText Markup Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IHE	Integrating the Healthcare Enterprise
ISO	International Organization for Standardization
ITS	Implementable Technology Specification
JAX-WS RI	Java API for XML Web Services – Reference Implementation
JAXB	Java Architecture for XML Binding
JAXP	Java API for XML Processing
LLP	Lifelong Learning Programme
LOINC	Logical Observation Identifiers Names and Codes
MD5	Message-Digest 5
MDC	Medical Device Communication
MDER	Medical Device Encoding Rules
MDF	Message Development Framework
MIF	Model Interchange Format
MITM	Man in the middle
MLM	Medical Logic Modules
MSC	Message Sequence Chart
MVC	Model View Controller
MWB	Messaging Workbench
NEHTA	National E-Health Transition Authority
NIST	National Institute of Standards and Technology
OEP	ISO/IEEE 11072-20601 Optimized Exchange Protocol
OOB	Out Of Band channel
OWL	Web Ontology Language
PA	Platform Adapter
PACS	Picture Archival and Communication System
PAN	Personal Area Network
PCC	Patient Care Coordination
PCC TF	Patient Care Coordination Technical Framework
PCD	Patient Care Device
PDU	Presentation Data Unit
PPPoE	Point-to-Point Protocol over Ethernet
QED	Query for Existing Data
R-MIM	Refined Message Information Model
ReTeMeS	Reliability Testing of Medical Systems

RIM	Reference Information Model
ROI	Region Of Interest
RS	Runtime System
RTDG	Raw Test Data Generator
SA	SUT Adapter
SDK	Software Development Kit
SDO	Standards Developing Organization
SHA	Secure Hash Algorithm
SIM	Serializable Information Model
SNOMED	Systematized Nomenclature of Medicine
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSP	Secure Simple Pairing
SUT	System Under Test
SVM	Support Vector Machines
TCI	TTCN-3 Control Interface
Tcl	Tool Command Language
TCP	Transmission Control Protocol
TDC	Test Data Converter
TDG	Test Data Generator
TE	TTCN-3 Executable
TH	HL7 V3 Test Harness
TM	Test Management
TMC	Test Management & Control
TRI	TTCN-3 Runtime Interface
TTCN-3	Testing and Test Control Notation version 3
UML	Unified Modeling Language
UML/XMI	Unified Modeling Language XML Metadata Interchange
VTS	Validation Test Suite
WAN	Wide Area Networks
WHO	World Health Organization
WSDL	Web Service Definition Language
XML	Extensible Markup Language
XPath	XML Path Language
XSD	XML Schema

LIST OF PUBLICATIONS RELATED TO THIS THESIS

A. Egner, A. Soceanu and F. Moldoveanu, "Biometric Identity Management for Standard Mobile Medical Networks," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, San Diego, CA, 2012, pp. 2186-2189.

A. Egner, A. Soceanu and F. Moldoveanu, "Managing secure authentication for standard mobile medical networks," in *IEEE Symposium on Computers and Communications (ISCC)*, Cappadocia, 2012, pp. 390-393.

A. Egner, F. Moldoveanu, N. Goga, A. Moldoveanu, V. Asavei and A. Morar, "Enhanced communication protocol for ISO/IEEE 11073-20601," *The Scientific Bulletin of University POLITEHNICA of Bucharest, Series C Electrical Engineering and Computer Science*, vol. 75, no.2, 2013, to be published.

A. Egner, A. Soceanu, F. Moldoveanu, C. Ferrari and M. Moro, "Towards Secure e-Health Interoperable Personal Networks," in *4th International Workshop on Ambient Assisted Living (IWAAL)*, Vitoria-Gasteiz, 2012, to be published.

A. Egner, F. Moldoveanu, A. Moldoveanu, V. Asavei, A. Morar and C. Boiangiu, "Testing the interoperability of HL7-based applications using TTCN-3," in *The 21st International DAAAM Symposium*, Zadar, Croatia, 2010, pp. 1279-1280. (ISI)

A. Egner, F. Moldoveanu and N. Goga, "Using TTCN-3 for testing the interoperability of HL7v3 based applications," *European Journal of Biomedical Informatics (EJBI)*, vol. 8, no. 4, 2012, to be published.

A. Egner, F. Moldoveanu, A. Moldoveanu, V. Asavei and A. Morar, "Automated generation of TTCN-3 type system used for testing of healthcare applications," in *The 18th International Conference on Control Systems and Computer Science (CSCS)*, Bucharest, 2011, pp. 794-799.

A. Egner, F. Moldoveanu, N. Goga, A. Moldoveanu, V. Asavei and A. Morar, "Customized Test Data Generator for HL7v3 based Healthcare Information Systems," *Journal of Control Engineering and Applied Informatics*, to be published. (ISI)

RESEARCH PROJECTS

EUREKA Project – 4053 ReTeMeS: Reliability Testing of Medical Systems (2009 – 2010) http://www.eurekanetwork.org/project/-/id/4053

Leonardo da Vinci Programme – SMECluster: Strategic Planning for Sustainable Clustering of Collaborative SMEs (2009-2010)

Leonardo da Vinci Programme – SMENet: Establishment of Sustainable Collaborative SME Networks (2009-2010)

Eurostars Project – 5112 RELIS: Risk Detection in Laboratory Information Systems (2010-2012) http://www.eurekanetwork.org/project/-/id/5112

Eurostars Project – 5119 EUGEN: Enterprise Unified Guideline Engine (2010-2012) http://www.eurekanetwork.org/project/-/id/5119

Eurostars Project – 6126 VISUAL-D: Visualization of Patient Data for Easy Management of Care Processes (2011-present) http://www.eurekanetwork.org/project/-/id/6126

EUREKA Project – MORIS F.D.: Medical Operational Risks Identification Service and Fraud Detection (2011-present) http://www.eurekanetwork.org/project/-/id/5884

BIBLIOGRAPHY

[BEN09] T. Benson, *Principles of Health Interoperability HL7 and SNOMED (Health Informatics)*, 1st ed.: Springer, 2009

[BHA10] A. Bhargav-Spantzel et al., "Biometrics-based identifiers for digital identity management," in *Proceedings of the 9th Symposium on Identity and Trust on the Internet*, Gaithersburg, MD, 2010, pp. 84-96

[BLU] Bluegiga. [Online]. Available: http://www.bluegiga.com/home

[BON] W. Bonney and P. Rontey. *Messaging Workbench (MWB)* [Online]. Available: http://gforge.hl7.org/gf/project/mwb

[BOX00] D. Box et al. (2000, May 8). *Simple Object Access Protocol (SOAP) 1.1*. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[BS04] *Health Informatics - Point-Of-Care Medical Device Communication - Part 20101: Application Profile - Base Standard*, IEEE Standard 11073-20101:2004

[CAR07] R. Carroll et al., "Continua: An Interoperable Personal Healthcare Ecosystem," *IEEE Pervasive Computing*, vol. 6, no. 4, 2007, pp. 90-94

[CCH] The Certification Commission for Health Information Technology. [Online]. Available: http://www.cchit.org

[CHI06] R. Chinnici et al. (2006). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language* [Online]. Available: http://www.w3.org/TR/wsdl20

[CIL07] R.L. Cilibrasi and P.M.B. Vitanyi, "The Google Similarity Distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, March 2007, pp. 370 - 383

[CIU06] I. Ciupa et al., "Object distance and its application to adaptive random testing of object-oriented programs," in *Proceedings of the 1st International Workshop on Random Testing*, 2006, pp. 55–63

[CLA99] J. Clark and S. DeRose. (1999, November 16). *XML Path Language (XPath)* [Online]. Available: http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/atc_comms_services/swim/documentation/media/compliancy/Xpathv1.0.pdf

[CON] Integrating the Healthcare Enterprise. *IHE Connectathons: A Unique Testing Opportunity* [Online]. Available: http://www.ihe.net/connectathon

[CON12] Continua Health Alliance. (2012, October 29). *About the Alliance* [Online]. Available: http://www.continuaalliance.org/about-the-alliance.html

[CYP] Project Cypress. *Clinical Quality Measure Testing And Certification Tool* [Online]. Available: http://projectcypress.org

[DAV88] M. Davis and E. Weyuker, "Metric space-based test-base adequacy criteria," *The Computer Journal*, vol. 31, no. 1, 1988, pp. 17-24

[DEL10] DeltaWare Systems. (2010, October). *HL7v3 Test Harness* [Online]. Available: http://sourceforge.net/projects/hl7-testharness

[DIM04] *Health Informatics - Point-Of-Care Medical Device Communication - Part 10201: Domain Information Model*, IEEE Standard 11073-10201:2004

[DUN10] J.P. Dunning, "Taming the Blue Beast: A Survey of Bluetooth Based Threats," *IEEE Security & Privacy*, vol. 8, no. 2, March-April 2010, pp. 20-27

[EGN10] A. Egner et al., "Testing the interoperability of HL7-based applications using TTCN-3," in *The 21st International DAAAM Symposium*, Zadar, Croatia, 2010, pp. 1279-1280

[EGN11] A. Egner et al., "Automated generation of TTCN-3 type system used for testing of healthcare applications," in *The 18th International Conference on Control Systems and Computer Science (CSCS)*, Bucharest, 2011, pp. 794-799

[EGN12a] A. Egner et al., "Biometric Identity Management for Standard Mobile Medical Networks," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, San Diego, CA, 2012, pp. 2186-2189

[EGN12b] A. Egner et al., "Managing secure authentication for standard mobile medical networks," in *IEEE Symposium on Computers and Communications (ISCC)*, Cappadocia, 2012, pp. 390-393

[EGN12c] A. Egner et al., "Towards Secure e-Health Interoperable Personal Networks," in *4th International Workshop on Ambient Assisted Living (IWAAL)*, Vitoria-Gasteiz, 2012, to be published

[EGN12d] A. Egner et al., "Using TTCN-3 for testing the interoperability of HL7v3 based applications," *European Journal of Biomedical Informatics (EJBI)*, vol. 8, no. 4, 2012, to be published

[EGN13a] A. Egner et al., "Enhanced communication protocol for ISO/IEEE 11073-20601," *The Scientific Bulletin of University POLITEHNICA of Bucharest, Series C Electrical Engineering and Computer Science*, vol. 75, no.2, 2013, to be published

[EGN13b] A. Egner et al., "Customized Test Data Generator for HL7v3 based Healthcare Information Systems," *Journal of Control Engineering and Applied Informatics*, to be published

[ENG04] T. R. Eng, "Population health technologies: Emerging innovations for the health of the public," *American Journal of Preventive Medicine*, vol. 26, no. 3, April 2004, pp. 237-242

[EUR10] Eureka. (2010, November). *Reliability Testing of Medical Systems* [Online]. Available: http://www.eurekanetwork.org/project/-/id/4053

[EYS01] G. Eysenbach, "What is e-health?," *Journal of Medical Internet Research*, vol. 3, no. 2, June 2001

[FMD11] *Biometrics — Biometric Data Interchange Formats — Part 2: Finger Minutiae Data*, ISO/IEC 19794-2:2011

[FOR07] B. Forouzan, *Cryptography & Network Security*, 1st ed.: McGraw-Hill, 2007

[FRO09] Frost & Sullivan, "E-Healthcare in Western Europe A Huge Market Opportunity for Wireless Technologies," 2009

[FYF12] J. Fyfe et al., "Everest: a framework for developing HL7v3 applications," *ACM SIGHIT Record*, vol. 2, no. 1, p. 24, March 2012

[GAZ] Integrating the Healthcare Enterprise Gazelle. *Interoperability Conformance Testing for eHealth Information Systems* [Online]. Available: http://gazelle.ihe.net

[GOW82] J.C. Gower, "Euclidean Distance Geometry," *Math. Scientist 7*, 1982, pp. 1-14

[GRI] Griaule Biometrics. *Fingerprint SDK* [Online]. Available: http://www.griaulebiometrics.com/page/en-us/fingerprint_sdk

[GUS97] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, 1st ed.: Cambridge University Press, 1997

[HAA08] K.M.J. Haataja and K. Hypponen, "Man-In-The-Middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures," in *3rd International Symposium on Communications, Control, and Signal Processing (ISCCSP2008)*, St Julians, 2008, pp. 1096 - 1102

[HAP] HL7 Application Programming Interface. [Online]. Available: http://hl7api.sourceforge.net

[HDF] Health Level 7 International. *HL7 Development Framework* [Online]. Available: http://hdf.wikispaces.com

[HGF] HL7 GForge. *Home for HL7 Projects* [Online]. Available: http://gforge.hl7.org/gf

[HIN05] A. Hinchley, *Understanding Version 3: A Primer on the HL7 Version 3 Communication Standard*, 3rd ed.: Alexander Mönch Publishing, 2005

[HJE06] Ø. Hjelle and M. Dæhlen, *Triangulations and Applications (Mathematics and Visualization)*, 1st ed. NL: Springer, 2006

[HL7] Health Level 7 International. [Online]. Available: http://www.hl7.org

[HTP] HL7 Application Programming Interface. *HAPI TestPanel* [Online]. Available: http://hl7api.sourceforge.net/hapi-testpanel

[IHE] Integrating the Healthcare Enterprise. *Welcome to Integrating the Healthcare Enterprise* [Online]. Available: http://www.ihe.net

[JAI01] A.K. Jain et al., "Fingerprint matching using minutiae and texture features," in *Proceedings of the International Conference on Image Processing (ICIP)*, Thessaloniki, Greece, 2001, pp. 282-285

[JAI99] A.K. Jain et al., "Data Clustering: A Review," *ACM Computing Surveys (CSUR)*, vol. 31, no. 3, September 1999, pp. 264-323

[JAR11] R. Jarrin, "Identifying barriers and enablers for device interoperability," *Continua Health Alliance*, 2011

[JAX12] Java.net. (2012, October 19). *JAX-WS Reference Implementation (RI) Project* [Online]. Available: http://jax-ws.java.net

[JSI07] Pragmatic Data. (2007, September 20). *The HL7 Java SIG Project* [Online]. Available: http://aurora.regenstrief.org/javasig

[JUE02] A. Juels and M. Sudan, "A fuzzy vault scheme," in *Proceedings of IEEE International Symposium on Information Theory*, 2002, p. 408

[KRA88] G.E. Krasner and S.T. Pope. (1988). *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System* [Online]. Available: http://www.itu.dk/courses/VOP/E2005/VOP2005E/8_mvc_krasner_and_pope.pdf

[LAR99] J. Larmouth, *ASN.1 Complete*, 1st ed.: Morgan Kaufmann, 1999

[LOH] LibreSoft. *OpenHealth* [Online]. Available: http://libresoft.es/research/software/openhealth

[MAI97] D. Maio and D. Maltoni, "Direct gray-scale minutiae detection in fingerprints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, January 1997, pp. 27 - 40

[MCA09] McAfee. (2009). *Mobile Security Report 2009* [Online]. Available: http://www.mcafee.com/us/resources/reports/rp-mobile-security-2009.pdf

[MCK] L. McKenzie et al. *V3 Generator* [Online]. Available: http://gforge.hl7.org/gf/project/v3-generator

[MOR00] R. Mordani et al., "Java API for XML Processing," Sun Microsystems, Inc., Palo Alto CA, Version 1.1 Final Release, 2000

[MTS08] *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3*; Part 1: TTCN-3 Core Language, ETSI Standard ES 201 873-1 V3.3.2, 2008

[MUH] R.B. Muhammad. *Delaunay Tesselation* [Online]. Available: http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/CG-Applets/DelaTessel/delacli.htm

[NEH] National E-Health Transition Authority. *E-Health Compliance and Conformance* [Online]. Available: http://www.nehta.gov.au/connecting-australia/cca

[NIS04] The National Institute of Standards and Technology. (2004, November 2) *HL7 Conformance Testing with Message Maker* [Online]. Available: http://www.itl.nist.gov/div897/ctg/messagemaker

[NIS11] The National Institute of Standards and Technology. (2011, July 29). *NIST HL7 V2 Testing Toolkit* [Online]. Available: http://hl7v2tools.nist.gov

[NOM04] *Health informatics - Point-of-care medical device communication - Part 10101: Nomenclature*, IEEE Standard 11073-10101:2004

[OEP08] *Health Informatics - Personal Health Device Communication Part 20601: Application Profile - Optimized Exchange Protocol*, IEEE Standard 11073-20601-2008

[OH05] H. Oh et al., „What Is eHealth (3): A Systematic Review of Published Definitions," *Journal of Medical Internet Research*, vol. 7, no. 1, 2005

[OHT10] Open Health Tools. (2010). *Model-Driven Health Tools (MDHT)* [Online]. Available: https://www.projects.openhealthtools.org/sf/projects/mdht

[OKA00] A. Okabe et al., *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. Chichester, England: John Wiley and Sons Ltd, 2000

[OUS89] J.K. Ousterhout, "Tcl: An Embeddable Command Language," University of California at Berkeley Berkeley, CA, USA, Technical Report 1989

[PCC08] *Patient Care Coordination (PCC) Technical Framework Supplement 2008-2009. Query for Existing Data (QED)*, IHE, 2008

[PCD12] *Patient Care Device Technical Framework Volume 1 (PCD TF-1) Integration Profiles*, Revision 2.0, IHE, 2012

[POT07] G. Potter et al., "Mastering DICOM with DVTk," *Journal of Digital Imaging*, vol. 20, no. 1, November 2007, pp. 47-62

[RIV04] J. desRivieres and J. Wiegand, "Eclipse: A platform for integrating development tools," *IBM Systems Journal*, vol. 43, no. 2, 2004, pp. 371 - 383

[SEL01] R. Seliger. (2001). *Overview of HL7's CCOW Standard* [Online]. Available: http://www.hl7.org/documentcenter/public/wg/sigvi/CCOW_overview_2001.doc

[SHA10] A. Shabo et al. (2010, March 4). *Use DB2 pureXML to implement a healthcare industry data solution* [Online]. Available: http://www.ibm.com/developerworks/data/library/techarticle/dm-1002purexmlqed

[SPR] R. Spronk. *V3 Software Implementation* [Online]. Available: http://www.ringholm.com/download/implementation_mechanics.ppt

[SPR09] W. Springer. (2009, October). *A Java library for generating random text from regular expressions* [Online]. Available: http://code.google.com/p/xeger

[TTW] Testing Technologies. *TTworkbench - The Reliable Test Automation Platform* [Online]. Available: http://www.testingtech.com/products/ttworkbench.php

[VAJ04] S. Vajjhala and J. Fialli, *The Java Architecture for XML Binding (JAXB) 2.0*, 4th ed., 2004

[VEG08] D. Vega et al., "A TTCN-3-based test automation framework for HL7-based applications and components," in *Proceeding of the Conference on Quality Engineering in Software Technology (CONQUEST)*, Potsdam, 2008

[VEG10] D.E. Vega et al., "Design of a Test Framework for Automated Interoperability Testing of Healthcare Information Systems," in *Second International Conference on eHealth, Telemedicine, and Social Medicine (ETELEMED)*, St. Maarten, 2010, pp. 134 - 140

[VLI07] E. van der Vlist, *Schematron*: O'Reilly Media, 2007

[VOR] Washington University in St. Louis. *Lecture 16: Fortune's Algorithm and Voronoi diagrams* [Online]. Available: http://www.cs.wustl.edu/~pless/546/lectures/L11.html

[WAR09] F. Wartena et al., "Continua: The Impact of a Personal Telehealth Ecosystem," in *International Conference on eHealth, Telemedicine, and Social Medicine, 2009. eTELEMED '09*, Cancun, 2009, pp. 13-18

[WHO05] World Health Organization. (2005, May). [Online]. Available: http://apps.who.int/iris/bitstream/10665/20378/1/WHA58_28-en.pdf

[WIL05] C. Willcock et al., *An introduction to TTCN-3*. Chichester, England: John Wiley & Sons, 2005

[WIR] Wireshark. [Online]. Available: http://www.wireshark.org

[WOO00] L. Wood et al. (2000, September 29). *Document Object Model (DOM) Level 1 Specification (2nd ed.)* [Online]. Available: http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/DOM.pdf

[YUJ07] L. Yujian and L. *Bo, "A Normalized Levenshtein Distance Metric," IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, June 2007, pp. 1091 - 1095

[ZHA04] W. Zhang et al., "Optimal thresholding for key generation based on biometrics," in *International Conference on Image Processing (ICIP)*, Singapore, 2004, pp. 3451–3454

[ZVE] Zvetco Biometrics. *P5000 Fingerprint Device* [Online]. Available: http://www.zvetcobiometrics.com/Products/P5000/overview.php