Development of software monitoring module for multi-angle electric impedance tomography method research

Grayr K. Aleksanyan*, *Ivan D.* Shcherbakov, *Artem I.* Kucher, *Andrey V.* Sulyz

Department of Information and Measuring Systems and Technologies, SRSPU (NPI), 346428 Novocherkassk, Russia

Abstract. This paper describes the experience of software development for multi-angle electrical impedance tomography. The relevance and necessity of an architecture for this kind of software product creating is considered, requirements for reliability and fault tolerance are defined. The choice of the architectural model has been made, design patterns have also been chosen taking into account the prospective scenario of using the developed software, their advantages are shown too. The most important attributes of software quality are described, which should implement the developed software product. The considered attributes of quality are projected on specific modules of the developed software. The architectural models and templates chosen during design are described, their advantages and disadvantages are considered. Support issues for the software developed by the cross-platform are discussed, and the possibility of working in various operating systems is demonstrated.

1 Introduction

Within the framework of multi-angle electric impedance tomography (MEIT) problems, software (software) has been developed to interact with the EIT device. The developed software is a desktop application running on Windows and Linux platforms, suitable for use in 2D and 3D electrical impedance tomography[1-3].

Two-dimensional electrical impedance tomography, in the context of software, should realize the possibility of receiving one-belt measurement data from the EIT device and visualizing the data, should be presented as a 2D image[4].

Three-dimensional electrical impedance tomography, in the context of software, should realize the possibility of receiving measurement data of an indeterminate number of belts from the EIT device and visualization of the data, should be represented as a (3D) image in several variations[5].

A desktop application is a client software. The application is installed on the user's workstation and runs locally, or it is launched remotely. The EIT software assumes the functioning of a system that continuously receives data from the EIT device, analyzes the

* Corresponding author: graer@yandex.ru

data received and displays the result in the application window in 2D or 3D format. The diagram of the UML deployment is shown in Fig. 1.

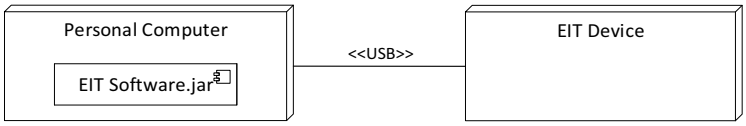

Fig.1. The UML deployment diagram.

At the planning stage, the quality attributes that are required to be implemented in the EIT software are highlighted [6]:

- scalability - the degree of simplicity of effective and rational change of the product or system without adding defects and reducing the quality of the product;
- flexibility - the extent to which a product or system can be used with efficiency, efficiency, freedom from risk and in accordance with requirements in circumstances that go beyond what was originally specified in the requirements;
- portability, mobility - the degree of simplicity of efficient and rational transfer of the system, product or component from one environment (hardware, software, operating conditions or conditions of use) to another;
- fault tolerance - the ability of the system, product or component to operate as intended, despite the presence of software or hardware defects.

2 Choosing a software architecture

Based on the requirements, the software architecture [6-8] was developed, a single-level (monolithic) architectural model was chosen [9]. A widespread problem of this model is low scalability. [10]

To split the application into components, the Model-View-Controller (MVC) design pattern is chosen, which provides separation of business rules, user interface and control logic [11].

The MVC separates the view from the model, establishing a "subscription / notification" protocol between them. The view must ensure that the external view reflects the state of the model. With each change of internal data, the model notifies all its dependent types, as a result of which the view updates itself. This approach allows you to attach several types to one model, thus providing different views. You can create a new view without overwriting the model. The scheme of the MVC template is shown in Figure 2.

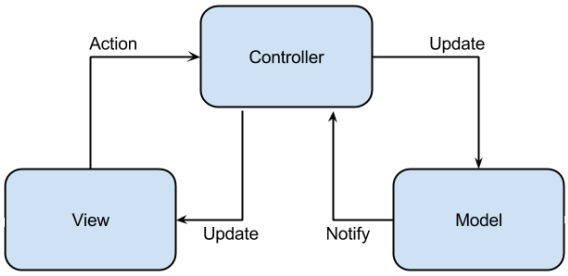

Fig. 2. The scheme of the MVC design pattern.

Based on the original solutions, the original version of the system architecture is constructed in the form of a packet diagram. Figure 3 shows the basic structure of software packages.

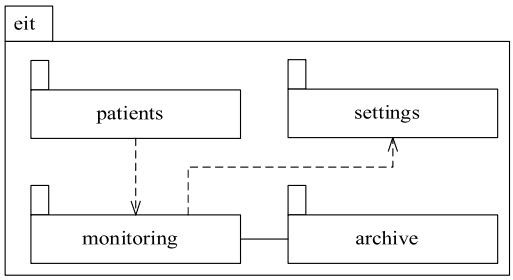

Fig.3. The basic structure of software packages.

The packages presented in Figure 3 contain software windows developed using the MVC template.

- monitoring package contains a window in which the main logic of the software operation takes place, namely the receipt, processing and analysis of measurement data in real time;
- archive package contains a window for processing the previously acquired measurement data.
- patients package contains a list of patients' medical records, with the ability to create and edit the selected card.
- settings package contains the settings window.

In this paper, we only consider the implementation of the monitoring package, whose packet diagram is shown in Figure 4.

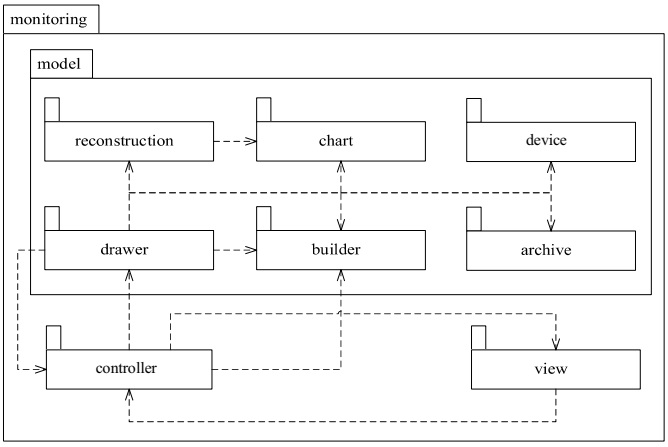

Fig. 4. Initial structure of monitoring component.

The reconstruction package contains a set of classes that implement algorithms for reconstructing the conductivity field of the human chest cavity. The device package contains a set of classes providing interaction with the EIT device: opening the COM port, initializing the device, acquiring the measurement data and closing the COM port, the chart package contains a set of classes for working with graphs reflecting the lung ventilation function with respect to the received data from the EIT device. The builder package in turn contains a set of classes for reading the Json [12] array, which contains the coordinates of the nodes of the finite element grid, as well as creating an array of structures to build it. The archive package contains a set of utilities for compressing the received measurement data, writing them to the Json file and saving the file in a specific directory. The drawer package contains a class that represents the implementation of the "Facade" design pattern. The

template provides a unified interface instead of a set of interfaces for some subsystem. The facade defines a higher-level interface that simplifies the use of the subsystem [13]. Thus, working with the modules of the monitoring_window package and returning the data to the controllers is enclosed in this package. The controller package contains a set of classes that interprets user actions, notifying the model of the need for changes. The view package contains a set of classes that is responsible for displaying the model data to the user, responding to model changes.

2.1 Scalability and flexibility

To solve the problem of complex scalability of modules, it was required to reduce the number of dependencies between packets and to arrange such that the specific implementation of these packages did not interfere with the rest of the system. Since under the existing implementation the change in some part of the module caused a change in the module depending on it, the cost of each modification of the application increased.

The solution to this problem was the delegation of the creation of objects of specific classes in a package containing a class that implements the "Factory method". Also, reconstruction, device and builder packages are included in a separate package of services.

The design pattern "Factory Method" defines the interface for creating the object, but leaves the subclasses deciding which class to instantiate. The factory method allows the class to delegate instantiation to subclasses [6].

The corrected diagram of EIT software packages is shown in Figure 5.

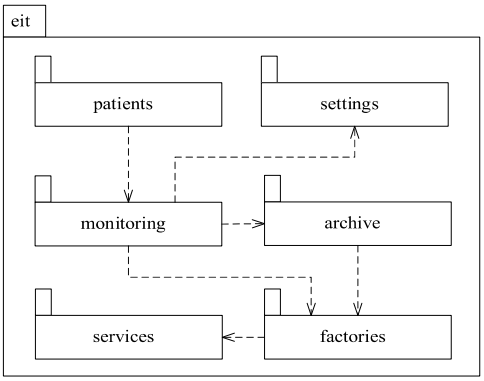

Fig.5. Corrected diagram of software packages EIT.

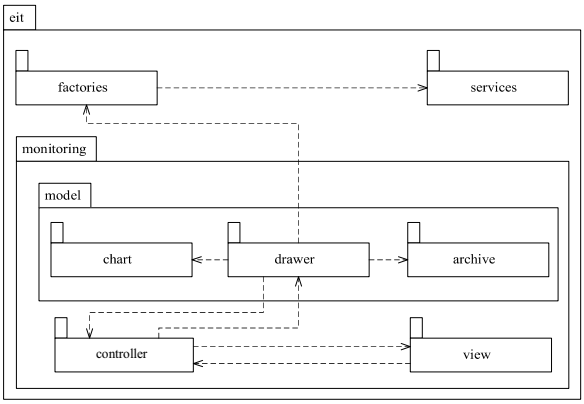

Fig.6. Corrected diagram of the monitoring package.

In the services package, there are also common interfaces for packages in the services package, so that subclasses can produce objects of different classes that follow the same interface. This introduction has led to the fact that modules that are subject to frequent modifications will not affect the operation of the rest of the system. When you make any changes to these packages, you only need to change the classes that are in the factories package.

These solutions allow to increase the flexibility of these modules and make them as independent as possible from other parts of the system. Thus, the problem of system scalability is reduced, which is caused by the use of a single-level architectural model.

2.2 Portability

The current version of the software is implemented in the Java [10] language and is a cross-platform application. However, some of the application modules are currently tied to a specific operating system (OS). The drawer package defines the COM port number of the device when it is initialized by idVendor and idProduct.

To do this, a package os has been created in the package eit.monitoring.model, the structure of which is shown in the figure. The idVendor and idProduct numbers are stored in an external XML file.

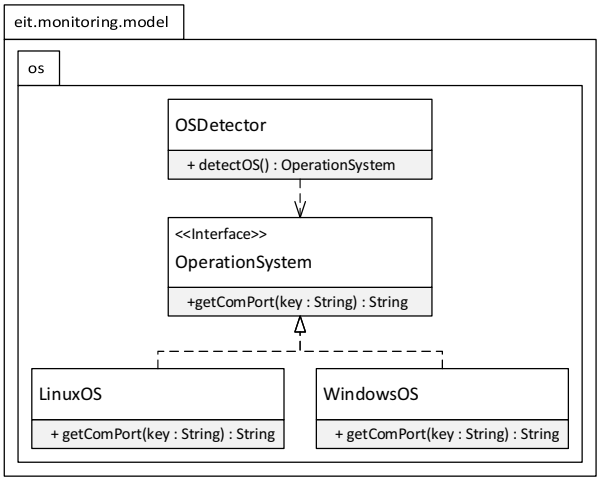

Fig.7. UML class diagram for the os package.

If the software is running on Windows, then the COM port of the device is determined by the OS registry, if the device is running on a Unix / Linux operating system, the connected devices are searched via USB. The search algorithm is shown in Figure 8.

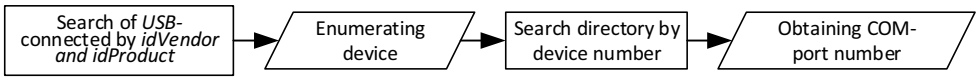

Fig.8. Block scheme for finding the COM port number of the device in the Linux OS.

2.3. Fault tolerance

To work with the device was chosen library jSSC [13], which makes convenient interaction with the microcontroller on the COM-port. This library assumes a number of solutions for

reading data from the controller: installing the listener on the COM port or standard reading of the number of bytes from the buffer.

When working with the EIT device, errors occur that return values different from the usual measurement data. To ensure the testability of this component and create a user-friendly interface, we used the standard wait and notify lockout for the Java programming language [14]. In Fig. 9 shows the sequence diagram of the module.

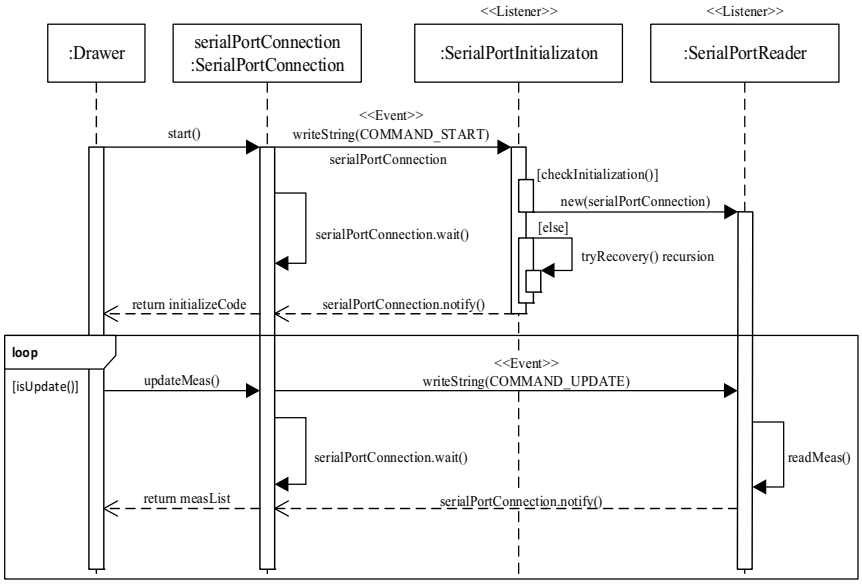

Fig. 9. The UML sequence diagram of the «device» component.

Figure 9 shows that when the device is successfully initialized, a new object of the SerialPortRead class is created, which receives the measurement data using the readMeas () method, while the isUpdate () condition is satisfied. In the event of a device initialization error, the SerialPortInitialization object returns a specific error code. It is also clear from the figure that all work with the initialization of the device and the reception of measurement data is achieved by using the events of the class called in the object SerialPortConnection.

The problem of using events is that their execution occurs in a separate thread and, in order to create a convenient interface for working with the device module, it was required to block the execution of the main stream of the object of the SerialPortConnection class for the time of initialization and reception of the measurement data. Thus, this implementation allows the client-class Drawer to use a simple interface when working with the EIT device without disturbing its execution flow.

Conclusion

The paper describes the experience of software development for the tasks of multi-angle electric impedance tomography. The choice of architecture for the software product was made and justified, design patterns were chosen.

The main attributes of the quality of the developed software are considered and selected; selected architecture, design patterns and means for implementing the main quality characteristics of the developed software are tested on a specific software module for multi-angle electrical impedance tomography, showing the advantages and

disadvantages of the developed product. The chosen solutions allowed creating a convenient interface for working with the MEIT device, creating tools for effective user interaction with the software, for example, switching the number of belts, displaying the reconstructed data (2D, 3D). Automatic recovery of the device from the fault condition allows to realize the fault tolerance of the system when working with the EIT device.

Work is performed within the grant of President of Russian Federation for state support of young Russian scientists MK-196.2017.8 "Development of theoretical foundations and algorithms for multi-view systems are three-dimensional electrical impedance tomography for non-invasive medical imaging".

References

1. G. K. Aleksanyan, I. D. Shcherbakov and A. I. Kucher. J. of Eng. and Appl. Sc., **12**(3), 587 (2017)
2. G. K. Aleksanyan, A. I. Kucher , A. D. Tarasov , N. M. Cuong, C. N. Phong. Int. J. of Soft Comp., **10**(6), 462 (2015)
3. G. K. Aleksanyan, N. I. Gorbatenko, A. I. Kucher, K. M. Shirokov, C. N. Phong, Biosc. Biotech. Res. Asia., **12**, 709 (2015)
4. G. K. Aleksanyan, N. I. Gorbatenko, V. V. Grechikhin, T. N. Phong, T. D. Lam, ARPN J. of Eng. and Appl. Sc., **11**(9), 5871 (2016)
5. G.K. Aleksanyan, N.I. Gorbatenko, A.D. Tarasov, Res. J. of Appl.Sc, **9** (12), 1030 (2014)
6. F.V. Stankevich, XIX Int. Sc. and Pract. Conf. "MOD. TECH. AND TECHN.", **19**, 366 (2013)
7. ANSI/IEEE Std 1471-2000 https://standards.ieee.org/standard/1471-2000.html
8. M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture* (Addison-Wesley, 2002)
9. Info: https://docs.microsoft.com/en-us/previous-versions/office/developer/server-technologies/aa480455(v=msdn.10)
10. I. A. Lenhorova, New res. in the dev. of tech. and techn., **2**, 48 (2015)
11. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Priyomy ob"yektno-oriyentirovannogo proyektirovaniya. Patterny proyektirovaniya* (Piter, Saint-Petersburg, 1994)
12. Info: http://json.org
13. R. C. Martin, *Chistaya arkhitektura. Iskusstvo razrabotki programmnogo obespecheniya* (Piter, Saint-Petersburg, 2018)
14. R. Martin, M. Martin, *Printsipy, patterny i metodiki gibkoy razrabotki na yazyke C#* (Symvol-Plus, Moscow, 2011)