Christoph ARTHOFER

Manufacturer Independent Interface for Cardiac Rhythm Management

Master thesis

Institute for Genomics and Bioinformatics
University of Technology Graz
Petersgasse 14, A - 8010 Graz

Mentor:
DI Dr. Dieter Hayn, AIT Austrian Institute of Technology GmbH

Reviewer:
Univ.-Doz. DI Dr. Günter Schreier, MSc

Graz, January, 2013

This thesis was established

in cooperation with

AIT Austrian Institute of Technology GmbH

Safety & Security Department

Assistive Healthcare Information Technology

Reininghausstraße 13/1

8020 Graz

Austria

Acknowledgements

I want to thank everyone who supported me in the development and completion of this thesis.

First and foremost I want to thank my supervisors Dieter Hayn and Günter Schreier for their constructive input and experienced guidance and the whole AHIT team of Austrian Institute of Technology for the great cooperation.

Furthermore special thanks go to my family for supporting me in every possible way during my entire education.

Zusammenfassung

Herstellerunabhängige Schnittstelle für Herzrhythmus-Management Systeme

Einleitung: Die hohe Prävalenz von kardiovaskulären Krankheiten weltweit führt zu einer großen Anzahl an Patienten mit implantierten Geräten. Um die Fülle an Daten aus Nachsorgeuntersuchungen zu verwalten wurde ein Herzrhythmus-Management System namens PICARD entwickelt. Obwohl Standards für den Austausch von Gerätedaten existieren, verwenden die meisten Hersteller nach wie vor proprietäre Formate. Deshalb konnten die Daten nur manuell in das PICARD-System eingegeben werden.

Ziele: Zusätzlich zur manuellen Dateneingabe sollte eine Schnittstelle entwickelt und in den bestehenden Ablauf integriert werden. Diese sollte dem Anwender die Möglichkeit bieten, proprietäre Datenformate schnell, einfach und automatisiert einzulesen.

Methoden: Die Schnittstelle (SPRINT) wurde als Java Web Start Anwendung realisiert und der Datenaustausch mit PICARD wurde nach dem Schema des IHE Profils RFD implementiert. Die proprietären Dateien zweier Hersteller (Biotronik und Boston Scientific) konnten über Bluetooth oder USB-Stick importiert und weiterverarbeitet werden.

Ergebnisse: SPRINT wurde in den bisherigen Ablauf integriert. Importierte herstellerspezifische Parameter von Programmern wurden automatisch in PICARD-spezifische Parameter konvertiert und mit den manuell eingegebenen Daten aus dem Webformular zusammengeführt. Die verarbeiteten Daten und aufgetretenen Konflikte wurden dann wieder an PICARD übertragen und dem Anwender dargestellt.

Schlussfolgerung: SPRINT wurde als Prototyp implementiert und mit exportierten Beispieldatensätzen von Programmern getestet. Die Schnittstelle bildete die Basis für eine homogene Datenstruktur in PICARD und einen standardisierten Austausch von Daten bzw. deren standardisierte Speicherung.

Keywords: Ambulante Nachsorgeuntersuchung von Implantaten, Kardiologie, Interoperabilität

Abstract

Manufacturer Independent Interface for Cardiac Rhythm Management

Introduction: The high prevalence of cardiovascular disease worldwide leads to a high number of indications for cardiovascular implantable devices and follow-up encounters. To manage this large amount of patient- and device-related data, the cardiac rhythm management system PICARD was developed. Although standards for the exchange of CIED data exist, most manufacturers still use proprietary data formats. Hence data had to be entered in the PICARD web form manually up to now.

Objectives: The aim of this thesis was to develop and integrate an interface in the current workflow, which allowed users a fast, easy and automated import of data from proprietary files of various manufacturers, additionally to the manual input of data in the web form.

Methods: The interface (SPRINT) was implemented as a Java Web Start application and the data exchange with PICARD was based on the scheme of the IHE Profile RFD. SPRINT supported a connection via Bluetooth and data import from an USB drive. It was able to process file formats from programmers of two different manufacturers (Biotronik and Boston Scientific).

Results: SPRINT was integrated in the current workflow. Imported vendor-specific data files from the programmers were automatically converted into a PICARD specific parameter format and then merged with manually entered data from the PICARD web form. The processed data and emerged discrepancies were transferred back to PICARD and displayed to the user.

Conclusion: SPRINT was prototypically implemented and tested using existing examples of exports from different programmers. The interface built the base for a homogeneous data structure in PICARD and further standardized data exchange and storage.

Keywords: Ambulatory follow-up of cardiac implants, Cardiology, Interoperability

Contents

Chapter 1

Introduction

1.1 Rationale

In the last few decades cardiovascular disease has become the leading cause of death worldwide, the number of patients with cardiovascular implantable electronic devices has rapidly increased and device complexity and indications have expanded. The huge amount of patient- and device-related data from various sources leads to an increasing follow-up burden and requires reorganization of the management.

Based on research done at AIT Austrian Institute of Technology GmbH a cardiac rhythm management system was developed. It represented a central management platform capable of handling all relevant information for an optimal therapy of the patient.

Although standardized medical data formats for the exchange and storage of cardiovascular device data exist, they are not applied in most of the cases. Instead, there exists a variety of proprietary file formats.

The objective of this thesis was to design and prototypically implement an interface which allows for the automated import of various vendor-specific data formats into the central management platform. The main focus was on the mapping of the proprietary parameters to platform-specific parameters and the integration of manually entered data and data from the proprietary files.

1.2 Physiology of the Heart

The heart is a hollow muscle, having the size of a fist and a mass between 250 and 350 g. It is divided into four chambers and provides the continuous blood circulation by its pumping action at a typical resting rate of about 60 to 80 beats per minute in adults.

Contraction of the heart is triggered by its electrical conduction system, which can be seen in Figure 1. The rhythmical stimulation is initiated by the sinoatrial node (SA node), which represents the primary impulse generator (pacemaker) located in the right atrium. Its cells spontaneously depolarise at a frequency of about 60 to 80 beats per minute. The stimulation spreads throughout the atrial myocardium and causes an atrial contraction. By way of the atria, the action potential reaches the atrioventricular node (AV node) which acts as a conductor between atria and ventricles. It is the only pathway for the impulses to enter the ventricles, located in the posterior-inferior region of the interatrial septum and it delays the forwarded impulses. This delay is extremely important to ensure that the atria are completely contracted prior to ventricular depolarisation. If the impulse generation of the SA node fails, the AV node is still able to take over. It stimulates the heart at a frequency of about 40 beats per minute and acts as a secondary pacemaker. Then the impulses travel to the Bundle of His, located at the base of the ventricle from where they are forwarded to the left and right bundle branches. Furthermore the left bundle branch consists of two fascicles. The branches divide into an extensive system of Purkinje fibers, which leads to a rapid depolarisation of the ventricular myocytes throughout both ventricles. The action potential of the myocytes lasts for about 200 to 400 ms, which ensures that sooner stimulated parts are still in refractory state while later parts are stimulated [1].

The electrical activation of the heart can be visualized by the Electrocardiogram (ECG). Within the ECG, conduction through the atria is represented by the so-called P wave, which is followed by an isoelectric line. Thereafter, the QRS complex can be seen in the ECG, which corresponds to conduction of the electrical activation through the ventricles. The PQ interval is defined as the time from the beginning of the P wave to the beginning of the QRS complex. Each heart beat ends with the T wave which originates from the re-polarization of the ventricles.

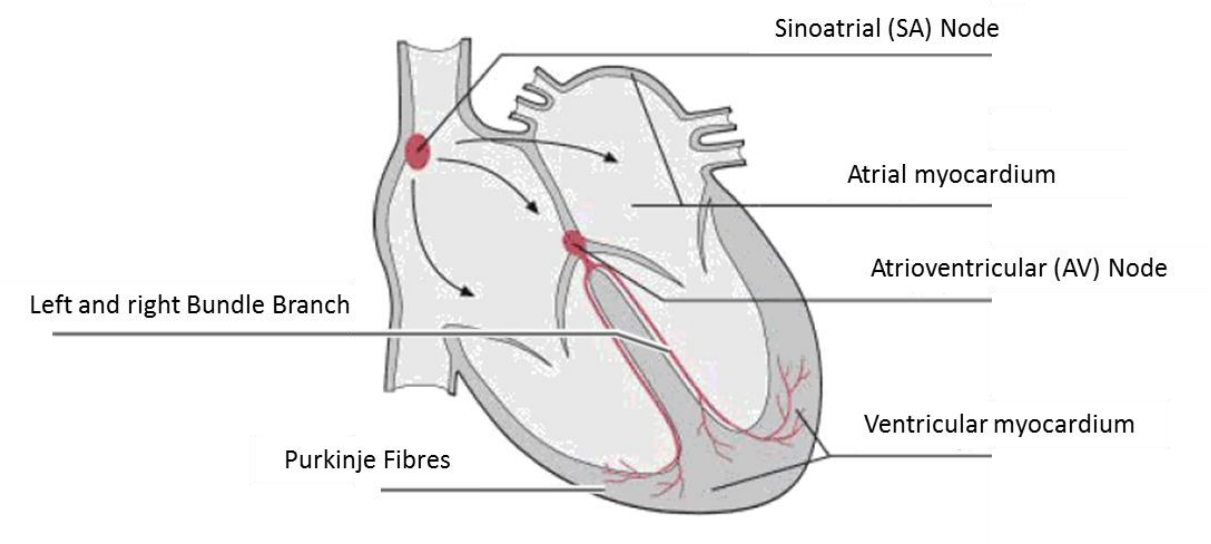

Figure 1: Electrical conduction system of the human heart.
The impulse is generated by the SA Node and spreads throughout the atrial myocardium. The AV Node conducts the impulse to the left and right bundle branches, the Purkinje fibres and finally the ventricular myocardium, which results in a contraction of the ventricles. Adapted from [1].

1.3 Pathology

This chapter describes cardiovascular disease in general and common cardiac arrhythmias (bradycardia, tachycardia, heart failure) in particular. The description focuses on those aspects that are related to devices for cardiac rhythm management and to their follow-ups. More details can be found within the references.

1.3.1 Cardiovascular Disease

As Fuster et al. [2] describe, cardiovascular disease is the leading cause of death worldwide. It is not only a major problem of developed and high-income countries anymore. In fact the prevalence of risk factors for cardiovascular disease and related chronic diseases has increased rapidly and now has a major impact on both, developed and developing nations. More than three times the annual deaths of malaria, tuberculosis and HIV/AIDS combined are attributable to cardiovascular disease, which accounts for nearly 30 percent of all deaths in low and middle income countries. In 2008 the number of deaths, caused by cardiovascular disease, has reached a total of 17.3 million globally, which is an increase of 2.9 million compared to 1990 [3]. Although north-western continental European countries show a decrease in coronary heart disease mortality rates, which is according to several studies [4, 5, 6, 7] attributed to the reduction of risk factors, cardiovascular disease still causes 47 percent of all deaths in Europe. According to the American Heart Association, cardiovascular disease

includes high blood pressure, coronary heart disease, heart failure, stroke and congenital cardiovascular defects [8]. It is estimated that cardiovascular disease costs the European Union economy almost € 196 billion a year, which can be divided into direct health care costs (54%), productivity losses (24%) and informal care of people with cardiovascular disease (22%) [9].

1.3.2 Cardiac Arrhythmia

Cardiac arrhythmia is an abnormality in the electrical conduction system of the heart, which can be caused by scarred heart tissue, coronary artery disease, high blood pressure, diabetes, intoxication and medication [10]. According to Scharf et al. [11] arrhythmias can be divided into disorders of impulse generation and disorders of impulse conduction. This chapter focuses on heart diseases which are important in the context of the thesis including bradycardia, tachycardia and heart failure [12].

1.3.2.1 Bradycardia

Bradycardia is defined as a heart rate below 60 beats per minute [11] and includes sinus node dysfunction, atrioventricular block and bundle branch block.

Sinus node dysfunction is an abnormality in the generation of the action potential caused by a dysfunctional sinus node. The disorder of the atrial pulse generation leads to an inappropriate atrial rate, which cannot meet physiological requirements. Sinus node dysfunction occurs at a mean age of 68 years and its incidence increases exponentially with age [13, 14, 15].

The *atrioventricular block* (AV block) is classified in three types. First-degree AV block is defined as an abnormal prolongation of the PR interval (PR-interval >200 ms) – indicating delayed conduction through the AV node. All atrial impulses are still conducted to the ventricles. Second-degree AV block is subdivided into Mobitz Type I and Mobitz Type II. Mobitz I is characterized by a progressive prolongation of the PR interval from one beat to the other, followed by a non-conducted beat. Mobitz II is characterized by blocked atrial impulses between constant PR intervals. Third-degree AV block is defined as a complete block. Hence, no atrial impulses are conducted to the ventricles [13, 16].

A *bundle branch block* exists when the electrical signal cannot be conducted by the branches or fascicles. If the impulses do not travel down both branches at the same speed, one ventricle contracts earlier than the other which leads to a ventricular asynchrony and a corresponding

drop in the cardiac output. There are different types of blocks like left- and right-bundle branch blocks, bifascicular blocks and trifascicular blocks [17].

1.3.2.2 Tachycardia

Tachycardia is defined as a heart rate exceeding 100 beats per minute [11] and includes atrial fibrillation, atrial flutter, supraventricular tachycardia, ventricular tachycardia and ventricular fibrillation [10].

Atrial fibrillation describes the fast and chaotic beating of the atria caused by an uncoordinated electrical signal within the atria. Atrial fibrillation affects mostly people past age 60. The rate within the atria can increase up to 350 to 600 beats per minute, resulting in irregular rates of conduction through the AV node, irregular ventricular rhythms and finally an inappropriate contraction. Recurrent atrial fibrillation can lead to more serious conditions, such as stroke [10, 12].

Atrial flutter shows nearly the same characteristics as atrial fibrillation but atrial depolarisations are more organized and more rhythmic with a frequency of about 220 to 350 beats per minute. Atrial fibrillation and flutter are always caused by an abnormality of the impulse conduction [10, 12].

Supraventricular tachycardia is an arrhythmia originated above the ventricles and it causes a burst of rapid heartbeats at a frequency of about 100 to 240 beats per minute. It can occur spontaneously and can last from seconds to hours [10, 12].

Ventricular tachycardia is caused by abnormal electrical impulses that start in the ventricles and it often occurs in people with a damaged ventricular muscle. Unsustained ventricular tachycardia might not cause any symptoms but can lead to sustained ventricular tachycardia, which can end in ventricular fibrillation and sudden cardiac death.

Ventricular fibrillation is an uncoordinated electrical activity and is often characterized by small amplitudes in the ECG. The ventricles cannot contract efficiently anymore which results in a rapid decrease of the blood pressure leading to an inappropriate blood supply of organs and finally to death.

1.3.2.3 Heart Failure

Heart Failure (HF) occurs when due to an abnormality of the cardiac structure or function, the heart's pumping power is too weak to provide a sufficient amount of blood and consequently there is too little oxygen for the metabolism in tissue. This critical shortage can emerge at rest as well as at physical stress. According to the European Society of Cardiology (ESC), HF is clinically defined as a syndrome in which patients have symptoms like breathlessness, ankle swelling and fatigue, and signs like elevated jugular venous pressure, pulmonary crackles and displaced apex beat [18]. The Framingham Heart Study [19] showed that congestive heart failure (CHF) was the primary cause for hospitalization of people aged ≥65 years in the USA and affected 4.8 million Americans in 2002.

1.4 Cardiovascular Implantable Electronic Devices

Since the first implantation of a fully implantable pacemaker (PM) in 1958, the number and complexity of cardiovascular implantable electronic devices (CIEDs) has rapidly expanded. According to the Heart Rhythm Society (HRS)/ European Heart Rhythm Association (EHRA) consensus [20] PMs, implantable cardioverter defibrillators (ICDs), cardiac resynchronization therapy devices (CRTs), implantable loop recorders (ILRs) and implantable cardiovascular monitors (ICMs) are termed cardiovascular implantable electronic devices (CIEDs). They provide several programmable options and record and store large quantities of diagnostic data comprising the device function, arrhythmia frequency, cardiovascular hemodynamic parameters including transthoracic impedance and patient activity [20, 21]. Table 1 shows the number of CIED implantations in 2006, 2008 and 2011 in Austria and Germany.

	Austria				Germany			
	PM	CRT	ICD	Total CIEDs	PM	CRT	ICD	Total CIEDs
2006	7.306	563	895	8.764	96.906	6.969	15.874	119.749
2008	7.570	235	1.100	8.905	98.300	9.440	21.600	129.340
2011	7.810	958	1.805	10.573	106.953	16.425	26.579	149.957

Table 1: Number of CIED implantations in 2006, 2008 and 2011 in Austria and Germany (PM…Pacemaker, CRT…Cardiac resynchronization therapy, ICD…Implantable cardioverter defibrillator, CIED…Cardiovascular implantable electronic device) [22].

Up to now 26 manufacturers have put about 2000 different PM- and ICD-models on the market. Currently PM-/ICD systems of at least 6 manufacturers are used in Europe and each one provides several models with multiple modalities, programmable features and programmers. There is no programming device which is capable of managing all devices of

different manufacturers [23]. In the context of the current thesis particularly PMs, ICDs and CRTs are relevant and are described in this chapter.

1.4.1 Pacemaker (PM)

A PM consists of a pulse generator and connected leads. The pulse generator contains a long-lasting, tiny lithium-iodide battery and a complex circuitry, which is able to sense, filter, analyse and store intracardiac signals and if required, responds to them by delivering pacing impulses. There are up to three leads, which are electrical conductors and placed in the atrium or ventricle of the heart. They transmit electrical impulses from the pulse generator to the myocardial cells of the heart (pacing) and vice versa (sensing). Single-chamber devices are indicated for atrial or ventricular pacing. Dual-chamber devices sustain synchrony between the atrium and the ventricle. The invasive implantation procedure is a simple surgery which is done under local anaesthesia and takes less than an hour [24, 25, 26].
In 2001 the revised generic code of the North American Society of Pacing and Electrophysiology (NASPE) and the British Pacing and Electrophysiology Group (BPEG) was released. The five-letter code represents the pacing-mode of the device (Table 2) [27]. The first two positions (I & II) indicate the chambers in which pacing and sensing occur. Position III is used to indicate the effect of each instance of sensing on the triggering or inhibition of subsequent pacing stimuli. Position IV shows the presence or absence of an adaptive-rate mechanism. Position V represents the absence or presence of multisite pacing in none, one or both chambers.

A PM interferes in case of detected bradycardia [26]. According to the German Pacemaker Register 2010 [28] sinus node dysfunction accounted for approximately 37% of the implantations of PMs in Germany and third-degree AV block was diagnosed in about 26% of all PM implantations.

Position:	I	II	III	IV	V
Category:	Chamber(s) Paced	Chamber(s) Sensed	Response to Sensing	Rate Modulation	Multisite Pacing
	O = None A = Atrium V = Ventricle D = Dual (A+V)	O = None A = Atrium V = Ventricle D = Dual (A+V)	O = None T = Triggered I = Inhibited D = Dual (T+I)	O = None R = Rate modulation	O = None A = Atrium V = Ventricle D = Dual (A+V)
Manufacturers' designation only:	S = Single (A or V)	S = Single (A or V)			

Table 2: The five-letter revised NASPE/BPEG Generic Code for antibradycardia, adaptive-rate, and multisite pacing is used to represent the pacing mode of the device. Position I and II indicate the pacing and sensing sites. Position III represents the effect on the sensing. Position IV indicates if the adaptive-rate mechanism is present and Position V shows if multisite pacing is used [27].

1.4.2 Implantable Cardioverter-Defibrillator (ICD)

An ICD is a device which, like a PM, is implanted under the skin. It consists of an aggregate, which is connected to electrodes. The aggregate contains the battery, energy delivery components and electronic circuitry. The electrodes are placed in the veins and connect the aggregate to the heart. All modern ICDs are combinations of defibrillators and PMs. Compared to a PM, an ICD can additionally deliver higher energy electrical pulses (shocks) between 10 J and 40 J to the heart [29].

This is especially important for patients who have suffered or are at risk of suffering from serious rapid and sustained arrhythmias like ventricular fibrillation, ventricular tachycardia and/or atrial fibrillation, which is not correctable by low-energy electrical pulses from a PM. ICDs are the most effective instruments for prolonging the survival of patients with tachycardia and they are even superior to antiarrhythmic drugs [29]. ICDs are indicated in primary and secondary prevention of sudden cardiac death. Primary prevention is required for individuals who are at risk of an episode of sustained ventricular tachycardia or ventricular fibrillation but not yet had one. Secondary prevention is related to patients who have survived a prior sudden cardiac arrest or sustained ventricular tachycardia [13].

1.4.3 Cardiac Resynchronization Therapy (CRT)

CRT devices are much like PMs except that they use at least two leads, one in each ventricle. Some patients also get a lead implanted into the right atrium. CRT is indicated in case of symptomatic, drug-refractory heart failure. By altering the activation sequence in the paced chambers, regional contractility and central hemodynamics can be influenced. The relatively new therapy showed a decreased morbidity and mortality in the CARE-HF study [30] and the COMPANION study [31]. This is achieved by resynchronizing the timing of global left ventricular depolarisation which is accomplished by pacing the left and right ventricle. As a result mechanical contractility and mitral regurgitation are improved [32, 33].

1.5 Follow-ups

After diagnosis of the underlying heart disease and implantation of a CIED, regular follow-ups to check settings and stored diagnostic data are crucial. As already pointed out in Chapter 1.4 the frequency and complexity of implanted systems is steadily growing. Hence the number of follow-up encounters is also rapidly increasing. Table 3 gives an overview of the total numbers of follow-up encounters per year in 2007 [20].

	Pacemaker therapies	ICD therapies
North America	1.610.000	2.065.000
Europe	1.680.000	500.000
Total numbers	3.290.000	2.565.000

Table 3: Number of PM and ICD follow-up encounters per year in North America and Europe in 2007.

This chapter represents an overview of the goals, types and frequency of follow-ups. It is mainly based on literature including [20], [34] and [35].

1.5.1 Goals of Follow-ups

According to the HRS/EHRA [20] the major goals of monitoring programs can be divided into the following groups.

Patient related goals include optimization of the quality of life, adjustment of the device in the best possible way in order to match the patient's clinical requirements and detection of patients at risk to initiate appropriate follow-ups.

CIED related goals include documentation of the appropriate function and detection as well as correction of malfunctions. Furthermore they comprise achievement of high longevity of the device's battery, while still maintaining patient safety and monitoring of parameter values (e.g. end of battery life).

Disease related goals include documentation of arrhythmia type and frequency to correlate them with patient's symptoms and to ensure a correct device response. Additionally documentation and analysis of all relevant parameters like the hemodynamic status, the transthoracic impedance, the patient activity and other physiological parameters are important.

Communication goals include forwarding of important CIED- and disease-related information to the patient and to relevant health care providers in sufficient time to maintain patient safety. Moreover colleagues, patients and community should be provided with technical expertise and education.

1.5.2 Follow-up Types

In general follow-up types can be divided into two groups: face-to-face in-clinic visits and remote follow-ups (including remote monitoring systems and remote device interrogation). While most remote device interrogation systems only support therapy management based on CIED data, current remote monitoring systems focus on data acquired by external devices (e.g. vital signs). Systems that support both kinds of data can be rarely found (see Chapter 1.7 for details).
The current CIED-therapy management in Austria provides in-clinic follow-ups in specialized cardiological departments in combination with remote device interrogations.

In-clinic follow-up: Traditionally the monitoring of CIEDs is performed by trained physicians in specialized cardiological departments. The data are transferred from the CIED to a manufacturer specific device (programmer) via a bidirectional telemetry connection. This is achieved by a programming 'wand' which is connected to the programmer and placed above the implanted device. The programmer enables the physician to display and check the transmitted information and to adjust parameters of the CIED. Usually programmers provide additional interfaces for printers, storage devices and communication devices.
Furthermore in-clinic follow-ups can be subdivided into two types [35]:

- Basic follow-ups comprise sensing, pacing, battery and patient contentedness.
- Extended follow-ups include additional checks and optimization of pacemaker parameters.

Remote device interrogation: The primary telemedical concept is called transtelephonic monitoring (TTM) and it was widely used in the US and Canada. In its initial design the patient had to record an ECG and transmit the data to a service centre periodically. Meanwhile relevant data are transmitted automatically from the CIED to a transmitter in the patient's vicinity, from where it is forwarded to a telemedical service centre. In the centre a specialist analyses the data and just in case of abnormalities the patient is asked for an in-clinic follow-up interrogation. The current generation of devices is able to transfer diagnostic, therapeutic and technical information automatically to an external transmitter (aggregator), which forwards them to the provider's data repository. In the monitoring centre pre-processing like filtering and analysing is conducted. Then the physician gets access via a secured internet platform where the information can be reviewed. The whole data sequence is illustrated in Figure 2.

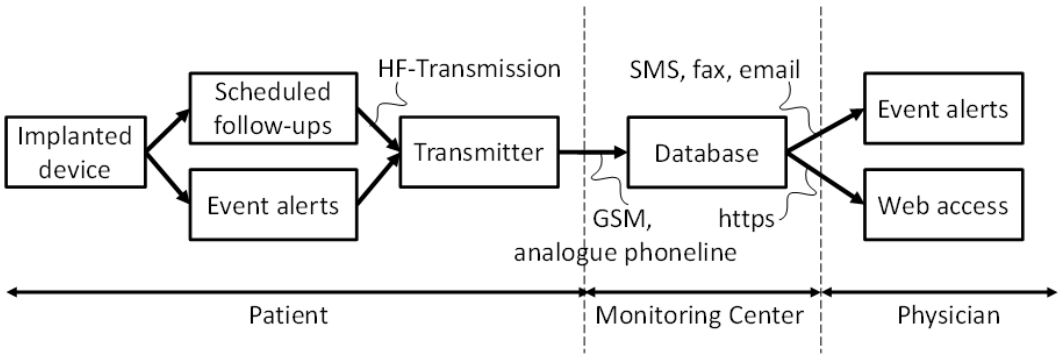

Figure 2: Data flow in the remote device interrogation system.
The implanted device regularly sends data to a transmitter, which then forwards them to a data repository in the monitoring centre. After pre-processing the physician can access the data [35].

Remote monitoring systems for chronic diseases: Patients with chronic diseases such as diabetes or adiposity, and in particular heart failure, are in need of enhanced treatment. In remote monitoring systems patients have to collect information according to their therapy, e.g. medication, blood pressure, body weight and well-being. These data are transmitted to a monitoring centre by an aggregator like a mobile phone, PC or laptop. In the monitoring centre data are pre-processed, analysed and accessible for specialists (see Figure 3). If the patient's condition gets worse, the physician can react in due time and optimize the treatment.

In 2010 a Cochrane review [36] showed that remote monitoring concepts, which supported the measurement of vital signs with external devices, are valuable tools to optimize the therapy of patients with heart failure. The evaluation of twenty-five studies pointed out that telemonitoring reduced all-cause mortality and congestive-heart-failure-related hospitalization.

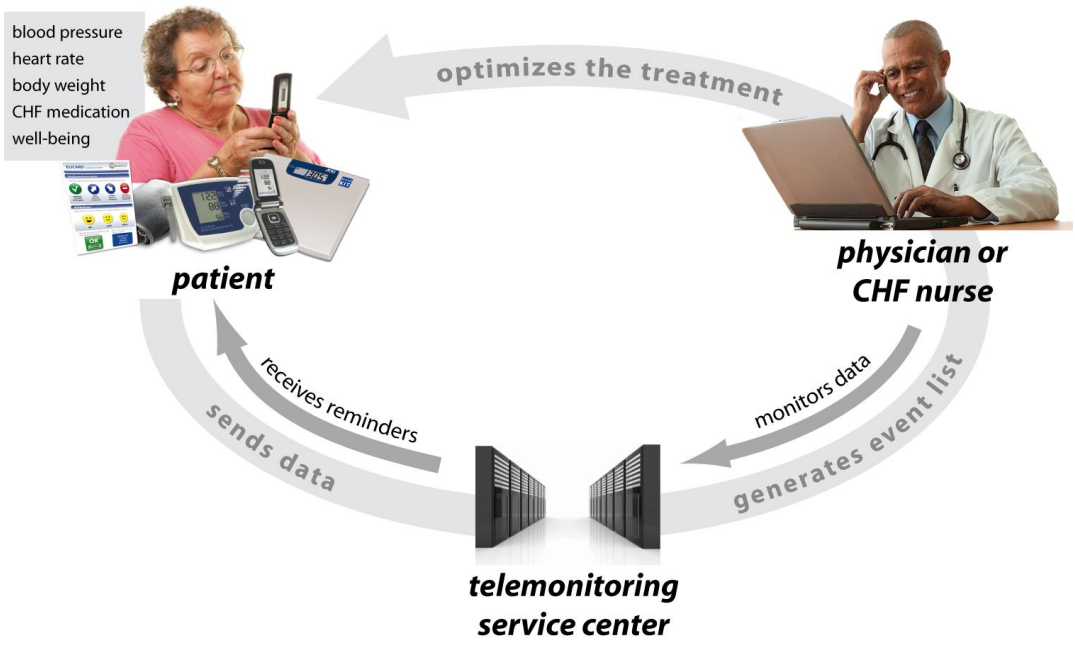

Figure 3: Remote monitoring system for chronic diseases.
The patient collects data from various sources (e.g. blood pressure, heart rate) and transmits them to a telemonitoring service centre. After data are pre-processed and analysed they are accessible to the physician. In case of a negative development the physician can optimize the treatment [35].

1.5.3 Follow-up Frequency

There are large differences in the frequency of device follow-ups worldwide. Basically type and frequency of follow-ups essentially depend on the individual medical condition of the patient. In case of unstable or frequently changing cardiovascular status, an in-clinic follow-up may be required and in case of a stable status and no necessity of device reprogramming, remote monitoring is preferred. Table 4 summarizes the Expert Consensus on the Monitoring of CIEDs defined by the Heart Rhythm Society and the European Heart Rhythm Association [20]. It shows the minimal recommended schedule of device follow-ups.

Device(s)	Frequency	In-Clinic	Remote
PM/ICD/CRT	Within 72 hours of CIED implantation	X	
	2 – 12 weeks post implantation	X	
	Every 3 – 12 months PM/CRT-P	X	X
	Every 3 – 6 months ICD/CRT-D	X	X
	Annually until battery depletion	X	
	Every 1 – 3 months at signs of battery depletion	X	X
ILR	Every 1 – 6 months depending on patient symptoms and indication	X	X
Implantable hemodynamic monitor	Every 1 – 6 months depending on indication	X	X
	More frequent assessment as clinically indicated	X	X

Table 4: Minimal recommended schedule of in-clinic or remotely monitored device follow-ups [20].

1.6 Platform for Integrated Cardiac Rhythm Disease Management (PICARD)

As already mentioned in previous chapters the prevalence of cardiovascular disease and the number of indications for CIEDs are growing. Therefore the number of follow-up encounters is steadily increasing and becoming a serious matter in terms of time and costs. Furthermore factors like new diagnostic options, technological developments in ICD-/CRT-systems and the trend towards automated aftercare requires reorganization of the management in terms of patient safety and follow-up appointments for CIED patients [21].
Based on research done at AIT Austrian Institute of Technology a web-based Platform for Integrated CArdiac Rhythm Disease management (PICARD) (AIT Austrian Institute of Technology, Graz, Austria) was developed. The first clinical trials of the project started in 2002 and a clinical pilot study was conducted in 2005. The results of the study showed that information and communication technology combined with telemedical systems facilitated basic follow-ups for patients and physicians [37].

1.6.1 Functionality and Structure

The objective of the project was to build up a central management platform, capable of handling relevant data from various sources (see Chapter 1.5.2) and providing necessary information for the physician and an optimal therapy of the patient. As Figure 4 shows, the system featured interfaces to a programmer device of one manufacturer, one telemedical follow-up system, the electronic patient record (EPR), the web-portal for physicians and the

electronic health record (EHR). Some of the interfaces were established prototypically, others were used in clinical trials.

An additional important aspect was to disburden clinicians concerning the increasing amount of follow-up encounters. The system offered the possibility to move basic follow-ups from specialized clinics to extramural institutions like general practitioners or arbitrary hospitals while providing an optimal therapy [35, 38, 26]. Figure 4 shows the functionality of the heterogeneous systems combined in PICARD.

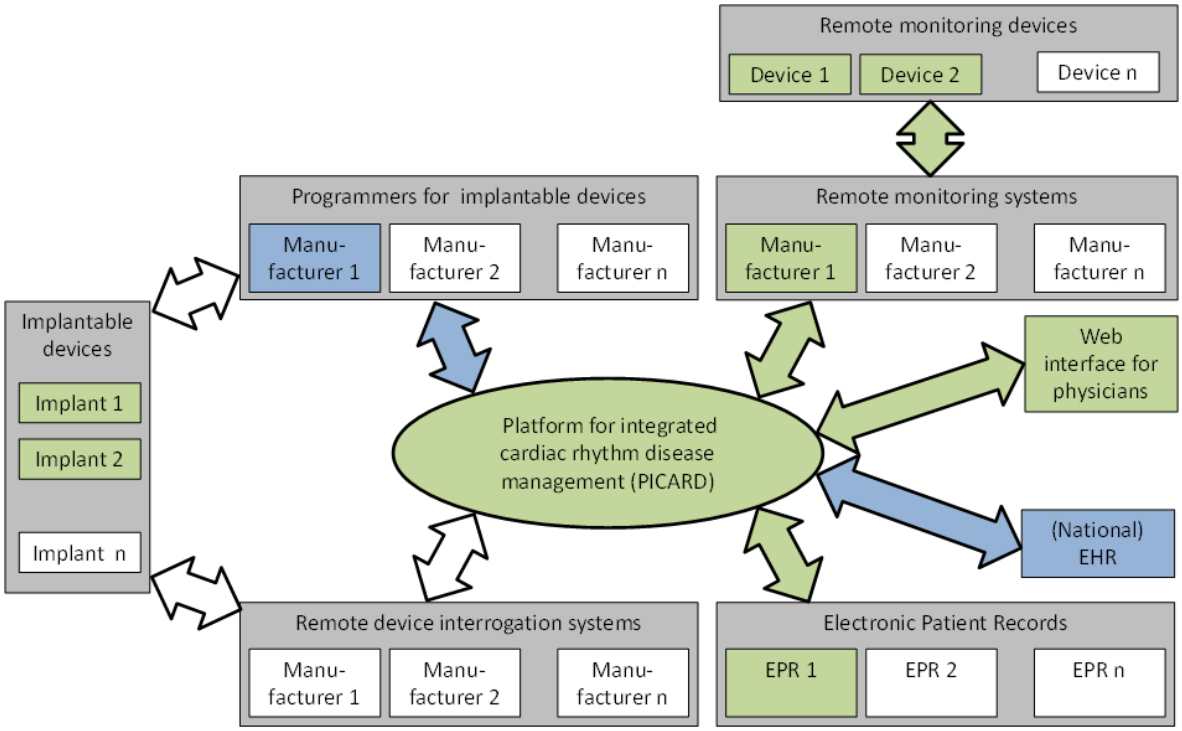

Figure 4: PICARD and its interfaces to a programmer, remote monitoring system, web-portal, EHR and EPR. Already implemented interfaces are green. Prototypically implemented interfaces are blue. Not implemented interfaces are white [35].

1.6.2 Standardized Programmer Interface (SPRINT)

The Standardized PRogrammer INTerface (SPRINT) was developed within the scope of PICARD and represented an interoperability framework. The main purpose was the automated processing and standardization of proprietary data formats from programmers of different manufacturers.

It was implemented as an interface between the programmer ICS 3000 (Biotronik, Berlin, Germany) and the database of PICARD. The Java-application was capable of receiving data from the programmer, converting them into a standardized form and sending them directly to the database of the system.

The application could process two proprietary XML (World Wide Web Consortium, Massachusetts, USA) data formats (CARDDAS, Paceart). The files were internally mapped in a standardized object, which was directly transmitted to the server. SPRINT could read data from an RS232 interface or a file system and send them to the server by using one of three supported protocols (MLLP, SOAP, HTTP). Figure 5 illustrates the data flow and process.

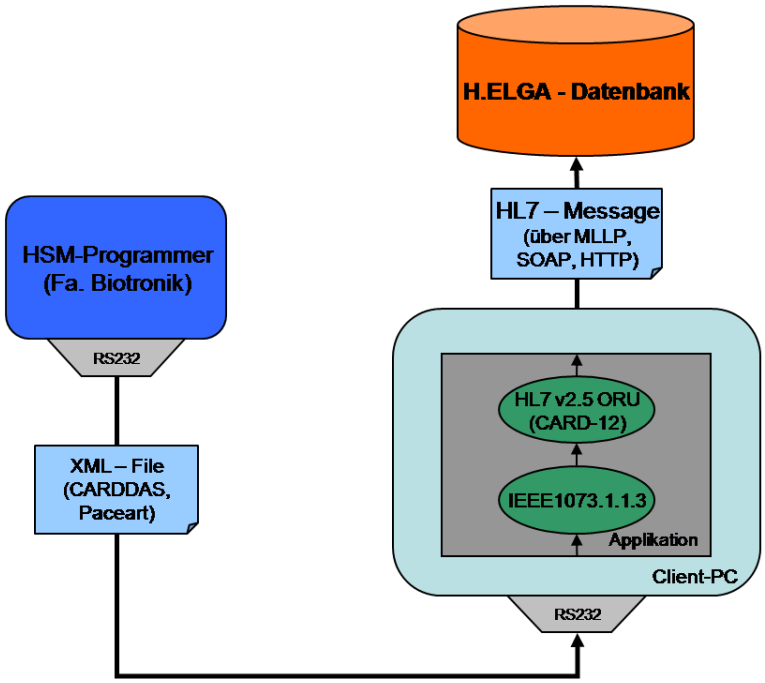

Figure 5: Schematic flow diagram of PICARD 1.0.
Data in XML files are transmitted to the client computer via a serial interface (RS232). Then SPRINT converted the proprietary data structure into a standardized format and passed it on to a database in a standardized message (HL7). Taken from AIT's internal project page - SPRINT.

SPRINT represented a prototype showing that data can be transferred automatically and standardized from the programmer to a superior system.

1.6.3 PICARD 2.0

In 2011 the development of a follow-up project called PICARD 2.0 started. The main goals included conceptual design, adaptation, system setup and maintenance of the primary PICARD for routine care as a clinical and research system operated by KAGes[1]. So far data of each patient were printed out and stored as hard copy in paper based patient records. PICARD 2.0 provided enhanced data management, sophisticated data processing, various interfaces and avoidance of problems occurring in paper based data storage systems.

[1] Steiermärkische Krankenanstaltengesellschaft m.b.H. (The major hospital operation organization in Styria)

In the first step the platform consisted of interfaces to a CIED database, a web portal and the hospital information system (HIS). Interfaces to programmers, telemedical follow-up systems and home monitoring systems existed in the conceptual design but were not yet implemented (Figure 6) [39].

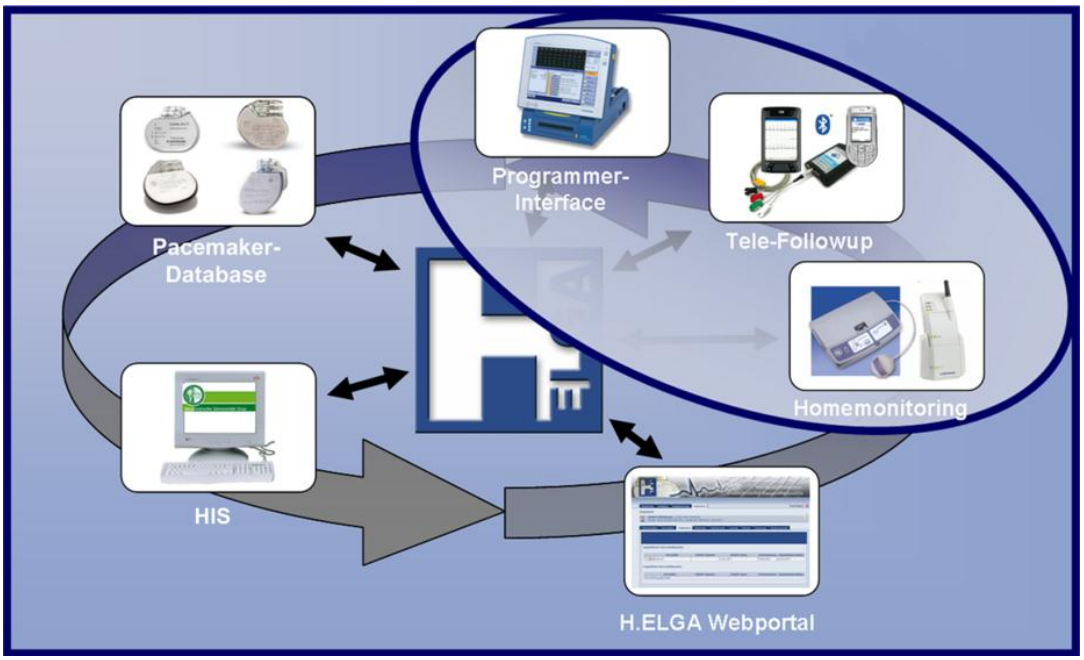

Figure 6: Conceptual design of PICARD 2.0.
PICARD's interfaces to a pacemaker database, the hospital information system (HIS) and the web-portal. Interfaces to programmers, telemedical follow-up systems and home monitoring systems were in the design but not yet implemented [39].

1.7 Commercially Available Products

The Paceart (Medtronic Inc., Minnesota, USA) system represents a central repository, which organizes and archives patient's arrhythmia information of cardiovascular implantable electronic devices of various manufacturers. The system operates as a gateway from programmers and remote monitoring systems to a clinic's electronic health record (EHR) system. It stores programmed device parameters, summarizes patient session data, schedules follow-ups and assists in the charge and billing management [40, 41].

Nearly every manufacturer provides a home monitoring system, such as CareLink Network (Medtronic Inc., MN, USA), Home Monitoring (Biotronik, Berlin, Germany), Latitude Patient Management System (Boston Scientific, St. Paul, USA) and Merlin.net Patient Care Network (St. Jude Medical, Sylmar, USA). While Boston Scientific's system facilitates the

transmission of additional information (e.g. blood pressure, weight), other manufacturers only support CIED data.

Compared to PICARD and SPRINT, these solutions have some drawbacks including the fact, that they are not easily adaptable to import data from other vendors and the systems might not be used by other manufacturers because they were developed by and under the control of a competitor.

1.8 Objectives

Due to the fact that most manufacturers provide a proprietary programmer interface (e.g. via USB, serial port, Bluetooth) and a proprietary data format to store the recorded data and parameter values, the import of data into PICARD 2.0 has been carried out manually up to now. Therefore the user has had to enter all requested data manually in the web portal, which is very time-consuming and carries the risk of inconsistency.

The main goal of the thesis was to advance this situation towards an automated import of relevant parameters and thereby build the basis for reliable further processing and standardized data exchange to other system components.

The main requirements included:

- Design of an interface capable of importing data from programmers of different manufacturers into PICARD 2.0 based on the formerly prototypically implemented Standardized Programmer Interface 1.0 (SPRINT 1.0).
- The user must still have the possibility to add, modify and delete data manually.
- Determination of the hardware interface for the data exchange.
- Research into standardizations and interoperability frameworks for data exchange.
- Specification of the workflow.
- Development and integration into the workflow architecture of PICARD 2.0 – considering its on-going development and changing requirements
- Testing against the prototypically implemented software and evaluation in a real-life scenario.

Chapter 2

Materials and Methods

The following chapter describes used materials including standards for the data exchange, devices (programmers) and the development environment with the used programming languages.

2.1 Standards

2.1.1 Organisations

Integrating the Healthcare Enterprise (IHE) (Brussels, Belgium) is an initiative by healthcare professionals and industry to improve the exchange of information among healthcare systems. The goal of the IHE working groups is not to define new integration standards, but rather to define technical frameworks for already existing standards that address particular clinical areas. The precise description of how these standards are to be implemented under specific clinical conditions is termed IHE Profile [42].

The International Organization for Standardization (ISO) (Geneva, Switzerland) is the world's largest developer of voluntary international standards, which cover almost all aspects of technology and business [43].

The Institute of Electrical and Electronics Engineers (IEEE) (New York City, USA) is a worldwide professional association of engineers with the goal to advance technological innovation. One important part of IEEE is the development of standards in a broad range of industries [44].

The Bluetooth Special Interest Group (SIG) (Washington, USA) is a privately held, not-for-profit trade association which publishes specifications, administers the qualification program, protects the Bluetooth trademark and evangelizes the technology. It is neither a manufacturer

of Bluetooth enabled products nor a seller of any devices. Companies from various different industries, for example telecommunications, computing, industrial automation, medical and health industries, are members of the Special Interest Group [45].

2.1.2 ISO/IEEE 11073-10103:2012 Standard

The ISO/IEEE 11073 is a family of joint standards for point-of-care (POC) medical device communication (MCD) and represents a new industrywide profile to the IHE Patient Care Device Technical Framework [42]. The standards enable communication between medical devices and external computer systems. They provide automatic and detailed electronic data capture of patient vital signs information and operational data of the device. The main goals are to offer real-time plug-and-play interoperability for patient-connected medical devices and to facilitate the efficient exchange of vital signs and medical device data in all health care environments [46].

The ISO/IEEE 11073-10103 standard is an extension of the base nomenclature provided in ISO/IEEE 11073-10101 and supports the terminology for implantable cardiac devices listed in Chapter 1.4. The nomenclature defines necessary terms for summarizing relevant data including settings and measurements of the device, obtained during the interrogation. This Implantable Device, Cardiac (IDC) Nomenclature provides the basis for the standardized management of summarized interrogation information from all vendor devices and systems in a central system such as an Electronic Health Record system. IEEE 11073-10103 IDC nomenclature could be used in various cases like the integration of in-clinic follow-up data or the information retrieved from home monitoring systems. Figure 7 illustrates some examples where the nomenclature could be used [47].

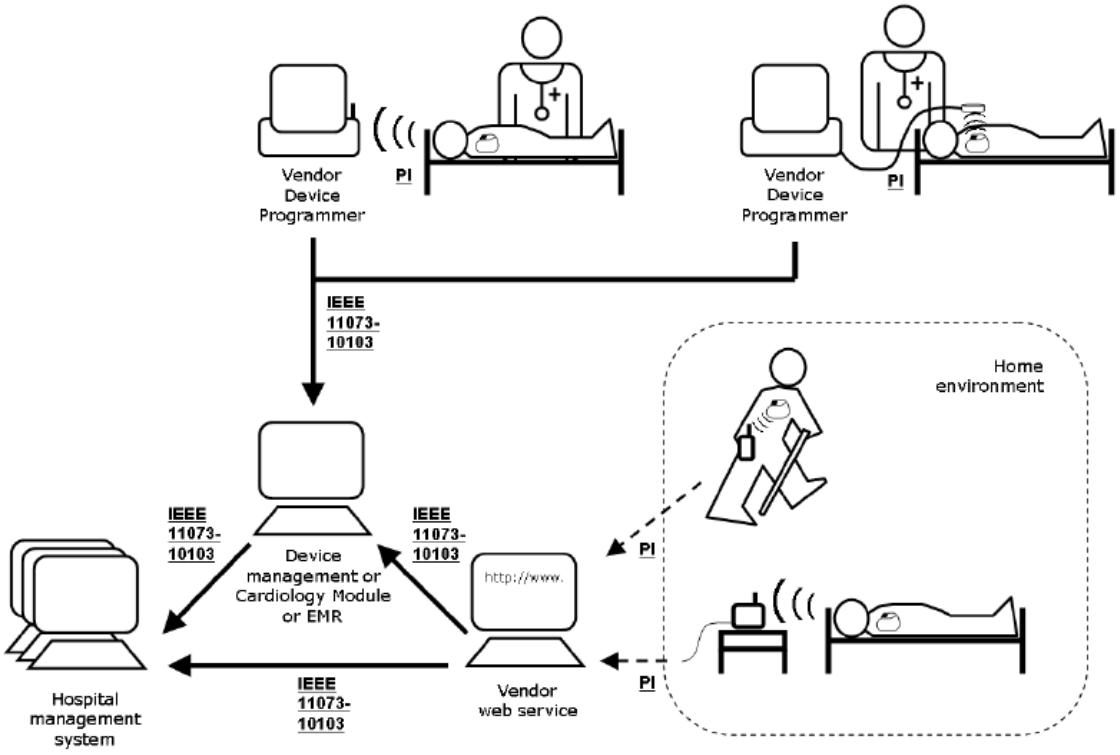

Figure 7: Field of application for the IEEE 11073-10103 IDC nomenclature.
Examples include the exchange of data between programmers, cardiac rhythm management systems, hospital information systems and vendor-specific web-portals (PI…Proprietary interface, EMR…Electronic medical record) [47].

The nomenclature is structured based on a containment hierarchy. Figure 8 shows the semantics of the terminology in a tree graph. The root node is MDC_IDC, which is short for Medical Device Communication – Implantable Device Cardiac. Discriminators are used to manage additional semantic refinement. They are represented by square brackets and a text token [Token], e.g. the node LEADCHNL has the discriminator [CHAMBER] with a set of heart chambers ([RA], [RV], [LA], [LV]) resulting in LEADCHNL_RA, LEADCHNL_RV, LEADCHNL_LA and LEADCHNL_LV. Terms are uniquely identified by a Reference ID, Systematic Name and Code according to ISO/IEEE 11073-10101:2004. The Reference ID consists of a sequential path through the containment hierarchy from the root node to a leaf node. Each node on this path becomes a component of the term's Reference ID and Systematic Name e.g. MDC_IDC_MSMT_LEADCHNL_[CHAMBER]_PACING_THRESHOLD. After expanding all discriminators the term gets a unique code from a 16-bit code block which is exclusively used of the IDC nomenclature. The nodes were determined through interaction with clinical domain experts. The highest hierarchy level contains the nodes Implantable Cardiac Device (DEV), Implantable Lead (LEAD), Interrogation Session (SESS), Measurement (MSMT),

Setting (SET), Statistic (STAT) and Episode (EPISODE) [47]. The detailed description of all terms can be found on the IEEE homepage [44].

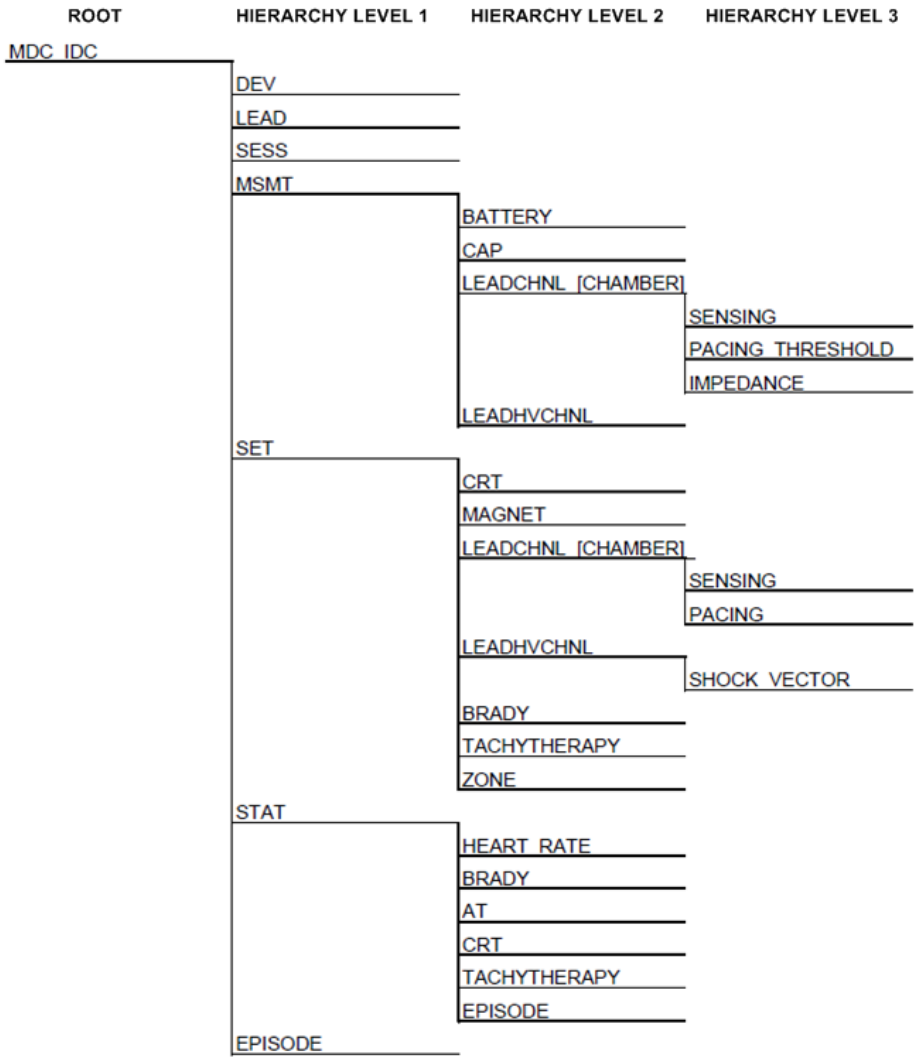

Figure 8: Semantical structure of the 11073-10103 terminology.
The root contains the implantable cardiac device (DEV), implantable lead (LEAD), session (SESS), measurements (MSMT), settings (SET), statistics (STAT) and episodes (EPISODE). Higher hierarchy levels include more specific parameters [47].

2.1.3 Retrieve Form for Data Capture

Retrieve Form for Data Capture represents an IHE Profile, which is part of the IT Infrastructure Technical Framework. The profile provides a method to collect data within a user's current application for various secondary external systems, e.g. Clinical Research, Disease Registries [48]. A data capture form can be retrieved from a form source (e.g. external agency), filled out by a home application (e.g. an electronic health record) and returned to a receiving application (e.g. external agency). The RFD Profile uses XForms

(World Wide Web Consortium, Massachusetts, USA) technology because these XML powered forms are platform independent, standards-based (World Wide Web Consortium specification) and gain support in various browsers [49].

There are four actors directly involved in the Retrieve Form for Data Capture Integration Profile. Figure 9 shows the transactions between them. The Form Manager receives form retrieval requests and responds to them by returning the appropriate forms. Based on the context information in the retrieval request, the form is selected from a set of predefined forms, a predefined form is modified or a new form is constructed. The Form Filler retrieves forms from the Form Manager, fills it with available data and sends it to the Receiver. The Form Receiver receives the completed or partially completed form data and triggers the further processing, which isn't part of the profile anymore. The Form Archiver receives the same data and stores them [49].

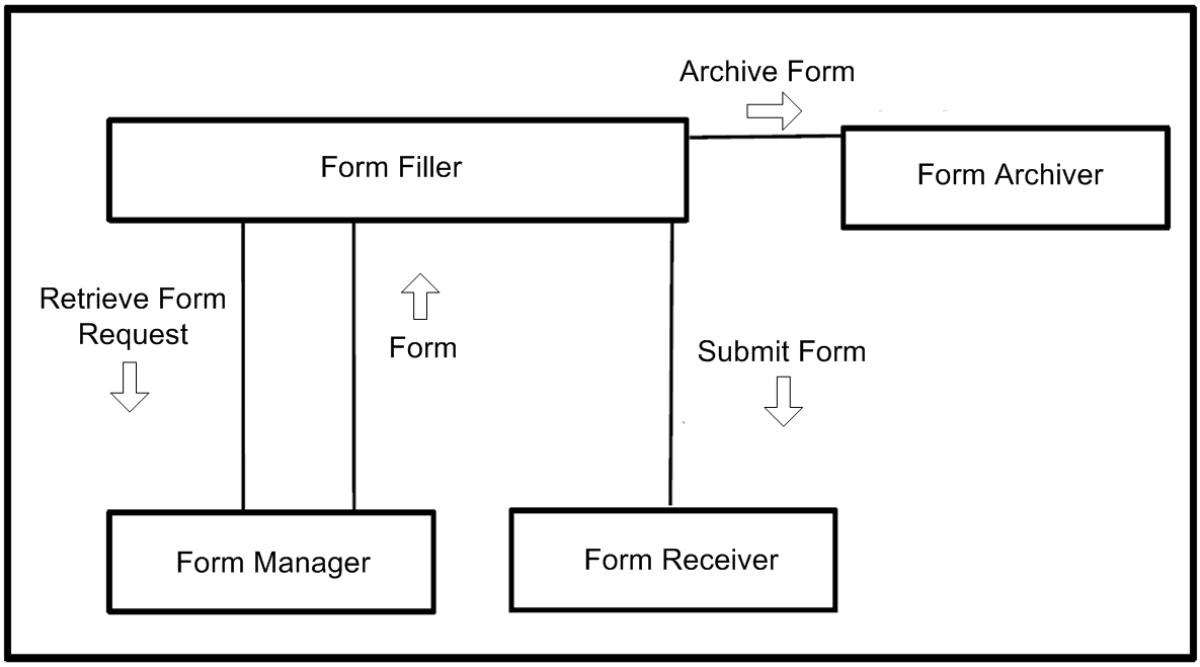

Figure 9: Directly involved actors in the Retrieve Form for Data Capture Integration Profile. The Form Filler sends a Retrieve Form Request to the Form Manager and gets the appropriate form returned. After processing the form is sent to the Form Receiver and the Form Archiver. Adapted from [49].

2.1.4 Bluetooth

The Bluetooth wireless technology, developed by the Bluetooth Special Interest Group, facilitates a short-range connection (personal area network) between Bluetooth enabled electronic devices. It operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz and adaptive frequency hopping reduces interferences between wireless

technologies in this spectrum. The minimum range is application-specific and varies from up to 1 meter to 100 meters depending on the used radio class. The most commonly used radio is Class 2 and uses 2.5 mW of power. There is also Bluetooth low energy technology available which consumes between 1/2 and 1/100 of the power of classic Bluetooth technology. The technology offers various ways to ensure a high level of security which includes authentication while pairing, encryption of data and signing data for authentication without encryption.

2.2 Programmer

As already mentioned in previous chapters, a programmer is a device which establishes a bidirectional telemetry connection to a CIED allowing the user to interrogate data and adjust the device's parameter settings [20]. Within the scope of the current thesis, Biotronik's latest model Renamic (Biotronik, Berlin, Germany) and Boston Scientific's ZOOM® LATITUDE® (Boston Scientific, St. Paul, USA) are supported. These two devices will coarsely be described in the following sections. Figure 10 shows an overview of the programmers from Medtronik, Biotronik, St. Jude and Boston Scientific.

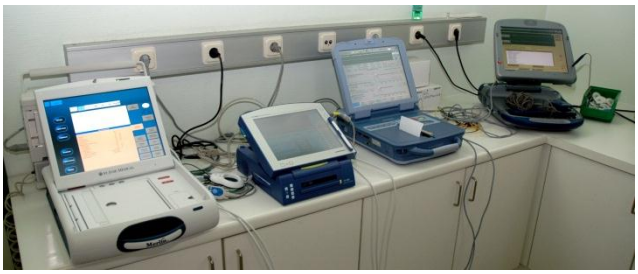

Figure 10: Programmers from Medtronic, Biotronik, St. Jude and Boston Scientific at the pacemaker outpatient ward at LKH University Hospital Graz [35].

2.2.1 Biotronik

In 2011 Biotronik launched their latest cardiac device programmer Renamic, which is shown in Figure 11.

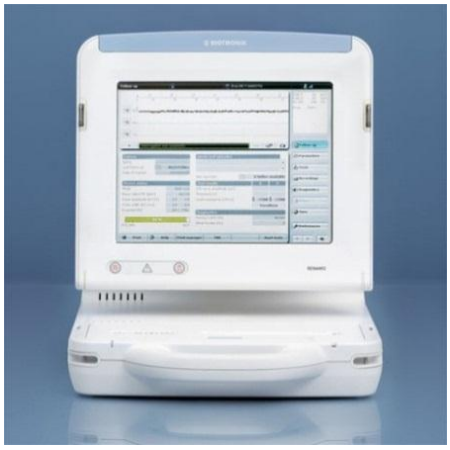

Figure 11: Biotronik's cardiac device programmer Renamic [54].

It is the smallest portable programming and monitoring device and provides the following main functionality which allows the user to [50]:

- Interrogate, save, display and print program parameters, recorded statistical data and episodes.
- Export follow-up reports via USB/Bluetooth and print them out by the internal or an external printer.
- Program Biotronik PMs and ICDs during implantation procedure or follow-up and conduct sensing/pacing thresholds and impedance tests.
- Display up to three leads of surface ECGs with the ECG recorder and ECG monitor.
- Store parameter values and ECG recordings for further computer based archiving and evaluation.

Additional features like Bluetooth, Global System for Mobile Communications (GSM) and wireless LAN (Wi-Fi) offer various ways of data transmission. The device is also prepared for Biotronik SafeSync (Biotronik, Berlin, Germany), which enables a wireless (without wand) interrogation of PMs and ICDs [51].

2.2.1.1 Biotronik Export and File Formats

The programmer supports the following data formats in the data export settings [52]:

- Backup: Encrypted file format, which can be used to import backup data on the models Renamic or ICS 3000.
- Paceart XML: Export format which is compatible to Medtronic's Paceart® System.
- Biotronik XML: An open, freely available proprietary format based on the XML standard. One XML file is created for each follow-up. Optionally the XML file and additionally a PDF document can be exported. In the current thesis a Biotronik XML file was exported and used for further processing (see Chapter 2.2.1.2).

Biotronik EHR DataSync® (see Chapter 2.2.1.2) facilitates the conversion from the Biotronik XML format into a standardized Biotronik 11073-10103 XML format, whose content is based on the IEEE 11073-10103:2012 standard, as already described in Chapter 2.1.1.

The Biotronik 11073-10103:2012 XML Structure contains all possible values/parameters supported by Biotronik implantable cardiac devices and embeds additional Biotronik specific elements (e.g. a Biotronik Status Report and Biotronik Episode Reports in PDF file format), separated from the Biotronik 11073-10103 XML Structure. Figure 12 gives an overview of the XML elements and their relationship.
The root element "biotronik-ieee11073-export" comprises the entire export content. It occurs only once and can basically have one or more child elements of type "dataset", but in the current version only one "dataset" is supported. It represents a collection for data belonging to the same event hence it contains all data for one interrogation. It has at least one child element of type "section". The "section" element represents a group of parameters and can contain 1 – n other "section" elements and 0 – n "value" elements. For example the "section" element "LEADCHNL" can contain "section" elements like "SENSING" or "IMPEDANCE". The element "value" represents a parameter value and can optionally include one "escape" element, which contains additional information e.g. in case a value is out of the measurement range [53].

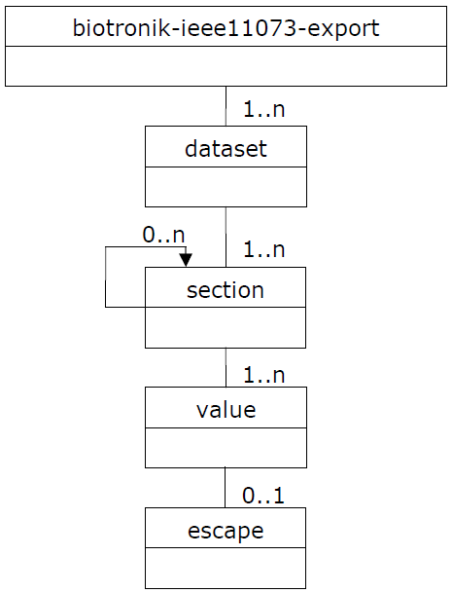

Figure 12: Biotronik 11073-10103 XML elements and their structure.
The root element contains at least one dataset element, which comprises at least one section element. The section element can include other section elements, but has to include at least one value element. The escape element is optional [53].

Figure 13 shows the content structure of the Biotronik IEEE 11073-10103 XML file. The entire content is enclosed by the "biotronik-ieee11073-export" and the "dataset" element. The main content is subdivided into the two "section" elements "MDC" and "BIO". "MDC" contains values for parameters and parameter groups, defined in IEEE 11073 and is split up in two further "section" elements named "ATTR" and "IDC". "ATTR" contains patient information from IEEE 11073-10101 and "IDC" includes parameters from IEEE 11073-10103 (see Chapter 2.2.1). "BIO" is from type "section" and comprises all Biotronik specific parameters and parameter groups that are not supported by the IEEE standards.
An extract of a sample file in BIO 11073-10103 XML format can be found in the Appendix (Figure 28).

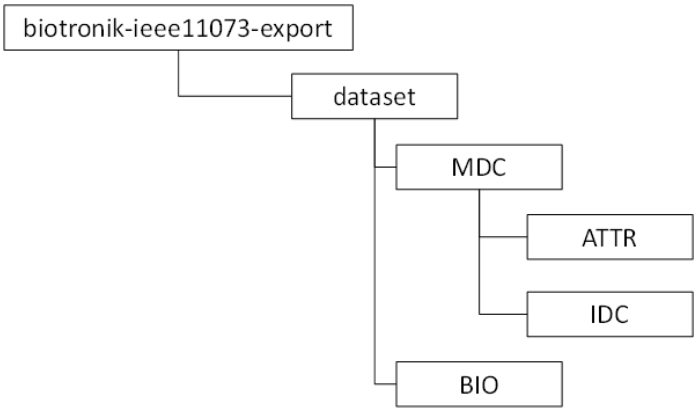

Figure 13: Content structure of the Biotronik IEEE 11073-10103 XML standard.
The MDC element contains parameters defined in IEEE 11073, particularly IEEE 11073-10101 in ATTR and IEEE 11073-10103 in IDC. BIO contains Biotronik specific data.

2.2.1.2 Biotronik EHR DataSync®

Biotronik EHR DataSync® (Biotronik, Berlin, Germany) is a data and software support tool which is capable of transferring the most important device parameters from Biotronik's Home Monitoring system or the programmer into the hospital electronic health record system. Therefore data are exported as Biotronik XML files using a USB drive, a CD or a Bluetooth connection. Additionally PDF files can be generated and transferred. Once the data are transmitted, they are converted into the standardized Biotronik 11073-10103 XML data format using a Biotronik software adapter ("Programmer adapter"). The adapter can be controlled by the user interface (Figure 14) or by setting up a console statement whose basic structure looks like `"BioExportIEEE.exe <Input file name with path> <Output file name with path>"`.

This software package is freely available on the Biotronik homepage [54] and contains the necessary software and interface to extract, transform and translate the data, hence facilitate an immediate and seamless export into the hospital EHR system [55]. Figure 15 shows the data flow from the data source (1) to the software adapter (2) and finally into the electronic health record system (3), which can be accessed by the physician.

Within the scope of this thesis Biotronik EHR DataSync® with its software adapter was used to export Biotronik XML files and convert them into the standardized BIO 11073-10103 XML format (see Chapter 2.2.1.1).

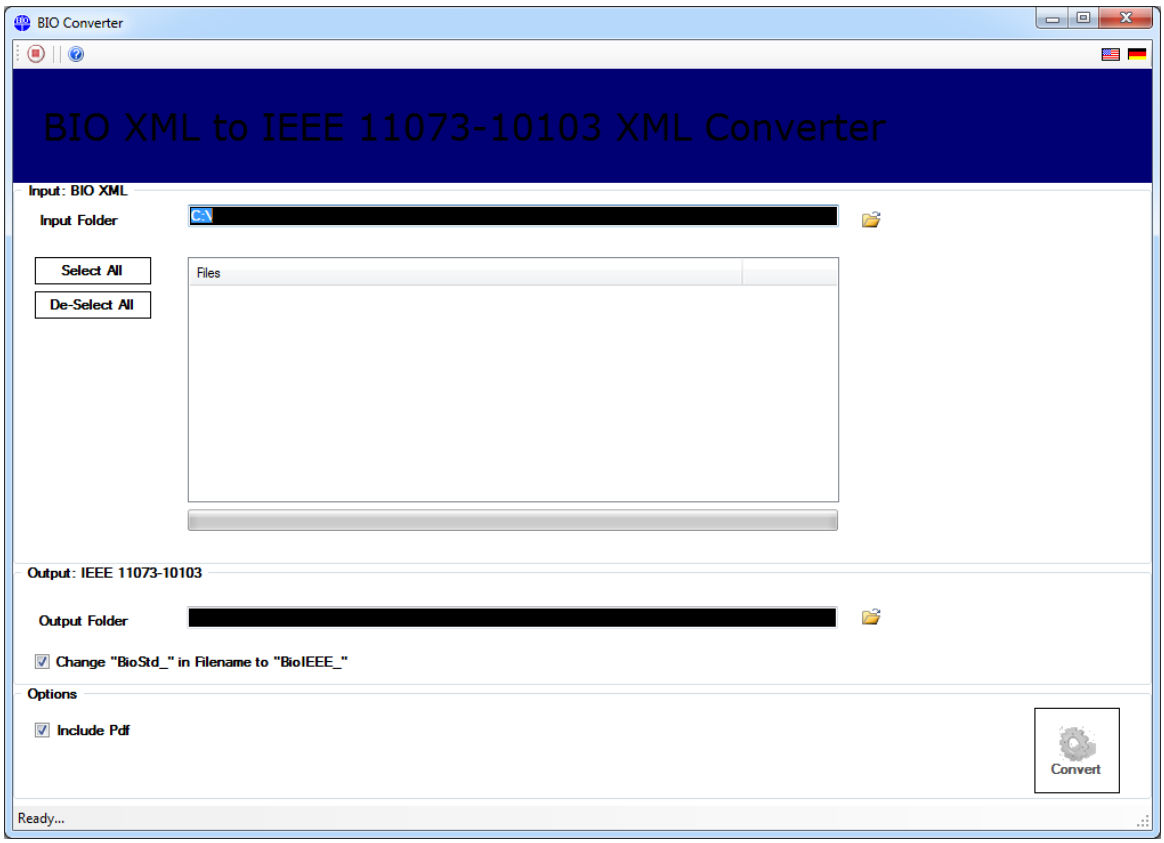

Figure 14: Graphical User Interface of the Biotronik Converter [54].

Figure 15: Data flow of Biotronik EHR DataSync®.
The acquired data are exported from the home monitoring system or programmer (1). Then they are converted with the Biotronik Adapter (2) and imported in the EHR system of the hospital (3) [54].

2.2.2 Boston Scientific

The Model 3120 ZOOM® Latitude® (Boston Scientific, St. Paul, USA) (Figure 16) is a portable system for programming, recording and monitoring Boston Scientific implantable pulse generators.

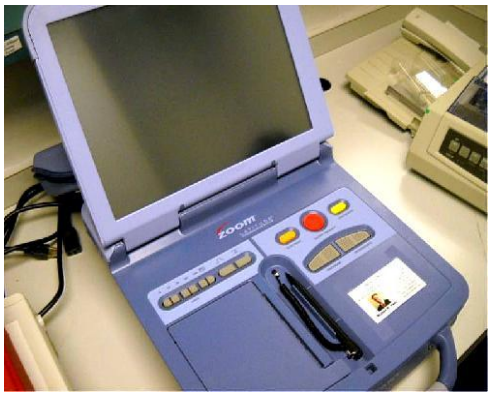

Figure 16: Boston Scientific Model 3120 ZOOM® Latitude® [56].

The Programmer/Recorder/Monitor (PRM) provides the following functionality [57, 58]:

- Interrogation and programming of the implanted device.
- Displaying, recording and storing of patient data.
- Storage on the internal hard drive or using the floppy disk drive.
- Creation of reports.
- Printing detailed patient, device or test information.
- Recording of a surface electrocardiogram.
- Export of patient data and reports in PDF format on a USB drive.

Optionally the ZOOM® LATITUDE® can be equipped with an external printer, monitor and ECG recorder.

2.2.2.1 Boston Scientific Export and File Formats

Patient and device data, stored on the internal memory of the implanted device, can be interrogated and downloaded to a specific Model 6627 Patient Data Disk (Boston Scientific, St. Paul, USA) or a properly formatted 3.5-inch floppy disk. In case no disk is inserted, data is

stored on the PRM's hard disk. In a second step data on the hard disk can be transferred to a USB pen drive, which can store up to 400 unique, encrypted and password-protected patient records. USB drives from Boston Scientific or commercially available USB drives can be used.
Either export option (floppy disk or USB drive) allows the user to select between saving all stored episodes or only particularly selected ones. This allows the user enhanced patient data management in external systems.
The following information can be transferred:

- Current programmed parameter values
- Battery status and lead measurements
- Paced/sensed counters and histograms
- Therapy history (including stored electrograms)
- Trending values
- Heart Rate Variability data (if applicable)

Usually data on a patient disk can only be reviewed via a Boston Scientific programmer and can't be read by a computer. The format, type and structure of the files is confidential hence is not explained in detail in this thesis. These files were used for further processing within the scope of this thesis.
The file format which is used for the export to a USB drive is PDF and can be reviewed on any computer and attached to electronic medical records [57, 59, 60].

2.3 Development Environment

2.3.1 Java

Java is both an object-oriented programming language and a platform, developed by Sun Microsystems (since 2010 Oracle, Massachusetts, USA). The Java Technology basically consists of the Java Development Kit (JDK) to develop software and the Java Runtime Environment (JRE) to execute it. It is used in various kinds of industries and runs on a wide range of different devices, for example Blu-ray players, computers, gaming consoles, cell phones and medical devices.

2.3.1.1 The Java Programming Language

The design goals of the Java programming language were defined in The Java Language Environment, a white paper written by Gosling J. and McGilton H. in 1996 [61]. Some of the buzzwords Java was characterized with are simple, object-oriented, familiar, robust, secure, architecture neutral, portable, distributed, multithreaded and dynamic. The most important characteristics are described below:

- Simple, object-oriented and familiar: The programming language has the ability to be easily programmed by most developers and learned by current developers (similar to C and C++). It is object-oriented to meet modern software development methodologies.

- Robust and secure: Compile-time checking and a second level of run-time checking ensures highly reliable code. There is no pointer arithmetic and memory space is automatically managed by the garbage collector. Features in the language and run-time system ensure that applications can't be invaded from outside.

- Architecture-neutral and portable: The same program can run on various systems independent from the underlying hardware architecture. The size of basic data types are specified, as well as the behaviour of its arithmetic operators.

As can be seen in Figure 17, Java source code is written in plain text files ending with .java extension which can't be executed by computer system. Then these files are compiled into bytecode. The bytecode-files with .class extension contain the same bytecode, independent from the system that created it respectively will execute it and represent the machine language which is interpreted by the Java Virtual Machine (JVM). Every application runs in an instance of the Java Virtual Machine that is specifically designed for a particular operating system and available on various different platforms like Microsoft Windows, Solaris OS, Linux or Mac OS.

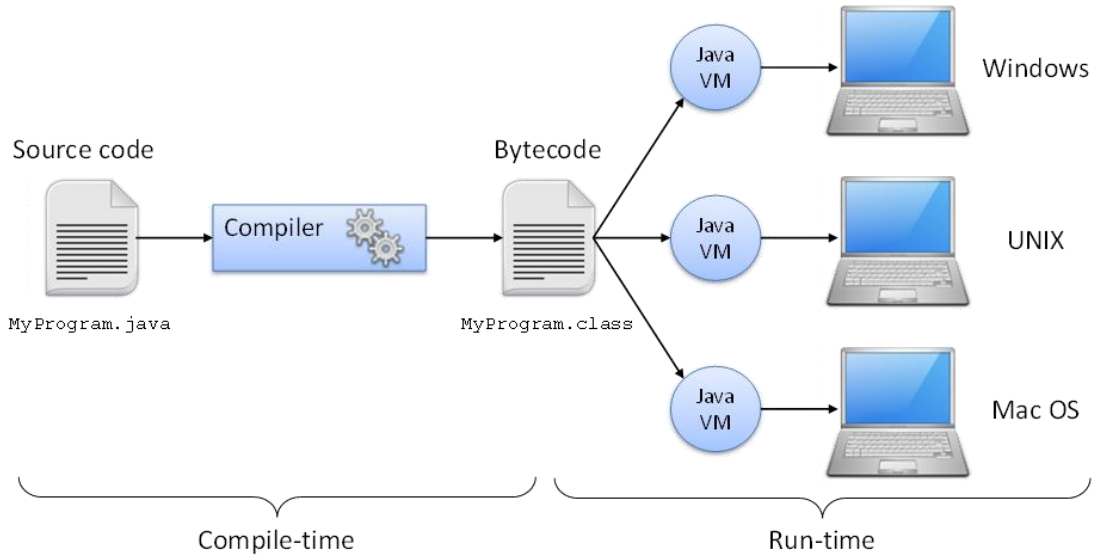

Figure 17: Concept of the Java Virtual Machine.
The source code is compiled into bytecode and then interpreted by a system-specific Java Virtual Machine. Redrawn from [62].

2.3.1.2 The Java Platform

The Java platform (Figure 18) has two components, namely the Java Virtual Machine and the Java Application Programming Interface (API) which ensure the separation of the source code and the underlying system architecture. The Java Virtual Machine was already described in the last chapter. The Application Programming Interface provides the core functionality and contains ready-to-use software components grouped in libraries (packages) of classes and interfaces. They allow developers to work with abstract interfaces (e.g. input/output interface) without knowing the underlying hardware architecture [62].

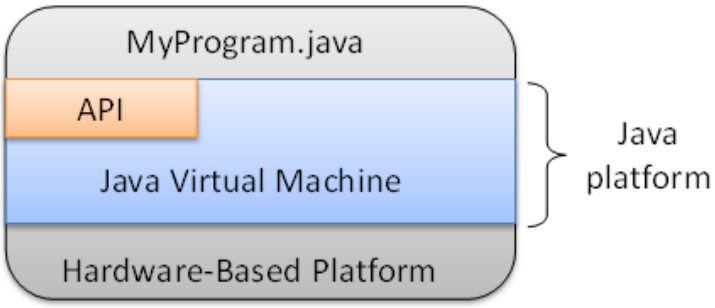

Figure 18: The Java platform consists of the Application Programming Interface and the Java Virtual Machine.
This principle separates the source code from the underlying operating system and hardware. Redrawn from [62].

The Java Development Kit contains the software packages, the Java compiler, the Java Runtime Environment (Java Virtual Machine) and additional features for the development of

a Java application. Its current version JDK 7 can be downloaded freely from the Oracle homepage[2].

2.3.1.3 The Java Security Architecture

The security aspect was an essential requirement in the development of the Java technology. Basically security is achieved by the implementation of three stages [63]:

Security in the Programming Language

The robustness of the language is achieved by avoidance of directly addressing memory by pointers. There is no such data type like pointers in C or C++. The memory management of Java automatically initializes classes and instance variables. As soon as a memory area is not referenced anymore the so called Garbage Collector de-allocates the unused memory. Objects can only be used with their defined data type and the keywords public, protected and private manage the access of attributes and methods.

Security in the Runtime Environment

The Java Security Model checks local and external code (e.g. internet) depending on the declared policy in a policy file. An important part of this model is the protection domain. A protection domain serves as a mechanism for grouping and isolating different units of protection. It encloses a set of classes whose instances are granted the same set of permissions. The Java application environment sets the relation between classes/instances, their protection domains and their permissions (Figure 19) [62].

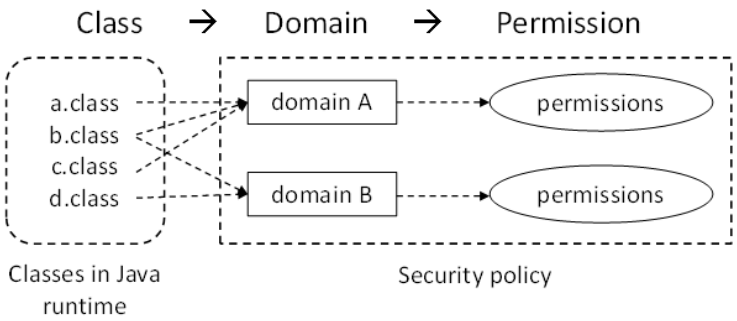

Figure 19: Security architecture in Java.
Different classes/instances are assigned to protection domains with a defined set of permissions. Redrawn from [62].

[2] http://www.oracle.com/technetwork/java/javase/downloads/index.html

One example for a domain is the so called Sandbox concept which was utilized in JDK 1.0. Each Java application runs in a Sandbox of the Java Virtual Machine. The Sandbox contains a Byte Code Verifier, a Class Loader and a Security Manager. The Verifier checks the code, the Class Loader decides if external files are loaded or not and the Security Manager grants/denies access to system resources.

Additional Security Mechanisms

The Java Security API provides various packages to realize concepts for permission and right management. The Java Cryptography architecture allows the developer to establish save communications.

2.3.1.4 Java Archive Files

To bundle multiple components like .class files, images and sounds into one file, Java Archive (JAR) files can be created. The file format is based on the ZIP format, is platform-independent and supports compression leading to a reduction of the file size. A JAR file is downloaded by the browser in a single HTTP transaction which significantly improves the download speed. On Microsoft Windows systems the extension .jar is associated with an executable JAR file. So when a JAR file is double-clicked it automatically starts.

JAR files can be digitally signed to get signature-related permissions. Therefor a Signing and Verification Tool called Jarsigner is available. First a private key has to be defined. This private key and its associated public key certificates are stored in databases called keystores. A keystore can comprise many keys, which can be identified by aliases.

The command for generating a key in a keystore:

```
keytool –genkey –keystore <path of the keystore> -alias <alias of the key>
```

The command for signing a JAR file is:

```
jarsigner –keystore <path of the keystore> <path of the JAR file> <alias of the keystore>
```

2.3.1.5 Java Web Start

The Java Web Start technology [62] represents an application, which is launched by clicking a link in a web page that points to a Java Network Launch Protocol (JNLP) file. Java Web Start then automatically downloads, caches, and runs a given Java technology-based, full-featured application without any installation procedure. Java Web Start is installed as part of the Java Runtime Environment (version JRE 1.3 or later) software and doesn't have to be installed separately.

The Java Network Launching Protocol technology defines a standard file format (JNLP file) which provides the necessary information on how to launch an application.

Java Web Start has the following key benefits:

- A single Java application on a web server can be deployed on various platforms like Windows, Linux or Solaris.
- Java Web Start allows the application to use a specific version of the Java Runtime environment, which means that multiple versions of the Java platform are supported simultaneously.
- Once an application was launched it is cached locally which leads to a better performance and can be used without a browser so it can run standalone from the user's desktop.
- The Java security architecture (e.g. Sandboxes) is used to run applications, which ensures a save environment with restricted access to critical system resources, according to the declared security policy for the specific application.
- If an application was updated, Java Web Start automatically downloads the new version.

Java Network Launching Protocol (JNLP)

JNLP is a platform independent protocol which specifies how an application is exactly installed, launched, distributed and updated with Java Web Start. Therefor a set of rules is defined in an XML file. The file is structured in the functional categories application descriptor, applet descriptor, component descriptor and installer descriptor and optionally

contains meta-information, security and resource information. When the user clicks on a link to start a Web Start application, the JNLP file is downloaded and forwarded to Java Web Start. Java Web Start reads the file, which contains amongst others the address to the application, checks if it has already been downloaded and checks for newer versions. Then it is downloaded and/or launched. A detailed description of the file syntax can be found on the Oracle homepage [62].

2.3.1.6 Libraries

JDOM

JDOM provides open source libraries to read and write XML data, specifically designed for the Java platform. Similar to the Document Object Model (DOM) an XML file is represented as a tree structure in the memory. Due to the Java-based approach each XML element is represented by a specific Java class (e.g. document, element). JDOM integrates with other APIs such as Document Object Model (DOM) and Simple API for XML (SAX), which provide the output of a document not only to a Stream or a Reader, but also as a SAX Event Stream or as a DOM Document.

Apache Commons

Apache Commons is an independent collection of libraries for Java and focuses on server relevant functionality. The HttpClient component provides an efficient package for the implementation of HTTP standards and recommendations. An important feature in the scope of this thesis was the full implementation of all HTTP methods (e.g. GET, POST).

JSON

JavaScript Object Notation (JSON) is a compact, text-based open standard for representing data structures and associative arrays in a human-readable data format. It is mainly used for the data interchange between applications, respectively between a server and a web application, and serves as an alternative to the XML standard in non-complex structured data. The org.json libraries support the reading and writing of this structure. The detailed structure is described on the homepage [64], where links to the libraries can be found as well. Here is a short example for the structure of a JSON file:

```
{
        "first name": "firsttest",
        "last name": "secondtest",
        "address": {
                "street": "teststreet",
                "city": "graz"
        }
}
```

2.3.2 JavaScript

JavaScript (JS) is a lightweight scripting language and is inserted into HTML pages to create dynamic websites. It is an object-oriented but classless language and can be used for procedural and functional programming. Despite syntactical similarities to Java, JavaScript is unrelated to Java. It is mainly implemented on the client side to dynamically manipulate websites by using the Document Object Model (DOM) and to send/receive data without forcing the browser to reload the page.

2.3.3 Python and Zope

Python (Python Software Foundation, Delaware, USA) is a dynamic, interpreted high-level programming language characterized by a very structured, easy-to-read syntax (e.g. nested code blocks, functions, classes). It supports multiple programming paradigms including object-oriented, imperative and functional schemes. Python automatically generates bytecode and executes it without a separate compile step. Hence it is often used as scripting language and for the implementation of web applications [65].

Zope (Zope Corporation, Virginia, USA) is an object-oriented open source web application server, written in Python. Because of the separation of data, logic and presentation of a website, Zope is ideally suited for creating dynamic pages. The functionality of Zope can be easily extended by many freely available features.

Chapter 3

Results

3.1 Integration of SPRINT into the Workflow[3]

The integration of SPRINT into the pre-existing workflow was based on the scheme of the IHE Profile Retrieve Form for Data Capture, which supports the retrieval of an empty or partly filled form from a form source, completion of the form and return of these data to a source application. In the context of this thesis, SPRINT represented the Form Filler actor, which retrieved an empty or partly filled form from PICARD (Form Manager), added all available data interrogated by the programmer and sent it back to PICARD (Form Receiver). As illustrated in Figure 20, the main actors of the workflow were PICARD (1 & 4), the programmers (2) and SPRINT (3). Basically there were two ways of gathering data, which then were merged, checked for discrepancies and returned to PICARD.

1. Manual Input of Data

In PICARD, documentation of a new follow-up was initialized by pressing the "New Follow-up" button on the PICARD Overview page (Figure 21). In the New Follow-up form (Figure 22) some of the follow-up fields were pre-filled with default values (e.g. current date, documenting physician, department, etc.) and the user had the chance to type in, modify or delete data manually. By pressing the "Import" button, currently entered data were extracted and transformed in standardized JSON format. A Python function launched the Java Web Start application SPRINT and forwarded the objects as arguments.

[3] From this chapter on, the term "SPRINT" and "PICARD" refer to "SPRINT 2.0" and "PICARD 2.0". If "SPRINT 1.0" or "PICARD 1.0" is meant, it is explicitly indicated.

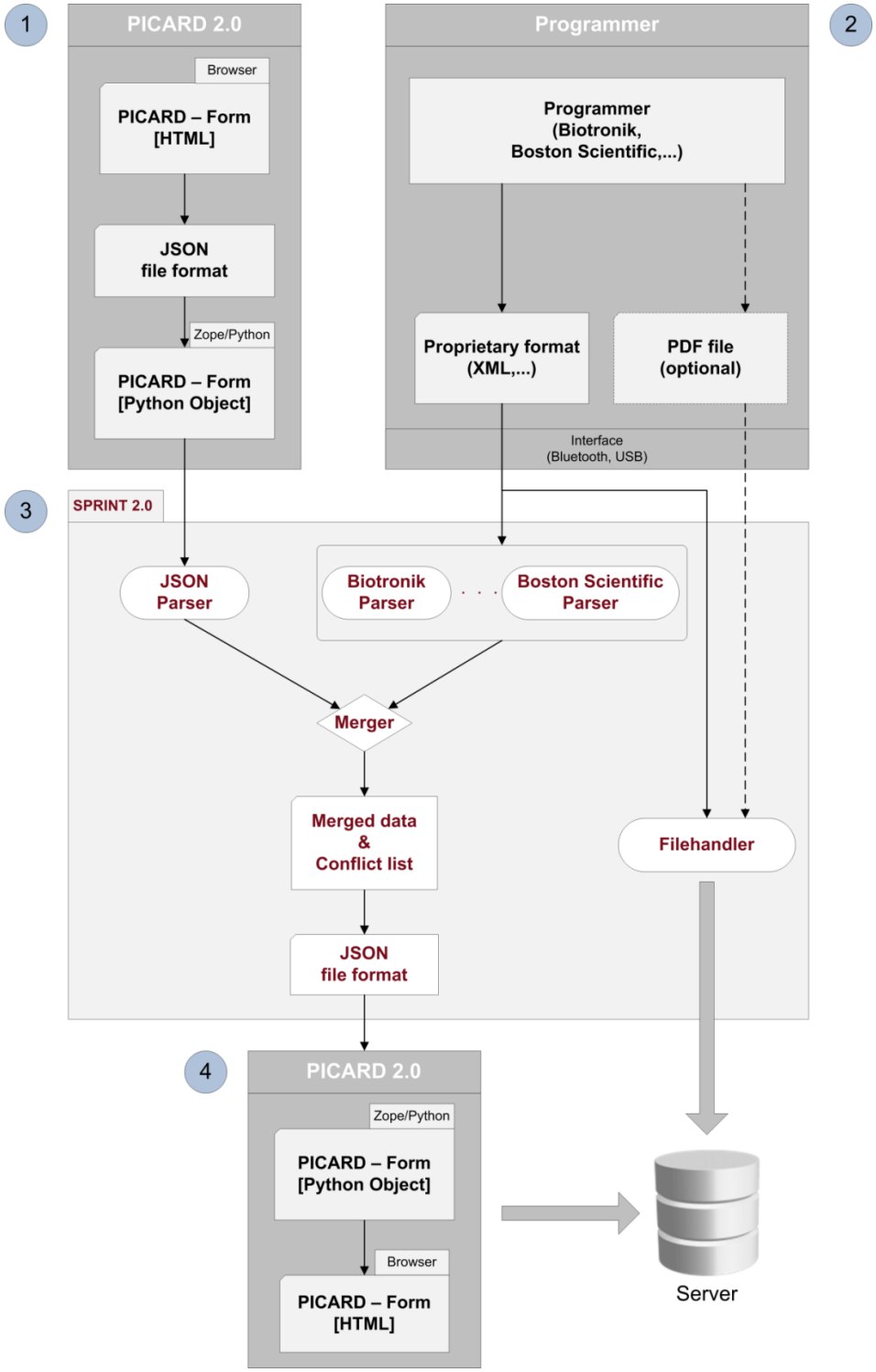

Figure 20: Schematic flow diagram of SPRINT integrated in PICARD.
(1) The manually entered data in the web form of PICARD were exported to SPRINT in JSON format. (2) Data from the programmer were transferred to SPRINT in a proprietary format and optionally in PDF format. (3) SPRINT parsed the data, mapped them to PICARD-specific parameters and merged and exported them back to PICARD. (4) The received data were entered in the web form and as the user confirmed their correctness, the data, source file and PDF file were stored on the server.

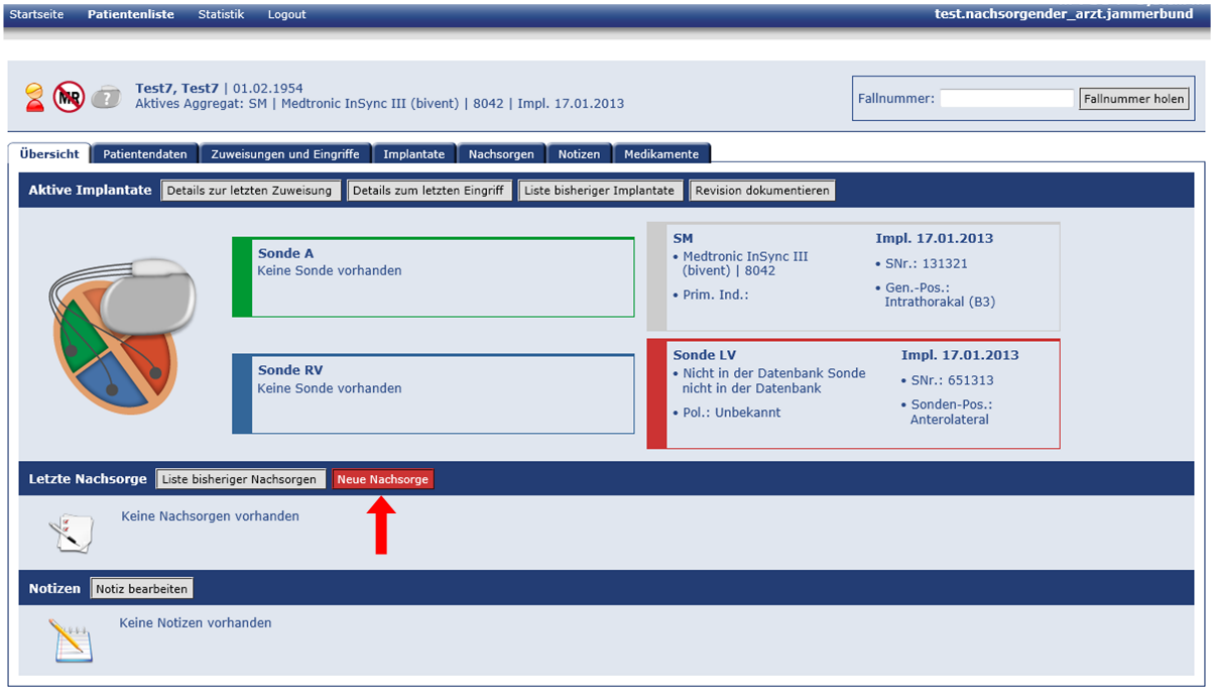

Figure 21: Overview page.
On this page all important information of the patient were summarized. The button "New Follow-up" is marked.

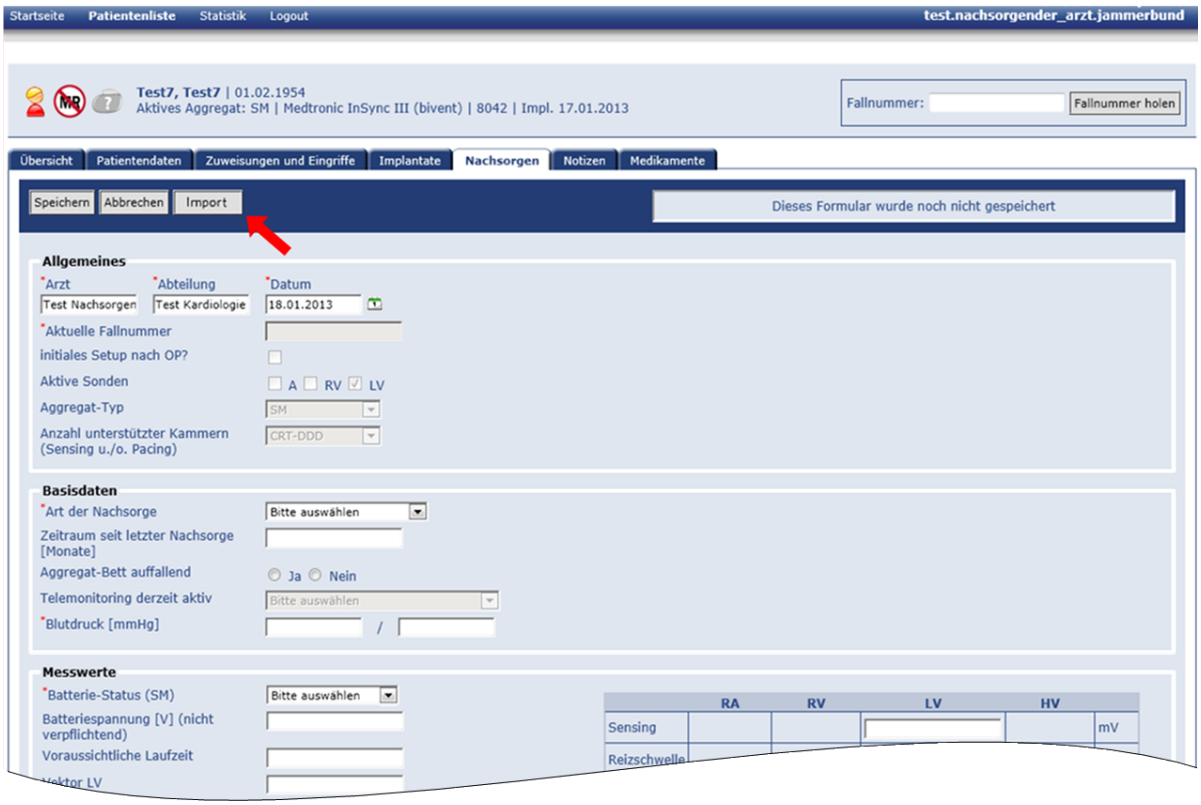

Figure 22: New Follow-up page. Some of the fields were already pre-filled.
Data could be added, deleted and modified. The button "Import" was added in the form.

2. Data Export from the Programmer

Selected proprietary files on the programmer could be exported to the local computer and processed by SPRINT. If available, a PDF file from the programmer was transferred to the client computer too. According to the supported technology of the programmer, a USB device, floppy disk or Bluetooth could be used for the data transfer.

3. Data Processing in SPRINT

The manually acquired and imported data were parsed and transformed from their proprietary format into Java objects. The parameters were mapped to PICARD specific parameters and merged by an algorithm that detected discrepancies and managed conflicts.

4. Data Export from PICARD

The processed data and the result of the conflict manager were returned to PICARD in JSON format. Then a Python function filled out the web form and in case of discrepancies they were displayed to the user. The proprietary file and PDF file were transferred to the server.

3.2 Data Extraction from the Web Form

Within the scope of the thesis the "Import" button in the "New Follow-up" form (Figure 22) was added. By clicking it, the user initiated the extraction of the manually entered data from the web form and the transformation in standardized JSON format with the following structure:

```
[["<name of HTML element 1>", "<value of element 1>"],
 ["<name of HTML element 2>", "<value of element 2>"],
                              .
                              .
                              .
 ["<name of HTML element N>", "<value of element N>"]]
```

This JSON String and additional database information, regarding the current patient and follow-up session, were then via a POST-request submitted to a python function. The function transformed the given arguments in Python objects, which were then forwarded to the JNLP file that launched the Java Web Start application SPRINT.

3.3 Data Export from the Programmer

Necessary available data, interrogated by programmers of different manufacturers, could be imported and automatically processed by SPRINT. Once the Java Web Start application (Figure 23) was launched, the user could start the standard Bluetooth connection manager of the operating system by clicking the button "Import via Bluetooth". Additionally the user had to select the files (XML, PDF optionally) on the programmer (Biotronik) and initiate the transfer to the computer. When Bluetooth was used for the first time, the pairing between the programmer and the computer had to be done.

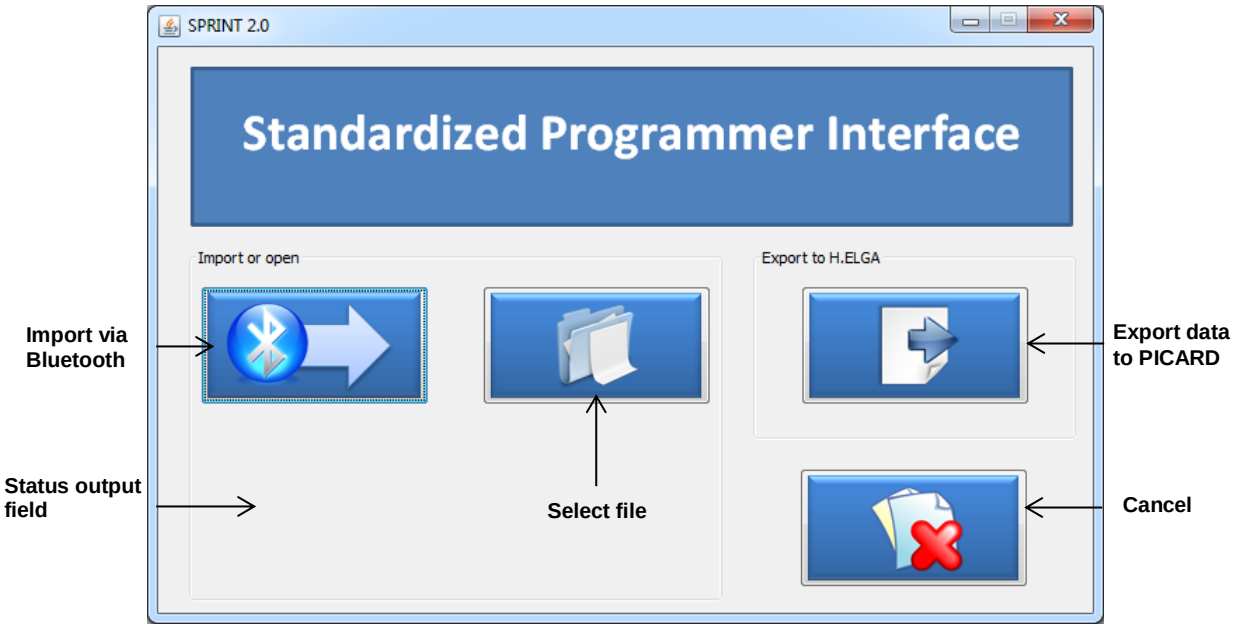

Figure 23: Graphical User Interface of the Java Web Start application - SPRINT 2.0.
The GUI contains four buttons and an output field for the current status of the process.

After the files were transferred, the "Select file" button enabled the user to choose the file that had to be processed. The two options of importing files via Bluetooth and selecting the file in a file chooser dialog were intentionally separated. Hence the user got the recurrent opportunity to import files and select one of them or select an already existing one from the file system.

3.4 Data Processing in SPRINT

At this stage of the procedure SPRINT had available data from two sources. On the one hand it got manually entered data, structured in JSON format and forwarded from PICARD. On the other hand there were data in a proprietary file format from the programmer available.

For further processing the JSON String was parsed and transformed in a JSON Array, using the Java package json.

Depending on the data format of the proprietary file (Biotronik, Boston Scientific) different actions had to be taken (Figure 24).

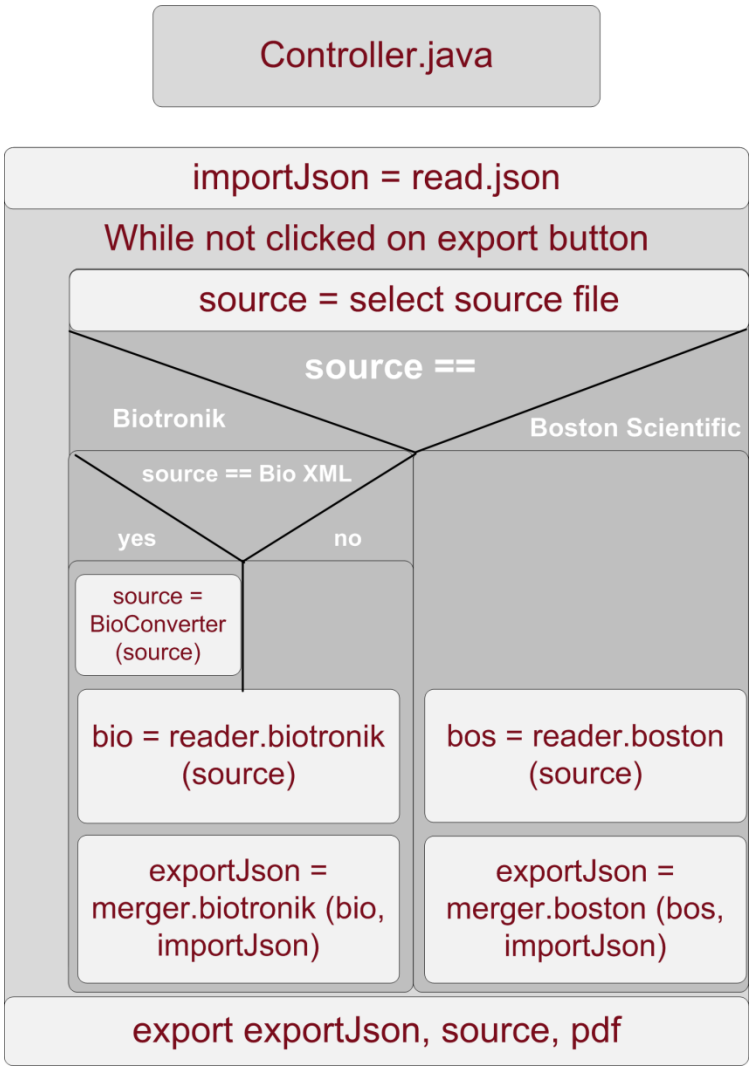

Figure 24: Concept of the controller class of SPRINT.
After importing all files, Biotronik files were converted with the adapter. Then both, Boston Scientific and Biotronik files were read in, transformed into Java objects and merged with the parameters imported in JSON format. As long as the "Export" button was not clicked new files could be selected.

In case of a Biotronik file the Biotronik XML format was converted into the IEEE 11073-10103:2012 standardized format. As already described, the Biotronik software adapter was used for this conversion. The Programmer adapter was integrated in the Java project. Hence it was part of the JAR package, which is automatically downloaded and saved locally on the computer by the Java Web Start technology. SPRINT located the folder of the package, extracted it and started the Biotronik software adapter. Then the converted data were parsed with the Java packages jdom (SAX) and transformed into a Java object hierarchy according to the IEEE standardization structure.
In case of a Boston Scientific file, it was parsed and the structure was represented by Java objects.

Once manually and automatically acquired data were available as Java objects, they had to be merged. Because the parameters were vendor specific (hence they had different proprietary naming conventions, formats and units), they had to be mapped to PICARD specific parameter conventions. Therefore mapping tables were created in MS Excel (see Appendix Table 6) and implemented by Java functions, which applied the necessary calculations and set parameter relations.

Then the merging algorithm processed the available parameters to detect discrepancies and manage conflicts. Each PICARD parameter was compared to the according transformed parameter of the programmer file. The results were classified in five data groups and one error group (Table 5).

Group	Programmer value	PICARD value	Result	Status code
1	null	null	Both null	0
2	available	null	Programmer value	1
3	null	available	PICARD value	2
4	available	available	if different – Conflict	3
5	available	available	if the same – Equal	4
6	-	-	Error	5

Table 5: Five possible results of the conflict manager.

The parameter name, both values and the status code of each comparison were saved for every parameter in a JSON Array, which was then converted into a String with the following structure:

```
[["<parameter name 1>","<programmer value 1>","<PICARD value 1>","<result 1>"],
 ["<parameter name 2>","<programmer value 2>","<PICARD value 2>","<result 2>"],
```

.
.
.

```
["<parameter name N>","<programmer value N>","<PICARD value N>","<result N>"]]
```

Reprogrammed-Parameter Section

Data from PICARD contained, amongst others, a section of data, stored in the proceeding session. If these parameters were available, SPRINT compared them to the new parameters to check if anything had changed since the last interrogation. According to the result of the comparison the section "Reprogrammed parameters" was filled out with the new interrogated values.

3.5 Data Export to PICARD

With a click on the "Export data to PICARD" button, SPRINT set up an Http-connection (apache commons libraries) and submitted the merged data in JSON format with a POST-request to the Zope server. Additionally the processed source file from the programmer and a PDF file, which could be optionally created by the programmer (Biotronik), were transferred to the server in a multipart request. Then SPRINT initiated the reload of the web form in the browser.

In the first step a Python function on the server received the JSON String and wrote it into a temporary text file on the server. The file was uniquely identifiable by the session ID and follow-up ID. In the second step a Python function read this temporary file in and processed the data, when the reload of the web form was initiated. Then the values were reassigned to the parameters in the HTML form (Figure 25) and the temporary file was deleted. In case of discrepancies the result of the conflict management was displayed to the user in a message field of the web form. In such a conflict situation the former value of PICARD was assigned to the corresponding parameter by default and the user got the opportunity to manually check the values as pointed out in the message.

Due to archiving reasons the transferred files were automatically stored on the server. If the PDF file was available, a link was added in the Follow-up Overview page (Figure 26). Hence the user always had the opportunity to quickly access the original file exported from the programmer.

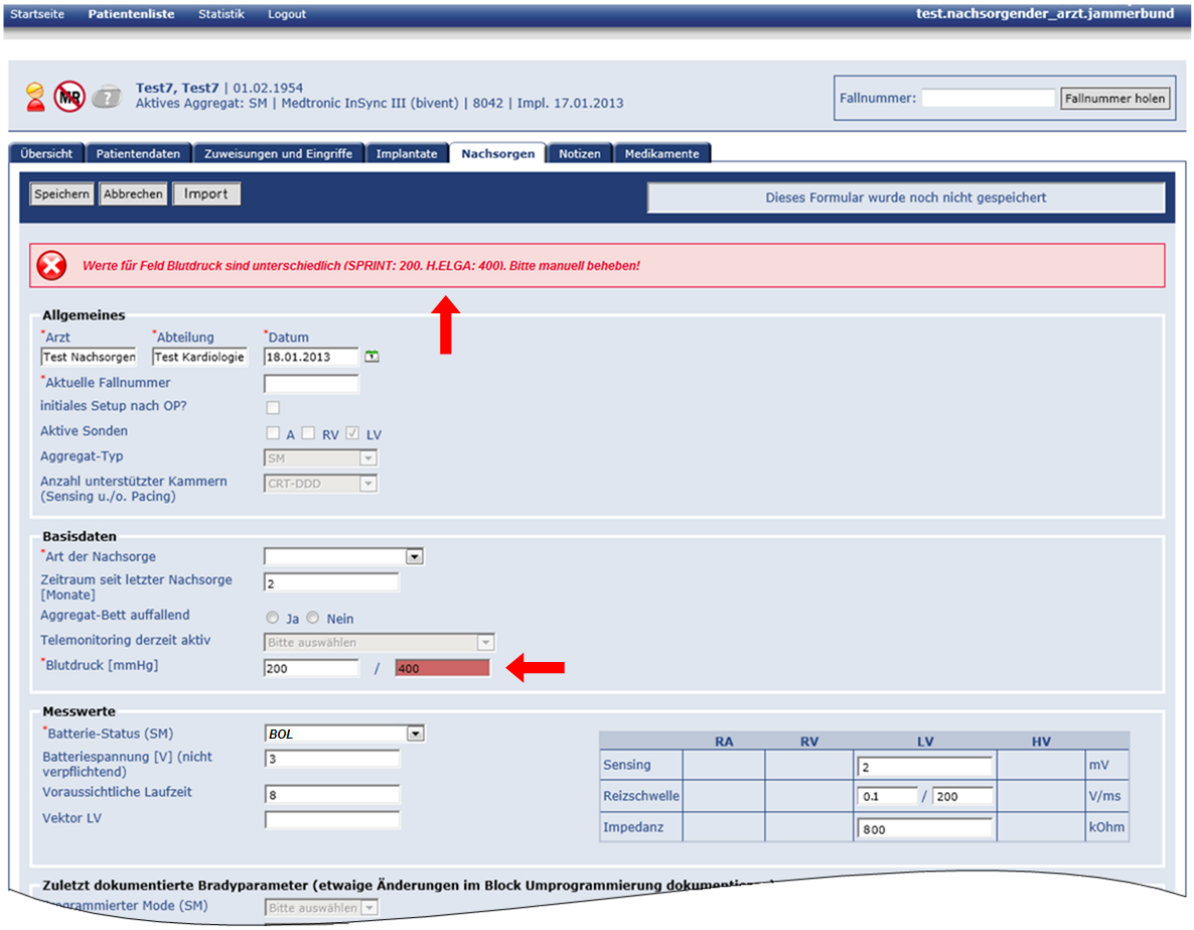

Figure 25: Reloaded Follow-up web form.
Values were automatically imported. A conflict was displayed in the message field and the PICARD (H.ELGA[4]) value was assigned to the field by default.

Figure 26: Follow-up Overview page.
The page contained all follow-ups of a patient that had been conducted so far. If a PDF file was available on the server, a direct link to it was added.

[4] German name of PICARD: H.ELGA… Herzschrittmacher.Elektronische Gesundheitsakte

3.6 Java Packages

For the development Eclipse IDE for Java Developers, Version Indigo Service Release 1 and Java 6 SE were used.

During the implementation of the interface in Java a very modular structure, based on the Model-View-Controller (MVC) architecture, was emphasized. This was extremely important because it offered the opportunity to easily upgrade the interface by adding data import-modules for other manufacturers.

As Figure 27 shows, the basic parts of the structure (JAR file) included the Controller, Reader, Merger, Selection, Structure packages and external libraries (jdom, apache.commons, json).

Controller: The Controller package comprised the main class, a class to control the program flow and react to user interactions and the Graphical User Interface (GUIDesigner – displayed to the user & optionally GUIDeveloper – for developers). Furthermore some utility classes are part of the package.

Reader: For each manufacturer, one package with the aim to read in vendor-specific file formats was created. These functions convert the structure of the vendor-specific file into a Java-based object structure (e.g. JDOM for the XML structure of Biotronik files).

Structure: These packages include Java classes that represent the vendor-specific file format structure. Each structure is mapped to classes in one package (e.g. IEEE11073-10103 structure).

Merger: For each manufacturer, one package with the aim to map the vendor-specific parameters to PICARD-specific parameters was created. The classes contain the necessary functionality (calculations, constants and utilities), to assign proprietary vendor parameters to PICARD parameters.

Selection: This package contains classes for the conflict management and constants defining the parameter names of PICARD.

In case of adding a module for another manufacturer a new Reader, Merger and Structure can be easily added to the current system.

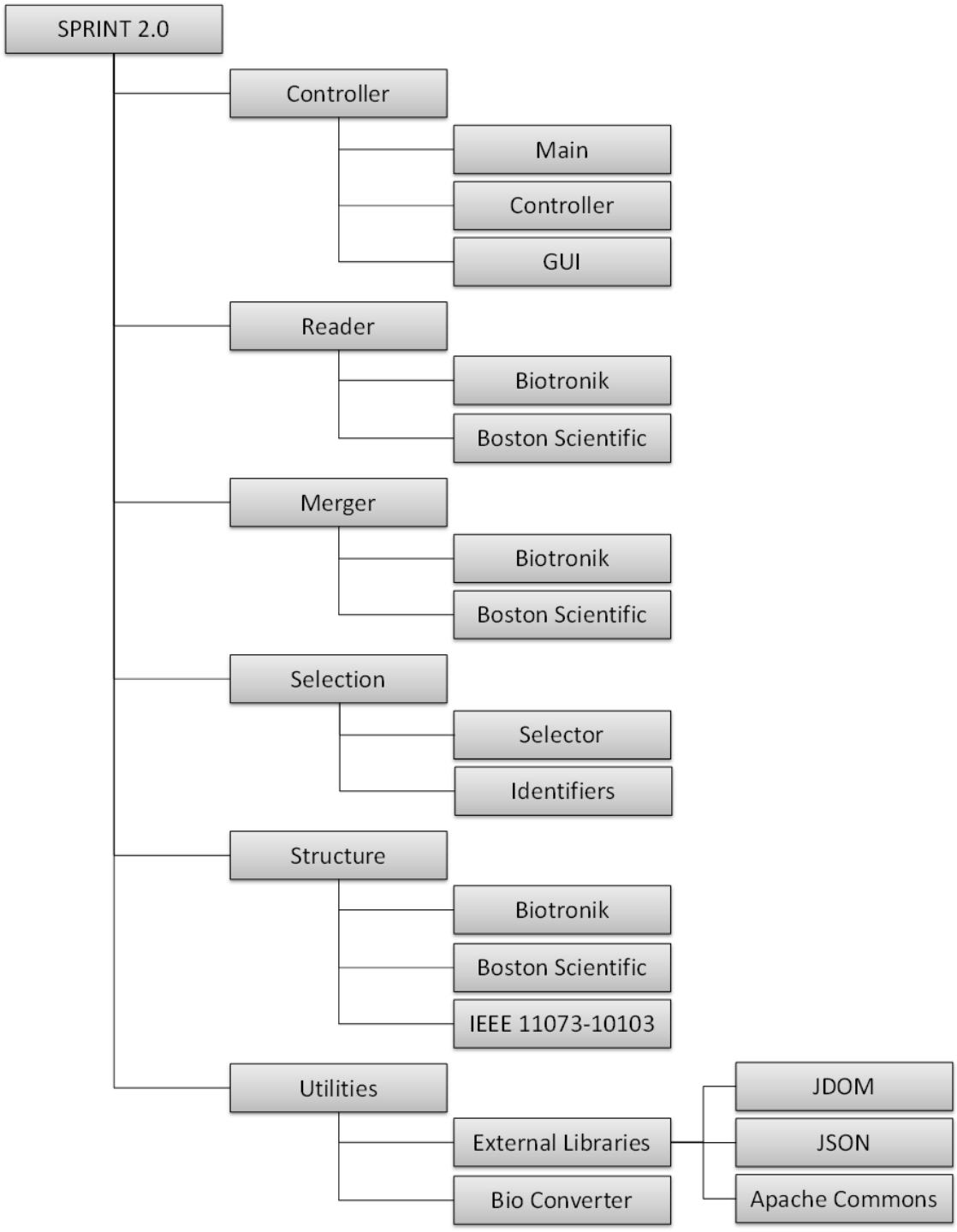

Figure 27: Basic structure of the Java packages in the JAR file.

3.7 Evaluation

The interface was evaluated using three sample files with data from interrogations of an ICD, an implantable pulse generator (IPG) and a CRT from the Biotronik webpage [54] and several test files from the models Cognis and Teligen, kindly provided by Boston Scientific. Additionally numerous follow-ups were documented with samples from interrogations directly from the Biotronik programmer (Renamic), which was provided by Biotronik.

Time for downloading and launching the Java Web Start application for the first time took about five seconds. This could be explained with the download of the application and the user's reliability confirmation of the signed application. This step was necessary because each time SPRINT was updated the new version had to be downloaded and needed the permissions to access the local file system for importing data files. This could be reduced to about three seconds when starting the application again, because this step had to be done only once after updating SPRINT.

For the tests a Dell laptop with the following specification was used:

- Windows 7 Enterprise Edition SP1
- Intel Core 2 Duo T7100 @ 1,8 GHz 1,8 GHz
- RAM 2 GB
- 32 Bit
- Java 6 Standard Edition

Time for the whole procedure from selecting the file in the file system to exporting the results back to PICARD took approximately 25 seconds. SPRINT could import, process and export all available test files of both manufacturers without any errors. The results, displayed in PICARD, were in all test cases correct.

Chapter 4

Discussion

A prototypical interface for automated import of proprietary CIED data into PICARD has been developed. The system supported the export from Biotronik and Boston Scientific programmers to the local client computer by Bluetooth and the export of manually entered data from the web form. It featured automated processing of proprietary data formats and facilitated the export into PICARD. Furthermore it was capable of uploading the original source files, gathered from the programmer, to the server and allowed the user a direct access to the PDF file, if available. Due to the integration of the interface in the existing workflow of the follow-up management of PICARD, users will be disburdened in documenting relevant follow-up data, which is expected to increase documentation quality, disburden medical staff and decrease the costs of CIED management.

The time needed for starting the application for the first time took about five seconds and three seconds thereafter. The time for the whole process from the import of the file to the export of the results back in PICARD took approximately 25 seconds. This time is expected to be tolerable as compared to the time needed for manually documenting the relevant data in PICARD (it is expected, that approximately 10 minutes per follow-up could be saved, if the presented interface was implemented).

4.1 Retrieve Form for Data Capture

Data exchange of SPRINT and PICARD was based on the IHE Profile, Retrieve Form for Data Capture. According to the specification of the profile [49] RFD uses XForms technology to support negotiation between the form display and form provider systems. Although XForms technology is recommended by W3C, there is an on-going discussion about the future of XForms in the community. Currently available browsers like Mozilla Firefox 18.0 or

Microsoft Internet Explorer 10 do not support XForms in the first place. The XForms support is only featured by browser plugins which have to be downloaded and installed. Furthermore the plugin for Mozilla Firefox is only supported by Firefox 2 and Firefox 3 and has not been maintained actively any more since about 2010.
SPRINT represents an additional feature of the already implemented PICARD system and is intended to be used on computers in hospitals. Hence aspects of security and maintenance are crucial. In order to avoid problems concerning the installation and maintenance of XForms plugins and changes in the current system, XForms technology was replaced by the commonly and widely used JSON standard for the data exchange between SPRINT and PICARD.

Furthermore RFD specifies a form retrieval request from the Form Filler to the Form Manager. According to the meta-information in the request, the Form Manager selects/creates the appropriate form. In this thesis SPRINT (Form Filler) always retrieved the same data with the same structure (form) from PICARD (Form Manager), which made the request unnecessary.

4.2 Java Applet versus Java Web Start

There are several different characteristics of Java Applets and Java Web Start technology, which were considered in the decision of how to deploy SPRINT. The main aspects why Web Start was used are listed in the following section.

Java Web Start combines advantages from Java Applications and Applets:

- Java Web Start applications are downloaded from the internet and due to security reasons have to be signed. They get the permissions to access persistent storage, file I/O, and other client related services.

- Once an application is downloaded, it is locally cached on the computer and does not have to be downloaded every time it is launched, which leads to an increase of the performance. The application could even be used without a browser.

- Java Web Start applications are automatically installed and if a newer version is available on the server it is automatically updated locally.

4.3 DOM versus SAX

DOM and SAX are the two most popular APIs for processing XML documents in Java. The following section describes why SAX was used for the parsing of Biotronik XML files [66]:

- SAX represents only a reader of XML documents. It doesn't support the manipulation of data, while DOM can be used for reading and writing. As SPRINT 2.0 didn't change any source files, SAX was adequate.
- DOM can be used for random processing of XML documents, which means that the whole DOM tree is loaded into the memory. SAX is fast and not that memory consuming because it can process data only sequentially. Hence there is no access to previously read data. As SPRINT 2.0 read in data and internally transformed them into a Java-based object structure, there was no need to use DOM and additionally load the whole DOM tree into the memory.
- Due to the sequential processing of SAX, mistakes in the document might attract attention not till the end of the processing procedure. DOM checks the whole document while reading it into the memory, before any further processing is done. As the Biotronik XML files were created automatically, it was supposed that there were no mistakes in the files.

4.4 Differences to SPRINT 1.0

SPRINT was based on the prototype SPRINT 1.0, but nearly the whole architecture was changed to meet the new requirements.

- SPRINT 1.0 only used a serial interface (RS232) to import files from the programmer to the client computer. Currently available programmers like the Biotronik Renamic do not have a serial port in the first place, although there is a proprietary adapter from the USB interface to a serial interface available. Because it could not be assumed that every programmer still has a serial interface, SPRINT was based on the USB- and/or Bluetooth interface, which is supported by almost all new programmers.
- SPRINT 1.0 was implemented as a separate application and was not integrated in the workflow.

- SPRINT 1.0 processed only data from Biotronik programmers and did not consider input from the user, which is essential in a real-life application.
- SPRINT 1.0 created standardized HL7 messages of the processed data and sent them directly to a database. In SPRINT the user should have the possibility to take a look at the processed data and if necessary change, add or delete them before confirmation. This was extremely important because the user (physician) was responsible for the imported data.

4.5 Limitations and Drawbacks

Data Export from SPRINT to PICARD

In the current implementation of SPRINT the merged data in JSON format were transferred to the server by a POST-request. Then a Python function saved the data in a temporary text file on the server. When the reload of the website was initiated by SPRINT, another function read the data from that temporary file and assigned them to the parameters on the page. Then the temporary file was deleted.
This complicated two-step procedure had been implemented, because the JSON String was too long to be forwarded as an argument in the URL (GET-request) that reloads the page. The Microsoft Internet Explorer supports a maximum length of 2083 characters and the Apache Webserver has a limit of 8192 byte. After a lot of research this was the most feasible way to avoid these problems in the pre-existing PICARD environment.

Steady USB Data Connection

In the first place a steady USB connection with a data link cable between the local client computer and the programmer was planned for the data exchange. The data link cable DIGITUS USB Data Transfer Cable (Assmann Electronic GmbH, Lüdenscheid, Germany) facilitated a USB connection between two host-controllers (masters) but although nothing had to be installed (Plug and Play), the embedded software could not be launched on the programmer (most likely due to security reasons).

Data Import from other Manufacturers

Data format descriptions, which explain the data structure and parameter names, have to be available to map them. Most of them are not publicly available and can only be accessed with

third party nondisclosure agreements (Boston Scientific).
Biotronik's data structure is explained in a PDF document on the company's homepage [54].

Evaluation in a Real-life Scenario

It was planned to conduct usability tests in a real-life scenario at the pacemaker outpatient ward at LKH University Hospital Graz. Due to the fact that PICARD started its routinely use at the end of this thesis, SPRINT could not be tested there anymore. The interface was prototypically implemented and tested using existing examples of exports from different programmers.

Requirements for the Client Computer

The data extraction from the web form was conducted by JavaScript functions, so JavaScript had to be enabled in the browser. The user had to install Java 6 SE or Java 7 SE to run Java Web Start. The installation required system administrator rights.
The Java Web Start application could be launched and permissions could be granted to it without administrator rights.

Safety and Security Issues

The files were exported from the programmer to the client computer, where they were saved for further processing. So far they were not deleted after processing and uploading them to the server, because it would have been impractically in the test environment. Due to security reasons this could become an issue, which can be easily avoided by just deleting them after processing.

4.6 Outlook

Future work will include the development of a better solution for the data export in JSON format from SPRINT to PICARD as already mentioned in the last chapter. To solve this problem, wider adaptations of PICARD have to be implemented.

Another important point is the conduction of a usability test and evaluation in a real-life scenario at the pacemaker outpatient ward at LKH University Hospital Graz and the inclusion into the routinely used PICARD system, which is planned for 2013.

So far only data from programmers were considered. SPRINT could be extended by modules capable of importing data from home monitoring systems from various manufacturers. As most manufacturers provide web-platforms for physicians to check on remotely acquired patient data, these data could also be imported and mapped into PICARD specific parameters.

As the number of loop recorders is increasing and PICARD supports only pacemakers, implantable cardiac defibrillators and cardiac resynchronization therapy devices both systems, PICARD and SPRINT, could be extended to support them too.

Chapter 5

Conclusion

The requirements specified in Chapter 1.9 were all accomplished, except tests in a real-life scenario. SPRINT facilitated an easy, fast and save import of data files via Bluetooth or USB drive from two programmers of different manufacturers (Biotronik Renamic and Boston Scientific Latitude) into PICARD, while the user still had the opportunity to add, modify or delete data manually. The user got feedback about differences between manually and automatically acquired data and had to confirm their correctness. SPRINT was integrated in the workflow architecture of PICARD, and the data exchange was based on the scheme of the IHE Profile Retrieve Form for Data Capture. Furthermore the ability to easily upgrade the interface was emphasized in the design and development. The interface was prototypically implemented and tested using existing examples of exports from different programmers.

Although medical standards such as IEEE 11073-10103 exist, standardized solutions for data exchange of cardiovascular implantable electronic devices are rarely implemented. SPRINT supported the development towards a consensus on transmitting and storing data from various proprietary sources by transforming them into a homogeneous data structure in PICARD. The platform then allowed further standardized data processing and storing.

Literature

[1] Huppelsberg J, Walter K: *Kurzlehrbuch Physiologie*, 3. Auflage, Stuttgart, Germany: Thieme Verlag; 2009.

[2] Fuster V, Kelly B: *Promoting Cardiovascular Health in the Developing World*, Washington, DC, USA: The National Academies Press; 2010.

[3] WHO - World Health Organization. Cardiovascular disease. http://www.who.int/en . 2013.

[4] Beaglehole R 1999: **International trends in coronary heart disease mortality and incidence rates.** *Journal of Cardiovascular Risk* 1999, **6(2)**:63-8.

[5] Capewell S, Morrison CE, et al.: **Contribution of modern cardiovascular treatment and risk factor changes to the decline in coronary heart disease mortality in Scotland between 1975 and 1994.** *Heart* 1999, **81(4)**:380-386.

[6] Unal B, Critchley JA, et al.: **Explaining the decline in coronary heart disease mortality in England and Wales between 1981 and 2000.** *Circulation* 2004, *109(9)*:1101-1107.

[7] Capewell S, Beaglehole R, et al.: **Explanation for the decline in coronary heart disease mortality rates in Auckland, New Zealand, between 1982 and 1993.** *Circulation* 2000, **102(13)**:1511-1516.

[8] American Heart Association. *Heart Disease & Stroke Statistics – 2009 Update.* American Heart Association 2009.

[9] Nichols M, Townsend N, et al.: *European Cardiovascular Disease Statistics.* 2012 edition. Brussels, Belgium: European Heart Network and European Society of Cardiology; 2012.

[10] Mayo Clinic. http://www.mayoclinic.com . 2013.

[11] Lewalter T, Lüderitz B: *Herzrhythmusstörungen: Diagnostik und Therapie*, 6. Auflage, Heidelberg, Germany: Springer Medizin Verlag; 2010.

[12] Scharf C, Duru F: **Herzrhythmusstörungen.** In *Siegenthalers Differenzialdiagnose*, 19. Auflage. Edited by Siegenthaler W. Stuttgart, Germany: Thieme Verlag; 2005:722-736.

[13] Epstein EA, DiMarco PJ, et al.: **Practice Guideline - ACC/AHA/HRS 2008 Guidelines for Device-Based Therapy of Cardiac Rhythm Abnormalities.** *Journal of the American College of Cardiology* 2008, **(51)21**.

[14] Buchanan Keller K, Lemberg L: **The Sick Sinus Syndrome.** *American Journal of Critical Care* 2006, **15**:226-229.

[15] Dobrzynski H, Boyett RM, et al.: **New Insights Into Pacemaker Activity: Promoting Understanding of Sick Sinus Syndrome.** *Circulation* 2007, **115**:1921-1932.

[16] Fröhlig G, Carlsson J, et al.: *Herzschrittmacher- und Defibrillator-Therapie,* 1. Auflage, Stuttgart, Germany: Thieme Verlag; 2006.

[17] American Heart Association. http://www.heart.org . 2013.

[18] Hoppe UC, Böhm M, et al.: **Leitlinien zur Therapie der chronischen Herzinsuffizienz.** *Zeitschrift für Kardiologie* 2005, **94**:488-509.

[19] Lloyd-Jones MD, Larson GM, et al.: **Lifetime Risk for Developing Congestive Heart Failure: The Framingham Heart Study.** *Circulation* 2002, **106**:3068-3072.

[20] Wilkoff LB, Auricchio A, et al.: **HRS/EHRA Expert Consensus on the Monitoring of CIEDs: Description of echniques, Indications, Personnel, Frequency and Ethical Considerations.** *Europace* 2008, 10, 707-725.

[21] Müller A, Helms MT, et al.: **Remote Monitoring in Patients with Pacemakers and Implantable Cardioverter-Defibrillators: New Perspectives for Complex Therapeutic Management.** In *Modern Pacemakers – Present and Future*. Edited by Das K M, Rijeka, Croatia: InTech; 2011:147-166

[22] Brugada J, Auricchio A, et al.: *The EHRA White Book 2008/2009/2012*, Sophia-Antipolis, France: European Heart Rhythm Association; 2008/2009/2012.

[23] Gomboth H, Anelli Monti M, et al.: **Perioperatives Management von Patienten mit implantiertem Schrittmacher oder Kardioverter/Defibrillator.** *Anaesthesist* 2009, **5**.

[24] Toogood G: **Pacemaker therapies in cardiology.** *Australien Family Physician*. 2007; **36(7)**:518-9,521-4.

[25] Tolat VA, Zimetbaum JP: **Pacemaker implantation and indications.** In *Implantable Cardiac Pacemakers and Defibrillators: All you wanted to know*. Edited by Chow W A, Buxton E A, Oxford, UK: Blackwell Publishing; 2006:53-69.

[26] Ulbel M: *EKG-Darstellungssoftware zur telemedizinischen Herzschrittmacher-Nachsorge*. Diploma thesis at Graz University of Technology, 2008.

[27] Bernstein DA, Daubert J-C, et al.: **The Revised NASPE/BPEG Generic Code for Antibradycardia, Adaptive-Rate and Multisite Pacing.** *Journal of Pacing and Clinical Electrophysiology* 2002, **(25)2**.

[28] Behrens S, Zrenner B, et al.: **Jahresbericht 2010 des Deutschen Herzschrittmacher und Defibrillatorregisters, Teil 1**. Fachgruppe Herzschrittmacher und AQUA – Institut für angewandte Qualitätsförderung und Forschung im Gesundheitswesen. 2010.

[29] Reiffel AJ, Dizon J: **The Implantable Cardioverter-Defibrillator: Patient Perspective.** *Circulation* 2002, **105**:1022-1024.

[30] Cleland GFJ, Daubert J-C, et al.: **The Effect of Cardiac Resynchronization on Morbidity and Mortality in Heart Failure.** *The New England Journal of Medicine 2005*, **352**:1539-49.

[31] Bristow RM, Saxon AL, et al**.: Cardiac-Resynchronization Therapy with or without an Implantable Defibrillator in Advanced Chronic Heart Failure.** *The New England Journal of Medicine* 2004, **350**:2140-50.

[32] Strickberger AS, Conti J, et al.: **Patient Selection for Cardiac Resynchronization Therapy.** *Circulation* 2005, **111**:2146-2150.

[33] Lane ER, Cowie RM, et al.: **Pacing therapies for heart failure.** In *Implantable Cardiac Pacemakers and Defibrillators: All you wanted to know*. Edited by Chow W A, Buxton E A, Oxford, UK: Blackwell Publishing; 2006:110-133.

[34] Kollmann A: **A platform for integrated and patient-centred therapy management of chronic diseases**. Doctoral thesis at Graz University of Technology, 2008.

[35] Hayn D, Kumpusch H, et al.: **Information Technology Aspects of Integrated Cardiac Rhythm Disease Management.** In *Modern Pacemakers – Present and Future*. Edited by Das K M, Rijeka, Croatia: InTech; 2011:3-24.

[36] Inglis SC, Clark RA, et al.: **Structured telephone support or telemonitoring programmes for patients with chronic heart failure (Review).** The Cochrane Library 2010, 8.

[37] Kollmann A, Hayn D, et al.: **Telemedizin in der Herzschrittmacher – Nachsorge.** *Conference proceedings of the TELEMED 2006: 7-8 April 2006; Berlin.*

[38] Kollmann A, Hayn D, et al.: **H.ELGA – Herzschrittmacher.Eletronische Gesundheitsakte: Erste Ergebnisse aus der Steiermark.** *Conference proceedings of the eHealth 2007: 1 June 2007; Vienna*.

[39] AIT Austrian Institute of Technology: Pflichtenheft der H.ELGA. 2011.

[40] Medtronic – Paceart System. http://www.medtronic.com/for-healthcare-professionals/products-therapies/cardiac-rhythm/patient-management-carelink/medtronic-paceart-system/index.htm#tab1 . 2013.

[41] Biotronik: **Biotronik EHR DataSync® - Paceart Integration – Frequently Asked Questions.** Biotronik SE & Co. KG, Berlin, Germany. 2010.

[42] Biotronik: **Biotronik EHR DataSync® - EHR Integration-IT Specialists – Frequently Asked Questions.** Biotronik SE & Co. KG, Berlin, Germany. 2010.

[43] International Organization for Standardization: http://www.iso.org/iso/home/about.htm . 2013.

[44] Institute of Electrical and Electronics Engineering: http://www.ieee.org . 2013.

[45] Bluetooth® Technology: https://www.bluetooth.com . 2013.

[46] ISO/IEEE: **International Standard ISO/IEEE 11073-10101 First edition: Health informatics – Point-of-care medical device communication.** IEEE Standards Association, New York, NY. 2004.

[47] IEEE Standards Association: **IEEE 11073-10103: Health informatics – Point-of-care medical device communication.** IEEE Standards Association, New York, NY. 2004.

[48] CDISC – IHE Retrieve Form for Data Capture Profile: http://www.cdisc.org/healthcare-link . 2013.

[49] ITI Technical Committee: **IHE IT Infrastructure Technical Framework Supplement: Retrieve Form for Data Capture (RFD).** Integrating the Healthcare Enterprise International, Inc. 2010.

[50] Biotronik: **Cardiac Rhythm Management – Renamic: Technical Manual.** Biotronik SE & Co. KG, Berlin, Germany. 2012.

[51] Biotronik: **Press Release - Biotronik Introduces Industry's smallest implantable cardiac device programmer.** Biotronik SE & Co. KG, Berlin, Germany. 2011.

[52] Biotronik: **Cardiac Rhythm Management – Renamic: Quick Reference Guide.** Biotronik SE & Co. KG, Berlin, Germany. 2010.

[53] Biotronik: **Biotronik Home Monitoring – EHR DataSync: Documentation of the Biotronik IEEE 11073-10103 XML Structure Version 4.0.** Biotronik SE & Co. KG, Berlin, Germany. 2012.

[54] Biotronik. EHR Overview and Integration. http://www.biotronik.de/wps/wcm/connect/en_de_web/biotronik/sub_top/healthcareprofessionals/products/home_monitoring?p=http://www.biotronik.de/wps/wcm/connect/en_de_web/biotronik/sub_top/healthcareprofessionals/products/home_monitoring/HM_EHR_DataSync/product_page_ehr_overview&pw=974&pt= . 2013.

[55] Biotronik: **Biotronik EHR DataSync® - EHR Integration-Overview – Frequently Asked Questions.** Biotronik SE & Co. KG, Berlin, Germany. 2011.

[56] MedWow-Global Medical Equipment platform. http://www.medwow.com/used-pacemaker-tester/guidant/zoom-latitude/660576955.item . 2013.

[57] Boston Scientific: **Gebrauchsanweisung Zoom® Latitude® Programmier-/Aufzeichnungs-/Überwachungsgerät.** Boston Scientific Corporation, MN, USA. 2011.

[58] Boston Scientific: **Model 3120 Zoom® Latitude® Programmer Maintenance.** Boston Scientific Corporation, MN, USA. 2008.

[59] Boston Scientific: **Saving Patient Data to Floppy Disk.** Boston Scientific Corporation, MN, USA. 2011.

[60] Boston Scientific: **Saving and Managing Patient Data on a USB Pen Drive.** Boston Scientific Corporation, MN, USA. 2011.

[61] Oracle – The Java Language Environment. http://www.oracle.com/technetwork/java/index-136113.html . 2013.

[62] Oracle – Documentation. http://docs.oracle.com . 2013.

[63] TEIA – Internet Akademie und Lehrbuch Verlag. Sicherheit in Java. http://www.teialehrbuch.de/Kostenlose-Kurse/JAVA/6535-Sicherheit.html . 2013.

[64] JSON – Introduction and Documentation. http://www.json.org . 2013.

[65] David Kuhlman: A Python Book: Beginning Python, Advanced Python, and Python Exercises. http://cutter.rexx.com/~dkuhlman/python_book_01.html#part-1-beginning-python . 2013.

[66] DOM and SAX – Differences and how to choose between them. http://geekexplains.blogspot.co.at/2009/04/sax-vs-dom-differences-between-dom-and.html . 2013.

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
AV	Atrioventricular
BPEG	British Pacing and Electrophysiology Group
CARE-HF	Cardiac Resynchronization in Heart Failure
CHF	Congestive Heart Failure
CIED	Cardiovascular Implantable Electronic Device
COMPANION	Comparison of Medical Therapy, Pacing and Defibrillation in Chronic Heart Failure
CRT	Cardiac Resynchronization Therapy
DOM	Document Object Model
ECG	Electrocardiogram
EHR	Electronic Health Record
EHRA	European Heart Rhythm Association
EMR	Electronic Medical Record
EPR	Electronic Patient Record
ESC	European Society of Cardiology
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HF	Heart Failure
HIS	Hospital Information System
HIV	Human Immunodeficiency Virus
HRS	Heart Rhythm Society
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICD	Implantable Cardioverter Defibrillator
ICM	Implantable Cardiovascular Monitor
IDC	Implantable Device, Cardiac
IEEE	Institute of Electrical and Electronics Engineers
IHE	Integrating the Healthcare Enterprise
ILR	Implantable Loop Recorder
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization
JAR	Java Archive
JDK	Java Development Kit
JNLP	Java Network Launching Protocol
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KAGes	Steiermärkische Krankenanstaltengesellschaft mbH
LAN	Local Area Network
MDC	Medical Device Communication
MLLP	Minimal Lower Layer Protocol

NASPE	North American Society of Pacing and Electrophysiology
PDF	Portable Document Format
PICARD	Platform for Integrated Cardiac Rhythm Disease Management
PM	Pacemaker
POC	Point of Care
PRM	Programmer, Recorder, Monitor
RFD	Retrieve Form for Data Capture
SA	Sinoatrial
SAX	Simple API for XML
SIG	Bluetooth Special Interest Group
SOAP	Simple Object Access Protocol
SPRINT	Standardized Programmer Interface
TTM	Transtelephonic Monitoring
USB	Universal Serial Bus
XML	Extensible Markup Language

Appendix

Figure 28: Example for Biotronik IEEE11073 structure after the CARDDAS file was converted with the Biotronik Converter [54].

```
- <biotronik-ieee11073-export format-version="2.0" creator="BioHMSC" creator-version="Stage L RC3 NB_20100823">
  - <dataset>
    - <section name="MDC">
      - <section name="IDC">
        - <section name="PG">
            <value name="TYPE" type="MDC_IDC_ENUM_DEVICE_TYPE">IPG</value>
            <value name="MODEL" type="String">Evia DR-T</value>
            <value name="SERIAL" type="String">66000743</value>
            <value name="MFG" type="MDC_IDC_ENUM_MFG">BIO</value>
            <value name="IMPLANT_DT" type="DateTime">20090329T220000</value>
          </section>
        - <section name="SESS">
            <value name="DTM" type="DateTime">20101001T023954+0200</value>
            <value name="TYPE" type="MDC_IDC_ENUM_SESS_TYPE">RemoteScheduled</value>
            <value name="REPROGRAMMED" type="MDC_IDC_ENUM_SESS_REPROGRAMMED">NO</value>
            <value name="DTM_PREVIOUS" type="DateTime">20100511T121038</value>
            <value name="TYPE_PREVIOUS" type="MDC_IDC_ENUM_SESS_TYPE">InClinic</value>
            <value name="CLINICIAN_NAME" type="String">b t</value>
            <value name="CLINICIAN_CONTACT_INFORMATION" type="String">Fax: +12345</value>
            <value name="CLINIC_NAME" type="String">any clinic</value>
          </section>
        - <section name="MSMT">
          - <section name="BATTERY">
              <value name="DTM" type="DateTime">20100625T023000</value>
            </section>
          + <section name="LEADCHNL_RV"></section>
          + <section name="LEADCHNL_RA"></section>
          </section>
        - <section name="SET">
          - <section name="LEADCHNL_RV">
            + <section name="SENSING"></section>
            - <section name="PACING">
                <value name="AMPLITUDE" type="Numeric" unit="V">4.2</value>
                <value name="PULSEWIDTH" type="Numeric" unit="ms">0.4</value>
                <value name="POLARITY" type="MDC_IDC_ENUM_POLARITY">UNI</value>
              </section>
            </section>
          + <section name="LEADCHNL_RA"></section>
          + <section name="BRADY"></section>
          </section>
        + <section name="STAT"></section>
        </section>
      + <section name="ATTR"></section>
      </section>
    + <section name="BIO"></section>
    </dataset>
  </biotronik-ieee11073-export>
```

Mapping table of Biotronik specific parameters (right column) to PICARD 2.0 specific parameters (left column):

Table 6: MS Excel mapping table of the PICARD and Biotronik IEEE11073 parameters

PICARD parameter	Biotronik IEEE11073 parameter
lead_a_active	MDC/IDC/LEAD/CONNECTION_STATUS + MDC/IDC/LEAD/LOCATION
lead_vr_active	MDC/IDC/LEAD/CONNECTION_STATUS + MDC/IDC/LEAD/LOCATION
lead_vl_active	MDC/IDC/LEAD/CONNECTION_STATUS + MDC/IDC/LEAD/LOCATION
interval	MDC/IDC/SESS/DTM + MDC/IDC/SESS/DTM_PREVIOUS
telemonitoring	MDC/IDC/SESS/TYPE
pm_magnet_effect	MDC/IDC/MSMT/BATTERY/STATUS
icd_magnet_effect	MDC/IDC/MSMT/BATTERY/STATUS
battery_voltage	MDC/IDC/MSMT/BATTERY/VOLTAGE + MDC/IDC/MSMT/BATTERY/DTM
expected_lifetime	(MDC/IDC/MSMT/BATTERY/REMAINING_LONGEVITY || MDC/IDC/MSMT/BATTERY/REMAINING_PERCENTAGE) + MDC/IDC/MSMT/BATTERY/DTM
sensing_ra	MDC/IDC/MSMT/LEADCHNL_RA/SENSING/INTR_AMPL_MIN || MDC/IDC/MSMT/LEADCHNL_RA/SENSING/INTR_AMPL_MEAN || MDC/IDC/MSMT/LEADCHNL_RA/SENSING/INTR_AMPL_MAX
sensing_rv	MDC/IDC/MSMT/LEADCHNL_RV/SENSING/INTR_AMPL_MIN || MDC/IDC/MSMT/LEADCHNL_RV/SENSING/INTR_AMPL_MEAN || MDC/IDC/MSMT/LEADCHNL_RV/SENSING/INTR_AMPL_MAX
sensing_lv	MDC/IDC/MSMT/LEADCHNL_LV/SENSING/INTR_AMPL_MIN || MDC/IDC/MSMT/LEADCHNL_LV/SENSING/INTR_AMPL_MEAN || MDC/IDC/MSMT/LEADCHNL_LV/SENSING/INTR_AMPL_MAX
threshold_ra	MDC/IDC/MSMT/LEADCHNL_RA/PACING_THRESHOLD/AMPLITUDE
threshold_rv	MDC/IDC/MSMT/LEADCHNL_RV/PACING_THRESHOLD/AMPLITUDE
threshold_lv	MDC/IDC/MSMT/LEADCHNL_LV/PACING_THRESHOLD/AMPLITUDE
threshold_time_ra	MDC/IDC/MSMT/LEADCHNL_RA/PACING_THRESHOLD/PULSEWIDTH
threshold_time_rv	MDC/IDC/MSMT/LEADCHNL_RV/PACING_THRESHOLD/PULSEWIDTH
threshold_time_lv	MDC/IDC/MSMT/LEADCHNL_LV/PACING_THRESHOLD/PULSEWIDTH
lead_impedance_ra	MDC/IDC/MSMT/LEADCHNL_RA/IMPEDANCE/VALUE
lead_impedance_rv	MDC/IDC/MSMT/LEADCHNL_RV/IMPEDANCE/VALUE
lead_impedance_lv	MDC/IDC/MSMT/LEADCHNL_LV/IMPEDANCE/VALUE
lead_impedance_hv	MDC/IDC/MSMT/LEADHVCHNL/IMPEDANCE + MDC/IDC/MSMT/LEADHVCHNL/DTM_END
icd_mode	MDC/IDC/SET/BRADY/MODE
pacing_rate_lower	MDC/IDC/SET/BRADY/LOWRATE
pacing_rate_upper	MDC/IDC/SET/BRADY/MAX_SENSOR_RATE
av_delay_sens	MDC/IDC/SET/BRADY/SAV_DELAY_LOW || MDC/IDC/SET/BRADY/SAV_DELAY_HIGH
av_delay_pace	MDC/IDC/SET/BRADY/PAV_DELAY_LOW || MDC/IDC/SET/BRADY/PAV_DELAY_HIGH
pulse_amplitude_a	MDC/IDC/SET/LEADCHNL_RA/PACING/AMPLITUDE
pulse_amplitude_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/AMPLITUDE

pulse_amplitude_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/AMPLITUDE
pulse_width_a	MDC/IDC/SET/LEADCHNL_RA/PACING/PULSEWIDTH
pulse_width_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/PULSEWIDTH
pulse_width_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/PULSEWIDTH
polarity_sensing_a	MDC/IDC/SET/LEADCHNL_RA/SENSING/POLARITY
polarity_sensing_rv	MDC/IDC/SET/LEADCHNL_RV/SENSING/POLARITY
polarity_sensing_lv	MDC/IDC/SET/LEADCHNL_LV/SENSING/POLARITY
polarity_pace_a	MDC/IDC/SET/LEADCHNL_RA/PACING/POLARITY
polarity_pace_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/POLARITY
polarity_pace_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/POLARITY
vt_zone_one_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
vt_zone_one_unit	from IEEEObject
detection_vt_zone_one	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
vt_zone_one_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
vt_zone_two_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
vt_zone_two_unit	from IEEEObject
detection_vt_zone_two	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
vt_zone_two_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
vt_zone_fast_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
vt_zone_fast_unit	from IEEEObject
detection_vt_zone_fast	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
vt_zone_fast_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
vf_zone_from	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
vf_zone_unit	from IEEEObject
detection_vf_zone	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
vf_zone_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
pm_egm_abnormality	MDC/IDC/STAT/EPISODE/TYPE + MDC/IDC/STAT/EPISODE/TOTAL_COUNT + MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_START + MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_END
icd_egm_abnormality	MDC/IDC/STAT/EPISODE/TYPE + MDC/IDC/STAT/EPISODE/TOTAL_COUNT + MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_START + MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_END
percent_stim_a	MDC/IDC/STAT/BRADY/RA_PERCENT_PACED
percent_stim_v	MDC/IDC/STAT/BRADY/RV_PERCENT_PACED
events_reported	MDC/IDC/STAT/EPISODE/TOTAL_COUNT
events_shock_2_per_month	MDC/IDC/STAT/TACHYTHERAPY/SHOCKS_DELIVERED_RECENT + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START

events_shock_2_per_week	MDC/IDC/STAT/TACHYTHERAPY/SHOCKS_DELIVERED_RECENT + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START
events_shock_3_per_day	MDC/IDC/STAT/TACHYTHERAPY/SHOCKS_DELIVERED_RECENT + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START + MDC/IDC/STAT/TACHYTHERAPY/RECENT_DTM_START
events_n_vt_sustained	MDC/IDC/STAT/EPISODE/TYPE + MDC/IDC/STAT/EPISODE/TOTAL_COUNT + MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_END
events_n_vt_notsustained	MDC/IDC/STAT/EPISODE/TYPE + MDC/IDC/STAT/EPISODE/TOTAL_COUNT MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_END
events_n_vf_notsustained	MDC/IDC/STAT/EPISODE/TYPE + MDC/IDC/STAT/EPISODE/TOTAL_COUNT MDC/IDC/STAT/EPISODE/TOTAL_COUNT_DTM_END
reprog_mode	MDC/IDC/SET/BRADY/MODE
reprog_pacing_rate_lower	MDC/IDC/SET/BRADY/LOWRATE
reprog_pacing_rate_upper	MDC/IDC/SET/BRADY/MAX_SENSOR_RATE
reprog_av_delay_sens	MDC/IDC/SET/BRADY/SAV_DELAY_LOW || MDC/IDC/SET/BRADY/SAV_DELAY_HIGH
reprog_av_delay_pace	MDC/IDC/SET/BRADY/PAV_DELAY_LOW || MDC/IDC/SET/BRADY/PAV_DELAY_HIGH
reprog_pulse_amplitude_a	MDC/IDC/SET/LEADCHNL_RA/PACING/AMPLITUDE
reprog_pulse_amplitude_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/AMPLITUDE
reprog_pulse_amplitude_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/AMPLITUDE
reprog_pulse_width_a	MDC/IDC/SET/LEADCHNL_RA/PACING/PULSEWIDTH
reprog_pulse_width_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/PULSEWIDTH
reprog_pulse_width_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/PULSEWIDTH
reprog_polarity_sensing_a	MDC/IDC/SET/LEADCHNL_RA/SENSING/POLARITY
reprog_polarity_sensing_rv	MDC/IDC/SET/LEADCHNL_RV/SENSING/POLARITY
reprog_polarity_sensing_lv	MDC/IDC/SET/LEADCHNL_LV/SENSING/POLARITY
reprog_polarity_pace_a	MDC/IDC/SET/LEADCHNL_RA/PACING/POLARITY
reprog_polarity_pace_rv	MDC/IDC/SET/LEADCHNL_RV/PACING/POLARITY
reprog_polarity_pace_lv	MDC/IDC/SET/LEADCHNL_LV/PACING/POLARITY
reprog_vt_zone_one_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
reprog_vt_zone_one_unit	from IEEEObject
reprog_detection_vt_zone_one	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
reprog_vt_zone_one_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
reprog_vt_zone_two_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
reprog_vt_zone_two_unit	from IEEEObject
reprog_detection_vt_zone_two	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
reprog_vt_zone_two_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
reprog_vt_zone_fast_to	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
reprog_vt_zone_fast_unit	from IEEEObject

reprog_detection_vt_zone_fast	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
reprog_vt_zone_fast_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
reprog_vf_zone_from	MDC/IDC/SET/ZONE/DETECTION_INTERVAL + MDC/IDC/SET/ZONE/TYPE
reprog_vf_zone_unit	from IEEEObject
reprog_detection_vf_zone	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE + MDC/IDC/SET/ZONE/DETECTION_DETAILS
reprog_vf_zone_therapy	MDC/IDC/SET/ZONE/STATUS + MDC/IDC/SET/ZONE/TYPE
ecg_base_heartrate	MDC/IDC/STAT/HEART_RATE/ATRIAL_MIN || MDC/IDC/STAT/HEART_RATE/ATRIAL_MAX || MDC/IDC/STAT/HEART_RATE/ATRIAL_MEAN || MDC/IDC/STAT/HEART_RATE/VENTRICULAR_MIN || MDC/IDC/STAT/HEART_RATE/VENTRICULAR_MAX || MDC/IDC/STAT/HEART_RATE/VENTRICULAR_MEAN

Deutsche Fassung:

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)