Journal of Computer Science 8 (10): 1601-1614, 2012
ISSN 1549-3636
© 2012 Science Publications

Applying Software Engineering Methodology for Designing Biomedical Software Devoted to Electronic Instrumentation

[1]Gilsa Aparecida de Lima Machado, [1]Patricia Mara Danella Zacaro,
[1]Alderico Rodrigues de Paula Junior and [2]Marcelo Lopes de Oliveira e Souza
[1]Research and Development Institute (IPD)
Course of Biomedical Engineering, University of Vale do Paraiba (UNIVAP),
Av. Shishima Hifumi, 2911, 12244-000, S. Jose dos Campos, SP, Brazil
[2]Course of Space Engineering and Technology (ETE),
Division of Space Mechanics and Control (DMC),
National Institute for Space Research (INPE),
Av. dos Astronautas, 1758, 12227-010, S. Jose dos Campos, SP, Brazil

Abstract: Problem statement: Significant effort goes into the development of biomedical software, which is integrated with computers/processors, sensors and electronic instrumentation devoted to a specific application. However, the scientific work on electronic instrumentation controlled by biomedical software has not emphasized software development, instead focusing mainly on electronics engineering. The development team is rarely composed of Software Engineering (SE) experts. Usually, a commercial automated tools environment is not used due to its high cost and complexity for researchers from other areas to understand. **Approach:** This present study reports how the SE approach was applied to design and develop biomedical software, which is part of a Computerized Electronic Instrumentation (CEI). This CEI comprises software and an electronic instrumentation based on a force sensor and electrogoniometer to monitor the hand exertion of computer user during typing task. The aim is to serve as a guideline for academic researchers who are not expert in software engineering methodology but usually develop their own software to run with their CEI. The specification of the requirements, presented as use case, includes the context diagram, the data flow diagram, the entity relationship diagram and test procedure. The Unified Modelling Language from the Enterprise Architect tool was used. The developed software and the electronic instrumentation were tested together. **Results:** A sample of the interface screen shows how the outcomes could be plotted in an integrated manner. By comparing the values with other values obtained by manual calculations and with those provided by sensor manufacturer, the repeatability of test procedure validated the results. Reliable electronic instrumentation when working with unreliable software can become unreliable. **Conclusion:** Applying software engineering methodology principles provided a simple and clear documentation that was helpful to establish the test procedures and the re-work.

Keywords: Biomedical software, signal processing, software engineering, structured analysis

INTRODUCTION

The scientific work in electronic instrumentation controlled by biomedical software has not emphasized the software development, but instead focused on electronics engineering, probably, because the development team is rarely composed of Software Engineering (SE) experts. However, in this study, software design and development was the target instead of electronic instrumentation development. This should help researchers in academic environments to better understand where the SE methodology approaches fit when they do electronic instrumentation that involves software development.

Biomedical software and mission critical applications: A lot of software is developed by people who are not experts in Software Engineering (SE) methodologies. A significant amount of software is developed by small and medium size software organizations, which do not have infrastructure and resources to implement a rigorous quality plan (Mishra and Mishra, 2009). Most cannot afford automated tools for SE due to their high cost and complexity.

Therefore, design faults may occur as a result of imperfections from specification requirements. To avoid design faults, according to Troubitsyna (2010)

Corresponding Author: Gilsa Aparecida de Lima Machado, Course of Biomedical Engineering, Research and Development Institute (IPD), University of Vale do Paraíba (UNIVAP), Av. Shishima Hifumi, 2911, 12244-000, S. José dos Campos, SP, Brazil

while developing a system by refinement, developers start from an abstract specification. Stepwise refinement allows them to incorporate system requirements into the specification gradually and eventually arrive at system implementation, which is correct by construction. Design faults may cause errors and even failures in the system. Wrong results could be due to code implementation faults or faults in the specification.

Nevertheless, when a failure occurs in biomedical computerized electronic equipment, it is usually attributed to a protocol error, equipment failure, or human error during system operation. Rarely, is it attributed to a software fault that induced an equipment error. Early reliability models are based on reliability engineering, particularly hardware reliability (Shanmugapriya and Suresh, 2012). However, quality goals can primarily be achieved if the software architecture is evaluated with respect to its specific quality requirements at the early stage of software development (Shanmugapriya and Suresh, 2012).

Redundancy is especially important in systems with critical missions. Software can have redundancy of processes, while hardware can have redundancy of electronic components such as sensors or input/output channels. A system is defined in the biomedical field as having a critical mission, if its failure could have life threatening consequences. Several authors have reported on such systems. Arpaia *et al.* (2012) proposed a method for home-care to predict a critical condition of a patient affected by a specific disease such as pulmonary disease.

Systems engineering can autonomously decide the next step for patient care based on the previous set. Kauffmann *et al*. (2011) offered approaches for verification and validation processes. The greater the risks, the more efforts must be used to reduce the probability of system failures. Depending on the mission of the system, its failure can result in lost of human life. For example, Garcia-Saez e*t al.* (2009) discussed a solution for the problem of diabetes care based on architecture and implementation of a mobile personal assistant that supports personal and remote control strategies for insulin-dependent patients supervised by healthcare professionals through a telemedicine information system. The system notifies the patient with a message whenever a new therapy is prescribed. In this case, incorrect information would generate incorrect prescription, which could be ultimately lethal.

Considering that an incident can occur when unprepared staff operates complex biomedical system, all biomedical systems should have access control for each available process into the system. One solution reported by Rezk *et al.* (2012) makes it possible to mine the dependency among user's data items from his transaction log and generates specific rules, which determine the context in which the user access the data items. According to them the challenge is building an efficient Database Intrusion Detection System, which can detect any malicious transaction with low false positive rate and high detection rate and integrating it with access control, in order to strength the database security.

Two issues could minimize the problems reported above, one of them is the software engineering methodology; the other is the certification of software and electronic instrumentation.

Software certification: Software certification is not usually taught in a university or a computer science course. Therefore, few software developers are prepared to develop software ready to be submitted to a certification process. Certification of medical software is not a recent approach; however, few researchers and governments have made any effort to do this.

Certification allows software developers in the country to market safe products with better quality and, therefore, are more competitive. In addition, certification facilitates the entry of such software into international markets. For this purpose, the way that software packages are tested, delivered and operated should contain written document. The International Standard Organization (ISO)/12119:1994 is concerned about this. Additionally, the ISO 9001 and FDA medical device good manufacturing process regulations have roles that reduce the risk of low quality software production.

Recent scientific study addressing certification of biomedical software that works with electronc instrumentation was not found. However, certification of systems containing software is increasingly important for governments, industry and consumers alike (Maibaum and Wassyng, 2008).

Guidelines from a certifier company are needed for biomedical electronic instrumentation that involves software development. Each certifier company has their own roles based on specific ISO standard. Due to the lack of roles based on ISO standard that could help researchers to conceive and develop their software for biomedical instrumentation, this computerized electronic instrumentation has not been submitted to a certification process yet. Nevertheless applying the SE methodology approach was an advance.

Software engineering methodologies: There are several approaches to SE and each lead to a specific methodology of SE. Structured Analysis of systems is a SE methodology used by several software developers. Nakanish *et al*. (2009) had the Data Flow Diagram (DFD) as their focus. The Agile is about rapid software design and development. The Scrum, based on agile methods, could be the best solution for project with rapidly changes and at first, it does not emphasize solid requirements as the structured analysis does. The ArchJava is a tool designed to allow programmers

assemble components, connection and ports. An ArchJava component is a special object, able to be equipped with ports through which it can interact with other components. Ports are dedicated to support connections and service invocations. A port corresponds to the concept of interface found in COM port frequently used in communication between machines. Analysis and Object Oriented are different techniques to develop the computer system. The Object Oriented (OO) is the newest approach that focuses on capturing the objects in the scenario of the current system. A tool of modelling called UML (Unified modelling Language) serves for the use case and DFD definition that could be applied in both approaches of structured analysis and OO.

However, some researchers have developed biomedical software for CEI, but the focus of their manuscripts has been electronic instrumentation. Therefore, the reader or some new team member does not understand how the software was designed and developed. As a consequence, the system could become unrepeatable and not maintainable. Amato *et al*. (2009) developed a biomedical system using Java, relational MySQL DBMS, MATLAB and XML, but their SE design was not included. Pereira *et al*. (2009) developed instrumentation to apply and assess locomotor training. They constructed a system using electrogoniometers, load cell and software developed in LabVIEW® environment. The LabView is one software environment provided with its respective data acquisition card.

Normally, the researchers generate documents applied to the electronic instrumentation only, while the software documentation is stored only in the mind of the developer. When the project moves out of laboratory and onto the market, the software should not be inserted into the project documentation as a *"black box"*, but a systematic documentation of software is need to allow reproducibility of the project.

MATERIALS AND METHODS

Electronic instrumentation: The computerized electronic instrumentation is an integrated solution that proposes to record typing frequency, fingertip force and wrist posture (flexion, extension, ulnar and radial deviation) in computer user to detect hand overexertion, during typing task, that could put the computer user at risk of acquiring muscle skeletal injury.

Exactly how the electronic instrumentation was developed was addressed in our previous study: Machado and Villaverde (2011). Following is a brief description of electronic instrumentation.

The electronic instrumentation is comprised a two-axis capacitive sensor accelerometer±1.5g±90deg, with a 1Hz- 300Hz bandwidth response used as goniometer to register the wrist posture; and a Force Sensing Resistor (FSR), 4.4 N, with a response time of 5 µl, biased by dual ±9V and -5V power supplies used to register the applied force. An instrumented hand is shown in Fig. 1. This photograph was taken while a computer user was performing a typing test using the CEI.

The accelerometer is in the box positioned on the subject's right hand. The force sensor resembles an almost transparent ribbon. An empty box was attached to the subject's left hand to simulate the condition of the test hand; simulated force sensors were also placed on the left hand.

After the hands were properly instrumented, both sensor circuits (force and accelerometer) were energized and connected to a data acquisition card (with 12 bits and 32 channels from Lynxtec Technology-Brazil) inserted into a personal computer.

Figure 2 and 3 are the basic diagrams of circuit boards corresponding to the force sensors and the accelerometer, respectively. These diagrams show the output signal, which were acquired and processed by developed software.

The V_{out} in Fig. 2 is the output signal in response to a pressure that changes the sensor resistance (R_1) to a reference resistance (R_2).

The accelerometer sensor has two reading exits, one for the X-axis and another for Y-axis; Y_{out} is the output voltage of the accelerometer, in Y direction; X_{out} is the output voltage of the accelerometer, in X direction. The accelerometer was used as a goniometer to measure wrist flexion/extension and ulnar/radial deviation. The developed software acquired and processed the output signal in order to provide the correspondent angle for wrist posture during typing task.

The software requires calibration tables to execute calculations using the acquired signal. Therefore, the software has a process responsible for sensors calibration. The calibration task was described in the *Procedure for test* and in the *calibration outcomes* section. With more calibration points, more precise results could be obtained, because the calculations were based on a linear interpolation method (Eq. 1):

$$X_2 = X_1 + (a - Y_{X1})\frac{(X'_1 - X_1)}{(Y_{X'1} - Y_{x1})} \quad (1)$$

where, variable "a" represents V_{OUT} from Fig. 2 and V_{OUT}, X_{OUT} from Fig. 3; X_2 corresponds to the unknown fingertip force or wrist angle during typing task; while the interval value from the calibration table is X_1 with its respective tension Y_{X1} and X_1' with its respective tension $Y_{X'1}$. The X_2 values were saved as temporary data for graphic generation by developed software.

Fig. 1: View of the instrumented hand showing the position of the sensors

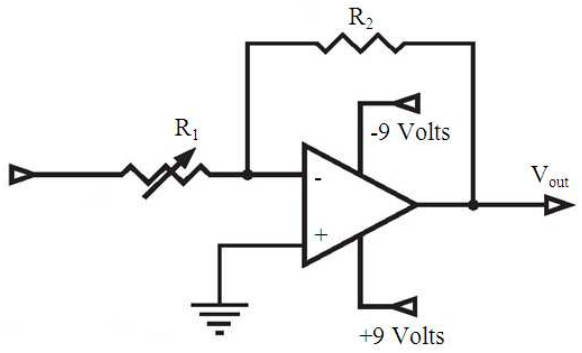

Fig. 2: Basic diagram for force sensor. Legend: R_1 = sensor resistance; R_2 = Reference resistance; V_{out} = output signal

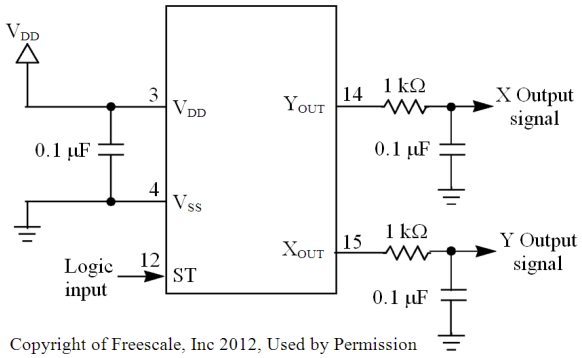

Fig. 3: Connection Diagram for Accelerometer. LEGEND: V_{DD}=the power supply input; V_{SS}=the power supply ground; ST=logic input pin used to initiate self test; X_{out}=output voltage of the accelerometer; Y_{out} =output voltage of the accelerometer

Software design: The structured analysis methodology was applied. The software requirements were described by use case. The use case diagram is commonly applied to the Object Oriented approach. For the structured analysis approach, the software requirements are described by narrative use case as a series of numbered steps and complemented by context diagram.

The relational database manager system Firebird®; the Delphi® language; Unified Modelling Language (UML) from the Enterprise Architect tool; a PC Computer with Microsoft Windows, a data acquisition card by Lynxtec Technology; and the electronic instrumentation previous described were used.

The first step of this methodology was the narrative use case as a series of numbered step of software requirements, present in the results section. The second step is the context diagram (Fig. 4) design that can be helpful in understanding the context that the system will be part of.

The next step was the Entity and Relationship Diagram (ERD) designing (Fig. 5-6). This ERD is a relational model to show a brief database composition and how the data are inter-correlated. Each entity was represented by a rectangle; while each data was defined as an attribute of its respective entity, known as table column. For each entity created, its attributes were defined and listed on the right side of Fig. 5 and represented by an ellipse in Fig. 6. The designed ERD had an indicator of its cardinality. The database modeling was performed using UML Data Modeling Profile. This tool maps the database concepts of tables and relationships onto the UML concepts of classes and association by using the stereotype.

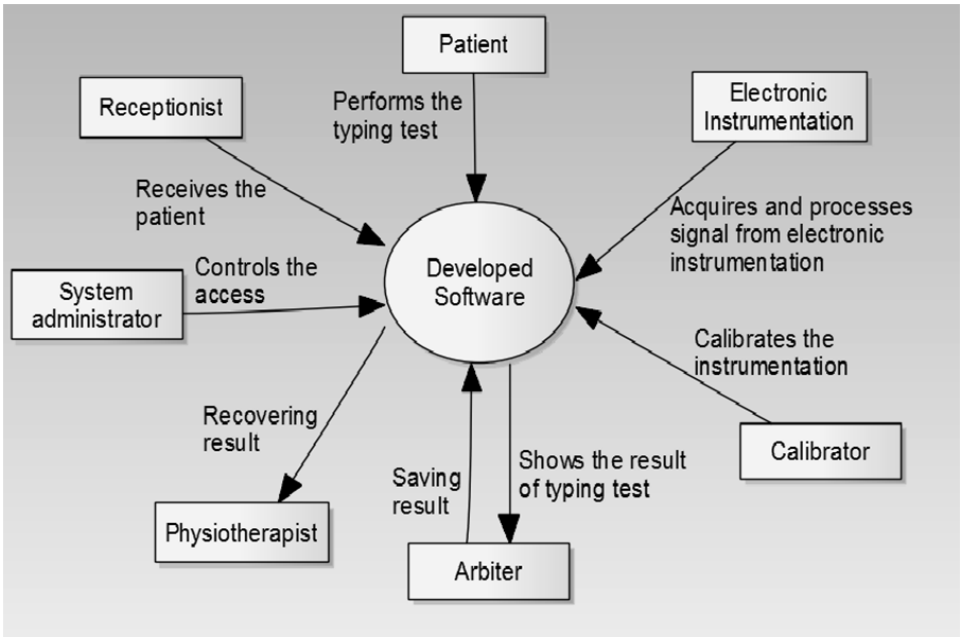

Fig. 4: Context diagram of the system

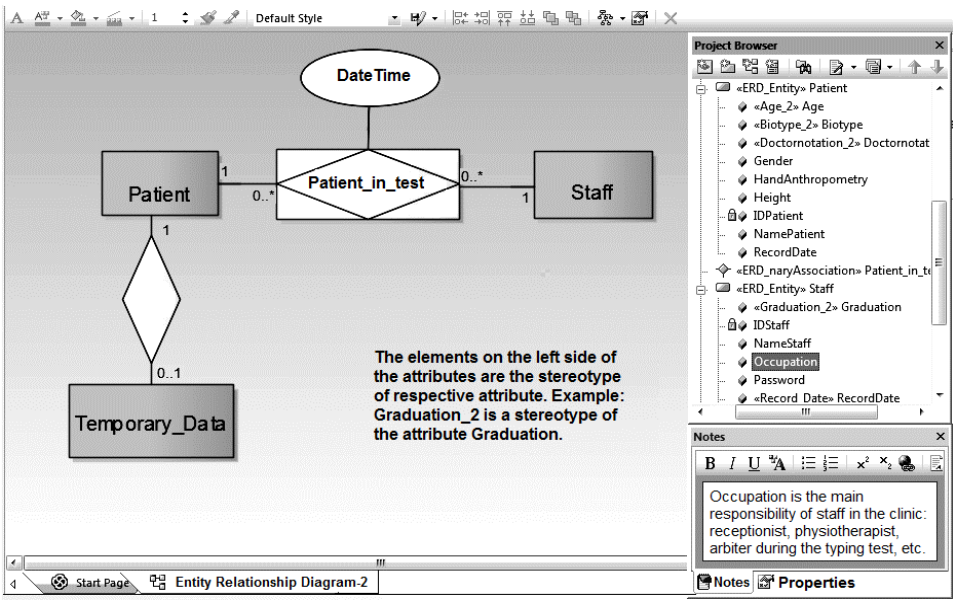

Fig. 5: The used tool case showing the starting of the entity and relationship diagram definition and its respective attributes

The fourth step of this SE methodology was to design the Data Flow Diagram (DFD), shown in Fig. 7, composed by process, data storages and external entities. This DFD helps visualize how the system should operate and how the system would be implemented. The used tool case provided the notation used for DFD design, where the ellipse represent a process in which data are used or generated; the rectangle represents an external source; the two parallel lines represents the stored data; and the arrow indicates the direction of data, i.e., how data flows through the system.

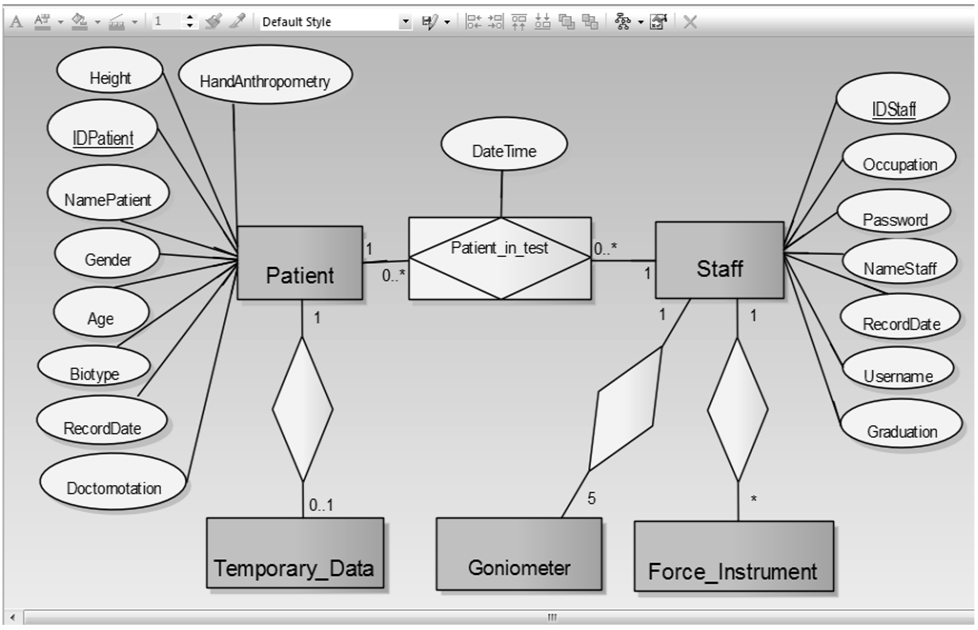

Fig. 6: Part of the entity and relationship diagram of the system

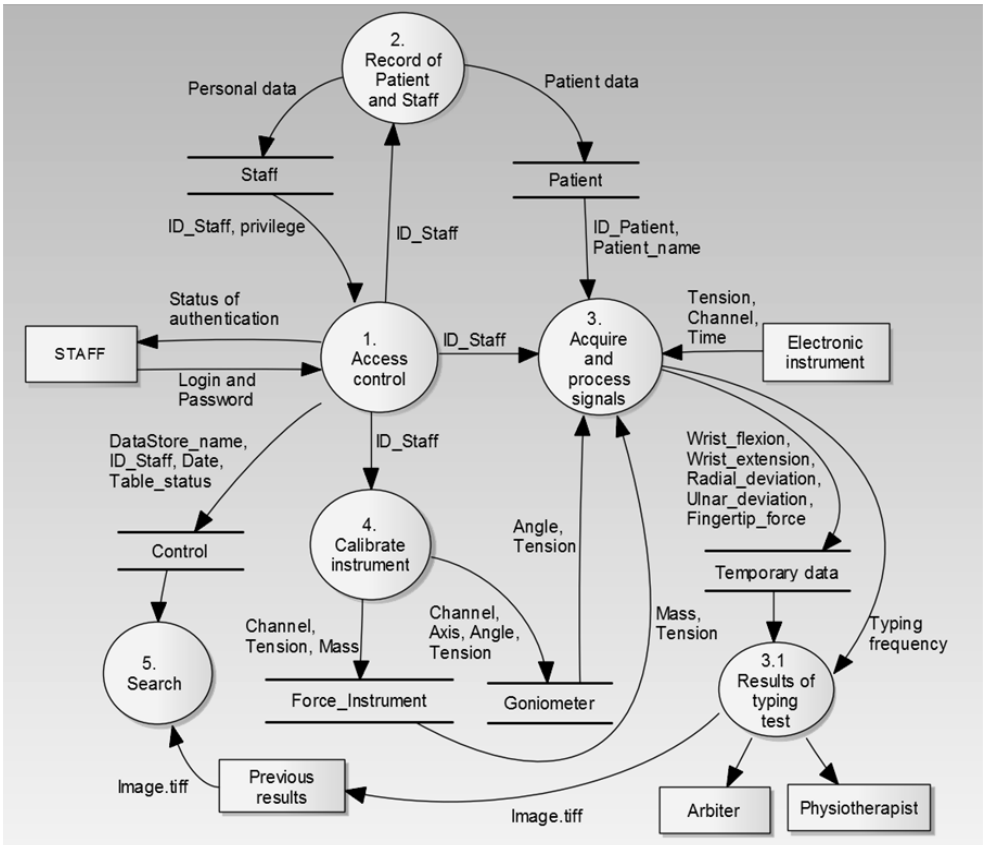

Fig. 7: Data flow diagram of the system-top level

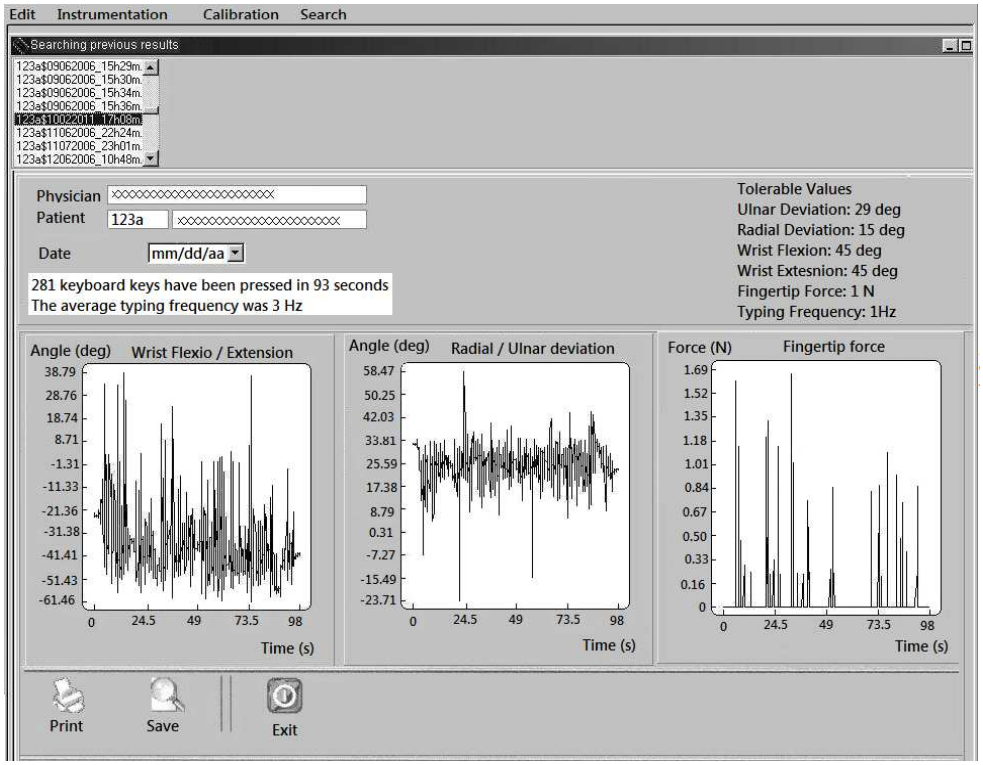

Fig. 8: How the outcomes were exhibited in an integrated manner. LEGEND: Deg=degree that refers to the threshold for wrist posture; N=Newton that refers to the fingertip force applied during typing; Hz =the average of keyswitch pressed in one second

The fifth step was the software implementation by code creation using the Delphi® language.

Code implementation: The driver of the acquisition card had its source code adapted to run as part of our software, to provide facilities to choose the channel and sample frequency, as well as to acquire and process data. Process 3 from Fig. 7 was programmed considering the pins where the sensors were connected to the data acquisition card. These signals were processed and stored in a Table as "Temporary-data" according to Fig. 5-7.

The access control process from Fig. 7 was the first to be implemented enabling and disenabling access to the other options of the system. Each process was implemented and tested in isolation. Finally, the integrated system was obtained when the electronic instrumentation was connected to the computer. Then, process 3 was tested to evaluate its capability to recognize the electronic instrumentation. Each channel from the data acquisition card was tested to assure the correct reading from each connected sensor.

The electronic instrumentation and software were tested together to evaluate the capability of the software to acquire and process data. The outcomes were compared to previous set values.

Procedures for test: The test team was composed of a computer scientist (software developer), electronic engineer and physiotherapist.

To check the access control and assure that all requirements were implemented, a person, who was unauthorized to access some source of the system, checked the security/safe condition. Two people were the system testers. Each logged in the system as administrator and all permission to the administrator profile was tested. This same person logged out as administrator and subsequently, logged in as receptionist; then the type of permissions granted her was tested. This same person logged out as receptionist and logged in as arbiter so she could setup the software configuration to recognize the electronic instrumentation, setup the channel from where the data would be acquire; choose the sample frequency before use; calibrate the electronic instrumentation; start a simulation of a test by pressing all force sensor and moving the electrogoniometer, stop the simulation; and observe how the outcomes were exhibited. The second tester repeated this sequence.

Table 1: The calibration table obtained during three trials with one force sensor

Masses (gram)	Trial-1 (volt) (volt)	Trial-2 (volt)	Trial-3 (volt)	Mean (volt)	SD
20	0.23	0.21	0.28	0.24	0.04
50	0.61	0.58	0.45	0.55	0.09
70	0.74	0.80	0.70	0.75	0.05
100	1.10	1.00	1.10	1.07	0.06
120	1.20	1.40	1.40	1.33	0.12
130	1.30	1.40	1.60	1.43	0.15
150	1.40	1.50	1.70	1.53	0.15
170	1.70	1.70	2.00	1.80	0.17
180	1.90	1.90	2.10	1.97	0.12
200	2.40	2.50	2.50	2.47	0.06
220	2.50	2.50	2.60	2.53	0.06
230	2.60	2.40	2.60	2.53	0.12
250	2.90	2.80	2.80	2.83	0.06
270	3.10	2.90	3.18	3.06	0.14
280	3.10	3.00	3.42	3.17	0.22
300	3.30	3.30	3.60	3.40	0.17
320	3.70	3.70	3.80	3.73	0.06
350	4.20	4.20	4.00	4.13	0.12
360	4.22	4.21	4.25	4.23	0.02
370	4.60	4.50	4.30	4.47	0.15
380	4.60	4.50	4.50	4.53	0.06
400	4.70	4.90	4.60	4.73	0.15
430	4.92	4.98	4.85	4.92	0.07

To validate the result provided by software and the electronic instrumentation working together, one force sensor was tested by using standard masses, when 23 calibrated points were obtained starting from 20-430 grams. For this purpose, the force sensor was connected to the data acquisition card inserted into the computer. The circuit with the force sensor was power supplied by an electronic device with a dual voltage of ±9V. The sensor, with a sensitive diameter, was placed on a flat area. The masses were placed on the sensitive perimeter of the sensor and the response was shown in tension through the software. When a pressure was exerted on the sensitive area, a reduction in its resistance occurred and, consequently, a voltage signal was released. To validate its reliability, the same force sensor was tested three times. The outcomes are presented in Table 1. The mean and standard deviation was calculated between the three trials to observe the error rate in order to estimate the reliability of the software in providing accurate results from electronic instrumentation in response to the fingertip force applied during typing task.

To obtain the calibration points from the electrogoniometer, it was connected to the computer, the software responsible for calibration task was set up and the electrogoniometer was positioned at 0° and ±90 degree for x-axis and y-axis. Then the outcomes were transferred to the calibration data storage labelled as Goniometer in Fig. 5-6. Five calibration points were recorded.

To check the reliability of this present software in calculating the typing frequency, no sensors were used; one subject typed a text using the computer keyboard for 90 s. Then, the system showed the typing frequency expressed in Hz, which was compared with manual calculation.

To check the accuracy of software in acquiring and processing signals, one computer user sat in an upright position with arm held at a flexion of 90 degree to the body. The forearm was in the resting position over a flat area, leaving the left hand free to move by flexing the wrist. The electrogoniometer was fixed onto the volunteer's hand, at the metacarpal junction of the middle finger and then this was connected to the computer through the data acquisition card. The software was configured at a sampling rate of 5 Hz. At the same time, a manual goniometer was placed on the side of the subject's hand, in the carpal area, with its movable spindle positioned at the edge of the little finger following the median line of the ulna. The subject did some wrist flexion movements, as requested by the arbiter. The manual goniometer followed the wrist flexion. The result from electrogoniometer was compared with the result from manual goniometer. This test procedure is the same reported by Machado and Villaverde (2011).

Later, the test procedure was set up with one computer user. For this purpose, the electronic instrumentation was positioned on the subject's right hand at the metacarpal junction of the middle finger (Fig. 1) to register wrist posture. With the subject's hand in a prone position, a flex force sensor was fixed with a ribbon sensor on the palmar aspect of fingers 2 through 5 in the medial phalanx region with the sensitive area of the sensor on the fingertip pulp to register the applied force. Fig. 1 shows an instrumented hand. After the hands were properly instrumented, both sensor circuits (force and accelerometer) were energized and connected to a data acquisition card inserted into a personal computer. The software was setup to enable the channels from data acquisition card and the sample frequency was established at 2000 Hz. The subject started typing a previously selected text and the software started to acquire and process data, ending as soon as the arbiter pressed the "Stop" button available on the interface screen.

The outcome was shown in an integrated manner on a computer screen, which is illustrated in Fig. 8. These included information about the typing frequency, wrist flexion and extension, radial and lunar deviation and fingertip force.

The outcomes were saved as *image.tiff*, as predicted in Fig. 7. The software offers a resource for recovering this image.

RESULTS

This methodology was applied in a simple form, which is understandable by anyone who intends to develop biomedical software for electronic instrumentation. The narrative use case, the context diagram, ERD, the DFD, the result of the calibration task and a sample of the outcomes provided by CEI during typing task in attendance with item 21 from use case are shown below.

Narrative use case of the software requirements: The system administrator creates the staff records for receptionist, arbiter, calibrator and physiotherapist to provide user authentication:

- One patient first arrives to do a test
- The system requires the receptionist authentication
- The system verifies the user authentication profile in order to enable or disable resources
- The system enables the receptionist to create the patient record
- The receptionist receives the patient by providing a patient record at his/her first time. The receptionist then logs out from the system
- The instrument calibrator logs in the system to obtain user authentication
- The system verifies the user profile to enable the screen interface for calibration task
- The instrument is connected to the computer
- The calibrator calibrates the force sensors and the electrogoniometer to actualize the database with the new calibration data and logs out of the system
- The system registers any changes made in the database
- The arbiter/physiotherapist logs into the system
- The record of respective patient who arrived to do the typing test is selected
- The subject sits upright in a chair with arms and knees flexed at 90 degree. The keyboard is approximately 10 cm from the edge of the table. The subject starts typing a previously selected text for 90s. The developed software obtains the typing frequency
- The typing frequency is saved by system
- The patient has the hand instrumented with force sensor and electrogoniometer attached
- The arbiter decides the duration of the test, selects the channel for signal acquisition and chooses the sample frequency. Then the patient starts the typing task
- The software starts the control of the channel/sensor, registers the signal in volts, records the reading time and transforms the values obtained in volts into the correct measurement units (degree, Newton and Hz).
- The result of calculation is retained in a temporary Table
- The arbiter administers the use of the instrument, tracks completion of the test and stops it by pressing a button available on user interface screen
- The system shows the outcomes on a screen interface. Data on flexion and extension of the wrist, radial and ulnar deviation, typing frequency and fingertip force during the typing are exhibited
- The arbiter saves or prints the result as an image
- The arbiter/physiotherapist and patient analyse the result together
- The system provides limits for wrist flexion and extension, lunar and radial deviation, fingertip force and typing frequency above which the patient is in risk of acquiring muscle skeletal disorder
- The physiotherapist researches previous result of the respective patient
- The patient receives guidance about extreme values observed
- The physiotherapist fills the electronic record with information related to the diagnostic conclusion
- The physiotherapist/ arbiter logs out from the system
- The system saves information about user name, date and database structure in which the respective user has just modified. The system allows the administrator to audit the database to assure data safety

Context diagram: The Context Diagram (Fig. 4) has just one process element representing the system being modelled, showing its relationship to external systems, person, equipment, or something that is necessary during software use.

The context diagram is composed of: Previous results-were saved in one folder into the hard disk of the computer where the screen of the outcomes was saved as *image.tiff*. The purpose of this is to track patient's improvement concerning hand overexertion.
Staff-people with authorization to operate the system and therefore register patients, be responsible for CEI use, search for previous results, analyse the outcomes. In summary, staff is a person who is not patient, but is involved in the typing test environment:

- Arbiter-health professional who is responsible for instrumentation use and analyses of the patient's test outcomes
- Electronic Instrumentation-the CEI composed by force sensor, electrogoniometer, data acquisition card and power supplier
- Patient-the computer users who will use the CEI solution to detect his/her vulnerabilities in acquiring muscle skeletal disorder
- System administrator-person responsible for database system and software performance and security
- Computer-used to run the developed software including the goniometer, force sensors and the data acquisition card. The typing task test was also performed on this computer

Entity and relationship diagram: There was no intention to write a tutorial; therefore, Fig. 5-6 is a brief demonstration of designed ERD. Consequentially, not all created entities were represented in Fig. 6. In Fig. 5, part of the tool case is shown to provide an idea how the entity and its respective attributes were created. In this case, the entity *Patient* and *Staff* are shown.

Figure 5 (b) is part of the Entity and Relationship Diagram of the system in a more advanced stage of creation than Fig. 5.

The diagrams in both Fig. 5-6 are not a flowchart. Therefore, the losangle does not represent decision made: this represents the relationship between the two entities. The relationships illustrate how two entities share information in the database structure. The relationship between entities Patient and Staff was the primary key from Patient and Staff, plus a new attribute labelled as *Date Time* that denotes the date of patient attendance. For each data storage in this ERD, there is an equivalent data storage in the DFD.

The underlined attribute (Fig. 6) represents a primary key. These keys were used for relationships between the data storages and warrant uniqueness.

The data storage named "Patient" has information about each patient. Whereas, the data storage named "Temporary_Data" contains processed signals obtained during typing task. The data storages named "Force_Instrument" and "Goniometer" were used for data storage from the instrument calibration; staff with calibrator profile is responsible for these table's contents.

This ERD diagram shows the attributes distributed among entities that represent the data storages, illustrated with rectangles. In reality, each data storage in ERD had an equivalent data storage in DFD; and each data storage generates one table in the Firefird® database. Two entities are shown on the left side of Fig. 6 to demonstrate how the attributes were designed by using an ellipse and line connector to the respective entity. The attributes of each respective entity are listed on the right side of Fig. 5.

The relationship named of "Patient_in_test" from Fig. 6 enables the system to provide information about patient's attributes and consultation date, as well as the attributes of involved staff. Additionally, when the staff has the health care profile defined by the occupation attribute, they have free access to the Temporary-Data.

There are many notation styles that express cardinality. The tool used for this ERD designing provides the used notation, where:

- 0..1 - zero or up to 1 instances, but no more than 1
- 0..* - zero, one or many instances
- 1..* - 1, or more than 1 instances
- - exactly 1 instances

Data flow diagram: To show the solution of the SE, the Data Flow Diagram (DFD) was designed with several levels. However, this study only shows the DFD in the top level of abstraction, i.e., level-1 composed by process, data storages and external entities.

The developed software was divided in many parts. Each part was responsible for one or more tasks from the use case. The DFD diagram in Fig. 7 shows each part of software defined as numbered process (1, 2, 3, 4 and 5).

Process-1 was created for user authentication and access management to each process in the system, guaranteeing set restrictions for system use. The user's profile determined which processes became available. Inserting a new patient record into the database was a task for staff, who are profile receivers. This key process works as a connector between other processes where security services were performed to guarantee confidentiality, integrity, access control and non-repudiation.

Process-2 is for registering patients or authorized staff to operate the instrument.

Process-3 controls the sampling frequencies and the number of samples in each channel. In addition, the calibration values obtained from process-4 are used to calculate values from hand exertion.

Process-4 helps to calibrate the instruments used for angle and force measurements. The respective calibration values were then stored in data storage Goniometer and Force_Instrument.

Process-5 performs support tasks such as searching previous patient test files, which were saved in the computer in an image format and controlling the

alterations made in the database, making it possible to gather information about who changed the data.

Data storage-Patient contains register information about patients; and data storage-Staff is to record the staff involved in the system operation. Temporary_Data is used to save temporary data about the acquired signal of hand exertion. Changes in the database, from any part of the system, are automatically recorded in data storage labelled as Control.

The calibrations outcomes performed via software: The measurement of wrist posture with a goniometer does not require decimal place precision (i.e., integer number). Oscillations or noise (drift problems) were observed in the fifth decimal places from the measured values. The used electrogoniometer had an offset in X and Y-axis of 3 and 6°C, respectively, when checking the accuracy of software in acquiring and processing signals, which was considered during software development.

The test with software and instrumentation for fingertip force measurement when repeated three times showed a mean standard deviation of 0.10 volts and the maximum was 0.18 volts (Table 1). These are compatible with the expected error rate (±5%) reported in the datasheet from the manufacturer of the force sensor.

For the calibration task, calibration table became empty automatically, because all previous points must be recalibrated. The history about who did what was recorded in the data storage for access control. A staff with system administrator profile had access to all information, especially to the access control, defined as Control in Fig. 7.

A sample of outcomes provided by software and electronic instrumentation together: The model of exhibition of integrated outcomes obtained from electronic instrumentation is exhibited in Fig. 8.

Figure 8 shows the outcomes: patient/subject code, patient/subject name; arbiter name; typing speed; typing frequency; flexion/extension of the wrist; lunar/radial deviation; fingertip force; tolerable values to be reached; and test date. This information helps the health professional to diagnose the computer user who is inclined to acquire the Work Related Upper Extremity Disorder (WRUED) due to inadequate hand exertion.

Improvements were done to this software in response to Machado and Villaverde, (2011), which did not successfully measure radial and lunar deviation.

In the upper left corner of Fig. 8, a sample shows how the saved outcomes were recovered. The saved file name was composed of subject code plus date and time of test, separated by special characters. The tolerable values shown in Fig. 8 were set by Machado and Villaverde (2011). The subject and arbiter analyse the outcomes regarding these tolerable values in order to make decisions for extreme hand exertion to avoid the risk of inciting muscle-skeletal illnesses.

Only one button needs to be pressed to save the test results, with the file name and location automatically chosen by the developed software.

DISCUSSION

Applied software engineering methodology: Any SE methodology approach could be an advance. The developers would choose the SE methodology and one appropriated tool case. Consequently, comprehension about what/how the software does provides information useful for software maintenance and validation method. Applying software engineering methodology, the requirements were easily traced after the software implementation. Moreover, perceived errors during the software test were quickly located in the respective source code and resolved. According to Constantine and Lockwood 2012, one advantage of narrative use case is that a series of numbered step is immediately apparent, because the separation into distinct steps makes it easier to skim the use case for an overview and the general nature of the interaction."

Functional similarity was not found in other software. Nakanish *et al*. (2009) presented a DFD, but their ERD diagram was not found, nor was a sample of their software requirement shown. Therefore, it was impossible to apply the same methodology of SE as they did. Nevertheless, their explanation about the DFD designed was fully comprehendible. Amato *et al*. (2009) seem to have used a systematic procedure. Respective documentation probably was done; however, their SE design was not included in their manuscript. On the other hand, the present study disclosed our software engineering design to encourage research of electronic instrumentation to make software documentation. No specific tool case is mandatory. There are several tool cases, some is totally free while others are temporary free provided as a demo-version. There are also the open source tools devoted to the designing and development of software that use open source platform such as Linux.

There are several commercial software to acquire and process signal. However, they do not allow changes to the source code in order to obtain personal software. Therefore, the requirements presented in the case use, would become limited to the scope of the software. This was one more reason to decide to develop the present software. The software for the signal acquisition was modified and embedded into this present system to be part of them. This software solution generated low production cost and anything can be changed at any time.

Contrary to this, Pereira *et al*. (2009) developed instrumentation and built three software programs using LabVIEW® environment. According to them, this tool provided agility to building the system but was certainly more expensive to reproduce. No documentation about software design was reported. However, they intend to develop their software in the conventional manner for lower production costs. However, if the same researcher is not available anymore, due to the lack of software documentation, more time would be demanded during software rebuilding. Hence, this present study can hopefully make the researchers realize the advantages in applying some SE methodology.

There are modern methodology for software development that could be used to develop biomedical applications (Object Oriented-OO, Artificial Inteligency-AI, Agile methods, SCRUM). However, when a methodology is new, the probability to find a partner or professional to work on the team could be reduced. In addition, depending on the application, one methodology fits better than another.

The OO approach is used less than structured analysis among veteran software developers, while AI is an approach specific for application that supports decision-making. The Agile methodology emphasizes communication among developers and users, rapid changing of information and adaptability to the change. The Scrum is an interactive and incremental agile software development method. The Aegis methods are for rapid software development, after that, the details of requirements are implemented gradually based on customer opinion, making it difficult to predict anything before the end. According to Hajjdiab *et al*. (2012), the lack of knowledge transfer between team members contributed for unsuccessful adoption of Scrum method in the United Arab Emirates. This may be due to the Scrum being a rarely used method. Conversely, for about three decades, structured analysis has been used; consequently, the possibility of find experts is larger. Structured analysis is based on process and relational database, also a consistent requirements definition is expected when changing of the requirements is not expected during implementation.

According to the chosen SE methodology, the specific programming language and tool case was chosen. This present work was based on structured analysis methodology. The Enterprise Architect tool case was chosen to design this present software because its free download was possible. However, for this computerized electronic instrumentation to be in marketing, the purchasing of commercial version of used tool case is needed.

When software is for health care, it was important to use a solid and safe database, as an example Oracle, Microsoft SQL, Firebird®, My SQL, PostgreSQL. Some of DBMS above are free while others must be purchased. Some specific training is necessary to work with any of these technologies. We used the Firebird®, because it was totally free in this version.

The use of a solid and safe database does not warrant data protect. Some specific training is necessary to work with any of these technologies. Few works have been done about protecting data for confidentiality, integrity and access control to sensitive information as those discussed by Rezk *et al.* (2012). In their purpose, when user submits a query to execute, the system checks if he has authority to access data items in the query, the context in which the user accesses the data items is checked to determine if he follows his normal behavior or not. In our case, the access control protects data by verifying the staff authentication profile in order to enable or disable resources. Also, this software controls the alterations made in the database, making it possible to gather information about who changed the data, indicating the user ID, data and time and the name of changed table.

Calibration outcomes performed via software: The sensor response capability was a linear curve; however, its predicted error rate could propagate among the acquired and processed signal, which could contribute to the standard deviations in the Table 1. The three trials present a mean standard deviation of 0.07 grams, taking into consideration the error rate of±5% in full scale predicted by the sensor manufacturer. Therefore, considering the full scale during the calibration section represented by 430 grams, producing mean tension of 4.92 volts (Table 1), the error rate of 5% was calculated. The outcomes showed that a standard deviation of 0.24 volts could be expected during calibration section. Consequentially, the reliability of software in providing accurate results for fingertip force measurement can be taken into consideration, because the maximum standard deviation observed among the three trials during calibration section was 0.14 volts.

The electrogoniometer working with developed software was previously tested and reported in Machado and Villaverde (2011). The result had an average deviation of 1.24 degrees between the electrogoniometer and the manual goniometer. This was considered satisfactory if compared with the nonlinearity of ±1% of its out signal predicted by sensor manufacturer.

Integrated use of software and electronic instrumentation: The general mechanism of error detection is to intercept outputs produced by a system (or a component) and to check whether those outputs conform to the specification (Troubitsyna, 2010). Manual calculations were used to validate the results relating to fingertip force, wrist flexion and extension, ulnar and radial deviation and typing frequency obtained by the software. Similar validation had already been efficiency applied by Kauffmann *et al*. (2011). It is extremely difficult to validate software that does not have written requirements.

Because the software and electronic instrumentation work together, a problem in one can make both become unreliable. We experienced this when the unsuccessful measurement of radial and ulnar deviations were attributed to the accelerometer technology. Therefore, another accelerometer was provided, but the problem remained, because this problem, attributed to an equipment failure, was actually due to a software implementation problem. A consultation with an expert in measuring with an accelerometer concluded that implementation code for wrist flexion and extension measurement is easier than radial and ulnar deviation, which requires signal composition during code/software implementation to reach accurate result. His suggestion was applied. This occurrence showed that human error during requirements specification or during design process can cause wrong results. This kind of error is not so easy to detect during the testing phase, because everything works normally and no error appears.

A challenge during developing of biomedical software is the requirements. For any software developer, the problem domain is very complex. The performance of our software working with electronic instrumentation reached the expectation of the physiotherapist, who used the CEI as a volunteer and arbiter. The use of this system by a physiotherapist was an important phase of the test. Upon her suggestion, changes in software occurred to use the correct medical vocabulary on the interface screen. A friendly software interface encourages the users to carry out instrumentation manipulation.

Further implementation would adopt specific database for image to prevent forced access by an unauthorized user to the test result provided in the Fig. 8 and stored as image. Another future improvement is to have the tolerable values presented as lines in the graphical output from Fig. 8, allowing the subject to train to within those limits and adjust his/her movements immediately."

This software guaranteed that the data generated and medical diagnoses remain private.

According to Shanmugapriya and Suresh (1012), a software system's reliability is defined as the probability of the software operating without failure for a specified period of time in a specified environment. One example is the system reported by Garcia-Saez *et al.* (2009) that supports remote control strategies for insulin-dependent patients. In this case, unreliable system would let to incorrect prescription that could be ultimately lethal. Because this present software was devoted to an application that does not fit into a critical mission, in that its failure will not result in loss of human life, it was not necessary to estimate the order of magnitude for probabilities of failure per hour.

Computerized electronic instrumentation should be certified in order to put it on the market. If an organization isn't worried about safety, it must consider the consequences of using mission-critical software that isn't certified or qualified as fit for purposes (Maibaum and Wassyng, 2008). For an electronic instrumentation to be marketed in Brazil, it must be submitted to the Brazilian National Health Surveillance Agency (*Agência Nacional de Vigilancia Sanitária*-ANVISA) to obtain registration/authorization. This is because some instrumentation could cause damage or harm to the patient.

In Brazil, the certification of biomedical electronic instrumentation can be obtained from a certifier company and INMETRO (*Instituto Nacional de Metrologia*). INMETRO certifies the accuracy and capability of the electronic instrumentation to accomplish its purpose. In addition, INMETRO evaluates if the risk of accident exists.

Normally, Certifier Companies follow some ISO standard, which are not easy to understand and accomplish to make the system ready to be submitted for a certification process.

Our software does not fulfil ISO standard, due to the difficulty in both adequate software and hardware. Mishra and Mishra (2009) reported a simplified software inspection process in compliance with international standards for software quality assurance. However, this is not enough to reach software certification.

The certification of medical software is a difficult due to limited experience of the software developer or the difficulties in accessing and applying the standards.

This is one challenge to be reached by the authors from this present study.

This would be useful if academic researchers would publish their experience while submitting their CEI to a certification process. This present study describes how the software engineering methodology was applied to design and develop software to acquire and process sign from a computerized electronic instrumentation. In both case, the authors intend to offer their experience as a guideline to another researchers.

The way that this study was reported could serve as a guideline for academic researchers who are not experts in software engineering methodology but usually develop their own software to run with their prototype of electronic instrumentation. However, this study cannot provide an example of how to obtain a software certification.

CONCLUSION

Development of biomedical software to work with electronic instrumentation needs to pay attention to both performance and system accuracy.

This software was developed to acquire and process signals from a specific instrumentation for measuring hand exertion to help the computer user detect their overexertion. The aim was to demonstrate how to apply the structured analysis in biomedical software to acquire and process signals when the researcher is not expert in SE methodologies. Usually, when the developer leaves the team, no one is able to maintain the software due to a lack of documentation. One capacity of SE is to produce software that is maintainable and reusable due to its clear documentation.

Due to the lack of guidelines on how to obtain certification of biomedical software that works as part of computerized electronic instrumentation, this present system was not submitted to a certification process. Despite this, the systematic documentation provided while the SE methodology was applied was an advance.

ACKNOWLEDGEMENT

This study was supported by Wagner Lima dos Santos (computer science) from WLSantos and Cia Ltda. Michelle Fernanda de Lima (Physiotherapist); Dr. Ricardo Toshiyutki Irita (Electronic engineer) from the Instituto Nacional de Pesquisas Espaciais-Brazil; and Alene Alder-Rangel from Univap (English support).

REFERENCES

Amato, F., M. Cannataro, C. Cosentino, A. Garozzo and N. Lombardo *et al*., 2009. Early detection of voice diseases via a web-based system, Biomed. Signal Process Control, 4: 206-211. DOI: 10.1016/j.bspc.2009.01.005

Arpaia, P., C. Manna, G. Montenero and G. D'Addio, 2012. In-Time Prognosis based on swarm intelligence for home-care monitoring: A case study on pulmonary disease. IEEE Sensors J., 12: 692-698. DOI: 10.1109/JSEN.2011.2158305

Constantine, L.L. and L.A.D. Lockwood, 2012. Structure and style in use cases for user interface design.

Garcia-Saez, G., M.E. Hernando, I. Martínez-Sarriegui, M. Rigla and V. Torralba et al., 2009. Architecture of a wireless personal assistant for telemedical diabetes care. Int. J. Med. Inform., 78: 391-403. DOI: 10.1016/j.ijmedinf.2008.12.003

Hajjdiab, H., S. A1-Taleb and A. Jauhar, 2012. An industrial case study for scrum adoption. J. Software, 7: 237-242. DOI: 10.4304/jsw.7.1.237-242

Kauffmann, C., A. Tang, A. Dugas, É. Therasse and V. Olivab *et al*., 2011. Clinical validation of a software for quantitative follow-up of abdominal aortic aneurysm maximal diameter and growth by CT angiography. Eur. J. Radiol., 77: 502-508. DOI: 10.1016/j.ejrad.2009.07.027

Machado, G.A.L. and A.J.B. Villaverde, 2011. Design of an electronic instrumentation for measuring repetitive hand movements during computer use to help prevent work related upper extremity disorder. Int. J. Ergon., 41: 1-9. DOI : 10.1016/j.ergon.2010.11.003

Maibaum, T. and A. Wassyng, 2008. A product-focused approach to software certification. IEEE Software Technol., 41: 91-93. DOI: 10.1109/MC.2008.37

Mishra, D. and A. Mishra, 2009. Simplified software inspection process in compliance with international standards. Comput. Stand. Inter., 31: 763-771. DOI: 10.1016/j.csi.2008.09.018

Nakanish, T., Y. Tsuchiya, T. Sakamoto and A. Fukuda, 2009. Structured analysis for software product lines. Proceedings of the 13th International Symposium on Consumer Electronics, May, 25-28, IEEE Xplore Press, Kyoto, pp: 915-919. DOI: 10.1109/ISCE.2009.5157027

Pereira, E., E.F. Manffra, J.A.P. Setti, C.M.R. Dutra and L.R. Aguiar, 2009. Development of instrumentation for application and assessment of locomotor training with partial body weight support. Brazilian J. Biomed. Eng., 25: 185-197.

Rezk, A., H.A Ali and S.I Barakat, 2012. Database security protection based on a new mechanism. Int. J. Comput. Appli., 49: 31-38. DOI: 10.5120/7879-1188

Shanmugapriya, P. and R.M. Suresh, 2012. Software architecture evaluation methods-A survey. Int. J. Comput. Appli., 49: 19-26.

Troubitsyna, E., 2010. Developing fault tolerant distributed systems by refinement. Proceedings of the 5th International Conference on Software Engineering Advances Software Engineering Advances ICSEA, Aug. 22-27, IEEE Xplore Press, Nice, pp: 178-183. DOI: 10.1109/ICSEA.2010.34