Method and System for Standardized and Platform Independent Medical Data Information Persistence in Telemedicine

M Struck[1], S Pramatarov[2], C Weigand[1]

[1]Fraunhofer Institute for Integrated Circuits, Germany
[2]Chair of Medical Informatics, Germany

Abstract

Wireless communication between sensors monitoring patient vital signs has become more and more important in the past few years. Essential requirements to integrate sensors of different manufacturers into a clinical network are standardized communication protocols and a unique data representation of the vital signs. Both issues are realized by CEN ENV 13734/35 "Vital Signs Information Representation" (VITAL) [1]. The standard was implemented and integrated into a generic framework with different interfaces allowing integration of extensions [2]. In order to guarantee readability of vital signs communicated with the VITAL framework in the future, standardized storing methods are indispensable. That is why a new user interface was created that allows standardized persistence of medical data information in real time. Focussing on our implementation, we evaluated three relevant file formats: the "European Data Format" (EDF/EDF+), the "File Exchange Format" (FEF) and SCIPHOX.

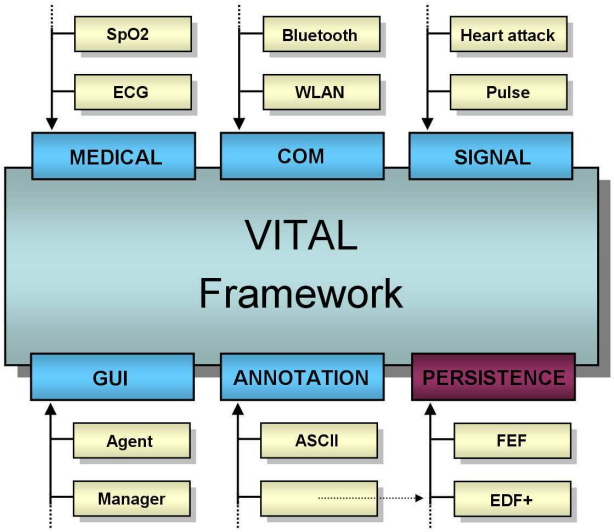

Figure 1. Interfaces of the VITAL Framework

1. Introduction

The trend of sensors acquiring patient vital signs is from the isolated ones towards a network of wireless connected medical devices. The purpose of the CEN ENV13734/35 standard for "Vital Signs Information Representation" (VITAL) is to define an object oriented model of medical devices and vital signs that are identified by a unique nomenclature. Furthermore, it specifies a communication model that consists of protocols and services based on the ISO/OSI [3] standard. The implementation of the VITAL standard was integrated into a software framework with different interfaces (see fig. 1).

These are interfaces for the integration of medical devices (ECG, temperature etc.), communication protocols, e.g. Bluetooth, WLAN and ZigBee [4], graphical user interfaces to visualize vital signs in an individual manner and signal processing units. Signal processing extensions analyzing physiologic parameters of patients can support physicians in their diagnosis. The automatic detection of cardiac infarctions or other cardiovascular diseases from the ECG are typical examples. The signal processing interface can also be used to derive important further parameters from the measured ones, e.g. the heart rate by computing the differences between the QRS complexes [5]. Furthermore, there is an interface allowing physicians to annotate continuous signal streams in real time. This interface gives them the opportunity to log important events. All the extensions can be realized by implementing the described interfaces and integrating them as plug-ins into the framework. Up to now, there is no possibility to store medical data information communicated with the VITAL framework. Standardized medical data persistence is mandatory for both, data exchange between different medical institutions and later observations made by physicians. In order to achieve interoperability, standardized file formats have to be implemented.

2. Methods

2.1. Interface requirements

In addition to the obligatory interface requirements, such as the declaration of abstract methods describing their specific challenge, the persistence interface has to deal with the following advanced aspects: The interface must be completely independent of any particular hardware properties and operating system characteristics. Therewith handling of the framework with each software system as well as data exchange between different healthcare providers is warranted. Another issue relates to the amount of information lost by a hardware respectively software crash. In this case, the interface should minimize data loss. The last aspect referring to the claims of the persistence interface concerns the need of handling with possible alerts, e.g. a low battery status, and with real time annotations made by a physician. That is the reason why annotation and persistence interface have to be combined.

2.2. Implementation

In order to handle with all those requirements, the interface was mainly separated into four abstract methods (realized in C++, we call them pure virtual methods) [6]:

```
class IPersistence
{
    virtual void writeMDS(MDS*) = 0;
    virtual void writeMetric(Metric*) = 0;
    virtual void writeAlert(AlertMonitor*) = 0;
    virtual void writeAnno(string&, QTime&) = 0;
};
```

The first method determines the *Medical Device System* (MDS) from the VITAL object hierarchy stored in the so-called *Medical Data Information Base* (MDIB) [1, 2]. This information is mandatory to organize the file structures. Furthermore, the method stores the patient demographic data (name, attending physician, sex etc.), which can be entered by an individual graphical user interface. The second method is called whenever the intern buffer storing the patient vital signs is full and can be displayed. Consequently, medical data persistence and monitoring of signal streams are progressed in parallel and continuously. Therewith a minimum of information loss can be achieved when having a system crash. The methods *writeAlert(...)* and *writeAnno(...)* write alerts respectively annotations on the storage medium. In order to get the actual time required for annotations, we use the cross-platform *QTime* class from the free *Qt library* by *Trolltech* [7]. The pure virtual methods have to be implemented by each specific file format and realized as plug-ins. Afterwards the plug-ins can be integrated into the VITAL framework.

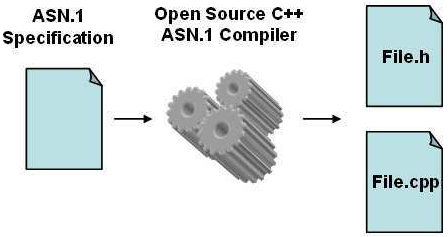

Figure 2. Automatic generation of C++ source code from a programming language independent ASN.1 specification

2.3. Abstract Syntax Notation One

The *Abstract Syntax Notation One* (ASN.1) is a standard that defines a formalism for the specification of abstract data structures independent of particular programming languages [8]. After specifying those structures, an ASN.1 compiler generates the source code for an arbitrary language (C/C++, Java etc.) (see fig. 2). In addition, the source code produced by the compiler provides serialization and deserialization methods. Those methods solve the byte order problem and, consequently, data structures can be written to hard disc or transferred over a network without implementing any further operations. There are a variety of methods to encode abstract data types. The choice depends on the available space as well as on the required efficiency and interoperability.

2.4. Standard encoding rules for ASN.1

In order to get an abridgement of the existing standard encoding rules based on the ASN.1 consider figure 3 [9]. The concrete rules for converting the specified data structures into bytes is given, e.g. by the *Basic Encoding Rules* (BER). The semantic can be generated either in XML or binary (see fig. 3). BER are commonly realized using the *type-length-value* or TLV encodings (see fig. 4). The field *type* describes the object class (kind of field) represented by the actual message part. The field *length* contains the size of the value field in bytes and the last field comprehends the information of the message.

2.5. Choice of file formats

Beyond the interface requirements described in section 2.1, there are additionally requirements to select between different file formats. One fundamental issue is their clinical extension. The existence of free tools visualizing the stored medical data information with good quality is another important item. Furthermore, the persistence files should allow fast access to all pieces of data and they should contain as less redundancy as possible.

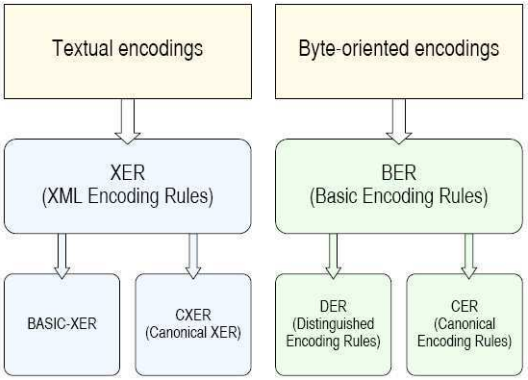

Figure 3. Standard encoding rules for ASN.1

Figure 4. Structure of the TLV encoding

2.6. Evaluation of file formats

The *European Data Format* (EDF) is a simple format specified for exchange and persistence of multi channel biological signals [10]. Its major benefits are the high extension in hospitals and the availability of many non-commercial viewing tools such as *Polyman* [10]. Unfortunately, it was not developed for storage of interrupted recordings of data streams. An extension of the EDF is the *European Data Format Plus* (EDF+). Both formats consist of a header followed by the data records. In contrast to the previous version, EDF+ can also deal with interrupted signals within one single file. Additionally, one of the channels can be encoded to store annotations and alerts, e.g. the battery status. The *File Exchange Format* (FEF) is more generic than EDF/EDF+. It is completely based on the VITAL CEN ENV 13734/35 [1] and is also a standard: ENV 14271 [11]. FEF uses the same nomenclature and *Domain Information Model* (DIM) [2] as the VITAL standard. That is the reason why persistence using FEF enables mapping of all the vital objects to the storage medium without information loss. In addition, FEF defines a multimedia section. This section may contain multimedia data, e.g. X-Ray images. In contrast to EDF+ and FEF, the *Standardized Communication of Information in Physician Offices and Hospitals using XML* standard (SCIPHOX) [12] is based on the XML syntax with the big advantage that the file contents can be interpreted by arbitrary XML parsers. Nevertheless, the redundancy encoding each digit of numerical data by one byte is high. Consequently, SCIPHOX concentrates on demographic and administrative medical data [6].

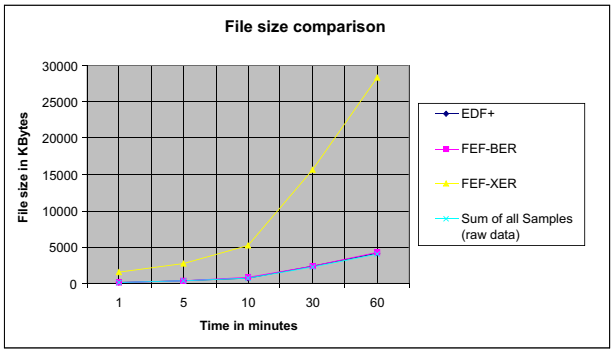

Figure 5. Comparison of file sizes

3. Results

We presented a generic VITAL framework interface for storage of medical data information. The interface can deal with several signals containing patient vital signs in real time. Different file formats can be integrated as plug-ins. The design and implementation of the persistence interface assure a minimum of information loss as well as consistent and readable file formats when having a system crash. We evaluated different file standards that can be implemented using an ASN.1 compiler and corresponding encoding rules. FEF has the advantage to store the complete information represented by VITAL objects. But EDF+ can be considered to be nearly a standard in hospitals. Figure 5 shows the progress of file sizes over time. Due to the large overhead storing vital signs, we skipped SCIPHOX and compared the sizes of the evaluated formats with a binary file including the raw data of the vital signs.

Figure 5 demonstrates that there are significant size differences between FEF-XER and the other formats. At the beginning the reason is stated by the complete storage of patient demographic characteristics and medical device information in XML. Later on the XML tags causes the bigger file size. The lower bound of the file size is given by the raw data file that includes neither annotations nor alerts. The marginal discrepancy between FEF-BER and EDF+ is given by the fact that EDF+ ignores some VITAL specific information such as the nomenclature. Another reason is the diverse space reservation for annotations and a different handling with alerts [6].

4. Discussion and conclusions

Due to its consistence with the medical communication standard VITAL, data persistence using FEF is an adequate solution. Unfortunately, the standard was first published after many stand-alone applications had already been introduced into the clinical workflow. That is the reason why it has not been used very often in practical applications up

to now. Consequently, it is an momentous criterion that the persistence interface is able to deal with different file standards. In order to guarantee both, full information acquisition of the VITAL standard and also its immediate practical usage, it is meaningful to persistence the data with FEF and EDF+. Therewith full information persistence as well as compatibility with actual clinical applications can be safeguarded.

Acknowledgements

Special thanks go to Prof. Dr. Hans-Ulrich Prokosch, head of the chair of medical informatics of the university Erlangen-Nuremberg, supervising this challenging topic.

References

[1] Zywietz C. Standardized Representation of Vital Signs for Continuous Care in Cardiology. IEEE Computers in Cardiology, 1998; 205–208.

[2] Weigand C. Use and Implementation of a Medical Communication Standard in Practice. IEEE Computers in Cardiology, 2005; 319–322.

[3] Tanenbaum AS. Computernetzwerke. 4. edition. Pearson Studium, 2003.

[4] Hofmann C, Weigand C, Bernhard J. Wireless medical sensor networks with ZigBee. Applied Informatics and Communications, 2006; .

[5] Trahanias PE. An Approach to QRS Complex Detection Using Mathematical Morphology. IEEE Transactions on Biomedical Engineering, 1993; .

[6] Pramatarov S. Implementierung einer Schnittstelle für den medizinischen Kommunikationsstandard VITAL zur Persistierung von Echtzeit-Datenströmen, 2007.

[7] Blanchette J. C++ GUI Programming with Qt 4. first edition. Prentice Hall International, 2006.

[8] Dubuisson O. ASN.1 - Communication between heterogeneous systems. first edition. Morgan Kaufmann Publishers, 2000.

[9] Walkin L. Open Source ASN.1 Compiler Data Sheet. URL `http://lionet.info/asn1c`.

[10] Kemp B. Formal specifications of EDF and EDF+. URL `http://www.edfplus.info/specs`.

[11] Värri A. ENV 14271, File Exchange Format for Vital Signs and its use in digital ECG archiving. OpenECG Workshop Integration of the ECG into EHR and Interoperability of ECG Device Systems, 2004; .

[12] Heitmann KU, Gehlen E. Arbeitsgemeinschaft SCIPHOX GbR mbH. URL `http://sciphox.hl7.de/`.

Address for correspondence:

Matthias Struck
Am Wolfsmantel 33
91058 Erlangen, Germany
Matthias.Struck@iis.fraunhofer.de