**35th Annual International Conference of the IEEE EMBS**
**Osaka, Japan, 3 - 7 July, 2013**

# Open Architecture Software Platform for Biomedical Signal Analysis

Juliano J. Duque[1], Luiz E. V. Silva[2] and Luiz O. Murta Junior[3]

*Abstract*— **Biomedical signals are very important reporters of the physiological status in human body. Therefore, great attention is devoted to the study of analysis methods that help extracting the greatest amount of relevant information from these signals. There are several free of charge softwares which can process biomedical data, but they are usually closed architecture, not allowing addition of new functionalities by users. This paper presents a proposal for free open architecture software platform for biomedical signal analysis, named JBioS. Implemented in Java, the platform offers some basic functionalities to load and display signals, and allows the integration of new software components through plugins. JBioS facilitates validation of new analysis methods and provides an environment for multi-methods analysis. Plugins can be developed for preprocessing, analyzing and simulating signals. Some applications have been done using this platform, suggesting that, with these features, JBioS presents itself as a software with potential applications in both research and clinical area.**

## I. INTRODUCTION

During the last decades, different areas of medicine have seen an increased use of biomedical signal processing to support the clinical evaluation of patients. The use of novel and sophisticated methods to analyze biomedical signals may add valuable information in the traditional assessment of the patient. Moreover, the application of advanced computational tools may help speeding up a diagnosis.

In the clinical practice, patients are subject to several tests, which may be invasive or noninvasive and generally result in biomedical signals. These signals can be of several kinds of source, such as electrical and magnetic activities, pressure, intervals between events of interest, etc. Examples of electrical signals are the electrocardiogram (ECG), electroencephalogram (EEG) and electromyogram (EMG). For magnetic sources, magnetocardiograms (MCG) and magnetogastrograms (MGG) are two examples that can be mentioned. Arterial blood pressure (ABP) and systolic blood pressure (SBP) are examples of blood pressure signals. In addition, it is also possible to generate biomedical signals from intervals between events, such as heart beats, eye blinks and strides.

Biomedical signals represent activities of the many organs which form the human body, and enable the identification of physiological and pathological conditions [1]. They are generally evaluated by visual inspection by the clinicians, or analyzed with dedicated software packages. However, the use of more advanced signal processing tools may provide a more accurate and rapid diagnosis, and a more suitable and timely treatment for the patient.

In this context, we propose an open architecture software platform for analysis of biomedical signals. This platform, named JBioS and based on Java technology, allows an easy way for implementation and integration of new software components, named plugins. Analysis methods, preprocessing tools and signal generators can be implemented as plugins. We expect JBioS contributes both to research and clinical practice, supporting novel methods validation as well as full clinical analysis of biomedical signals.

## II. SOFTWARE STRUCTURE

There are several free of charge computer softwares in the world, which can be used for biomedical data processing. For example, Kubios HRV [2] and CardioSeries [3] are very useful computer software for heart rate variability (HRV) analysis. The Physionet [4] portal hosts several biomedical signals, besides computational tools for handling and processing them. BioSig [5] is another software library that can process different types of biomedical signals. However, softwares like those usually have specific purpose and new features and functionalities can not be aggregated to them with ease by users.

In image processing area, ImageJ [6] is one of most famous and powerful softwares freely available. One of the most interesting features of ImageJ is its plugin functionality. It allows users to write their own code and to plug it in ImageJ, benefiting from the whole structure for opening and manipulating images. It makes programmers tasks easier and faster, encouraging researchers to advance on their fields. Furthermore, plugins that were created throughout the world, for many different purposes, can be shared with all the community, as they are on the ImageJ website.

Following the ImageJ plugin idea, we developed a software platform with this same plugin feature, but focusing on one dimensional biomedical signals instead of images. We named it JBioS (Biomedical Signal Analysis for Java), and its main purpose relies on the signal flow illustrated in Fig. 1. JBioS offers support for signal loading, preprocessing, analysis performing and results displaying. Developed in Java programming language, JBioS assures portability in different operating systems.

There are, basically, four types of data that JBioS can load: ASCII text files, SCP-ECG standardized files [7], [8], Physionet data and simulated signals. In JBioS, signals are seen as objects composed by an array of samples and the

This work was supported by São Paulo Research Foundation (FAPESP), under grant number 2013/00922-5.

The authors are with Department of Computing and Mathematics, Faculty of Philosophy, Sciences and Literature of Ribeirão Preto, University of São Paulo, Ribeirão Preto, SP, Brazil.
[1] `julianojd at gmail.com`
[2] `luizeduardovs at gmail.com`
[3] `murta at usp.br`

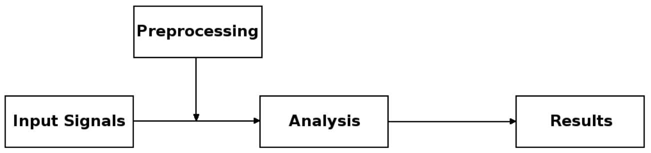


Fig. 1. JBioS basic idea for biomedical signal analysis flow. Firstly, signals are loaded into the system. Some tools can be applied to those signals, creating preprocessed versions of them. After that, users should select which signals will be passed to the analysis stage. Then, desired analysis methods are selected and, after running them, results are shown in charts or/and texts.

following attributes: a label, a type (for example, ECG), a sampling rate, units for X and Y axis and an initial X axis value.

Text files, which may have various columns, are considered as one signal, where each column represents one different acquired channel. These files also might have a header, which facilitates setting signal information. If no header is found, users might set signals information while loading. SCP-ECG communication protocol files are automatically imported by JBioS. This SCP-ECG module was supported by the PixelMed Java DICOM Toolkit [9]. All the information contained in SCP-ECG files can be viewed before importing, but only the information related to the attributes defined in signal objects will be imported to JBioS. Physionet data can be directly obtained from its database through a internet connection. For this function to work properly, a native dynamic library which utilizes some Physionet tools was developed. Finally, simulated signals makes use of JBioS plugin functionality, allowing users to create any type of signal they need to run analysis methods and do comparison with real signals.

JBioS loaded signals can also be saved as text files for further using. Signal files generated by JBioS have a header, containing its information, and columns representing each signal channel.

At preprocessing stage, users are able to apply some processing tools on loaded signals, intending to prepare them for the analysis stage. These tools generate new signals, and the old ones are still kept in the system. Filtering, shifting, cutting, peak detection on ECGs, etc., are some examples of preprocessing tools. Those tools are also plugins in JBioS platform, allowing users to create their own ones.

After loading and preprocessing signals, they can be passed to analysis stage. As analysis methods are plugins, they must be programmed by users. Signals are passed as parameter to the method and it must return result sets, containing charts and/or single values obtained from the analysis process.

Result charts are created using JFreeChart [10], a powerful and free Java library package. By default, JFreeChart enables users to customize some chart properties out of coding, such as background color and axis labels and fonts. JBioS plugin developers do not need to work directly with JFreeChart classes. Instead, they should use classes provided by JBioS, which provide support to make JFreeChart charts and requires minimal knowledge, still allowing some minor chart customization by coding.

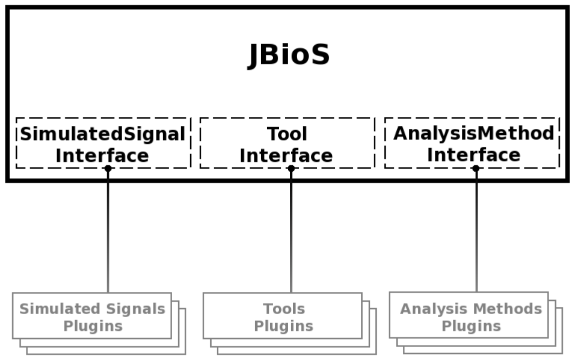


Fig. 2. Illustrative schema of JBioS and its connections with plugins. A plugin, according to its purpose, must implement one of the three different types of interface available.

## A. Plugins

As previously mentioned, plugins can be created for three basic purposes in JBioS: signal simulation, preprocessing and analysis. For each case, developers must write a Java class which implements a specif Java interface. Interfaces in Java are similar to classes, but must contain only methods signatures. It requires classes which implement interfaces to provide the methods implementations. In order for JBioS to recognize and properly load the plugins, their class files must be put in specific root folders, according to their purpose.

For signal simulation, a class must implements *SimulatedSignal* interface, which has three method signatures: *getName*, *inputParameters* and *getSignal*. The first method must return a *String* object, containing the signal name that appears for users in JBioS. The second one does not return anything and it is used to take all setting parameters, which can be asked to users through another JBioS class, namely *InputParametersDialog*. The last method is used to generate and return a *Signal* object. In addition to *SimulatedSignal* interface, JBioS also offers an abstract class, *AbstractSimulatedSignal*, which implements *SimulatedSignal*, and can be extended to create a plugin. This abstract class provides some facilities in case the simulated signals are described by a continuous function, such as $y = f(x)$.

For preprocessing tools, users must implement *Tool* interface, which also has three methods: *getName*, *inputParameters* and *process*. The first two methods have the same purpose as in *SimulatedSignal* interface. The last one receives an array of *Signal* objects and must be used in the preprocessing implementation, eventually returning another array of new signals, which will be included in JBioS loaded signals list.

Analysis methods must be created using *AnalysisMethod* interface, which have the following methods: *getDescription*, *inputParameters* and *doAnalysis*. The first one must return a *String* object that briefly describes the method. As the mouse is positioned over one method checkbox, a tooltip text shows this description. The second method, *inputParameters*, has the same purpose as in other interfaces. It must contain code to set necessary parameters to analysis methods. *InputParametersDialog* can be used to ask users to set the desired parameters. JBioS calls this method when users click on method checkbox. The last method, *doAnalysis*, gets an array of signals objects as parameter and must implement all the process of analysis, eventually returning results in an *AnalysisResult* object.

Fig. 2 illustrates the connections between plugins and JBioS, indicating which interfaces are responsible for each of the three plugin types. All names of plugins classes must end with an underline (_) in order for JBioS to recognize them as such.

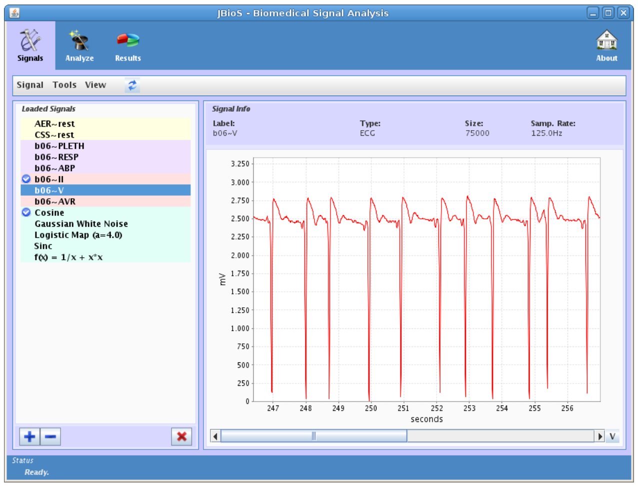
*AnalysisResults* objects might contain both values and charts, where signals groups can be plotted either separately or together. JBioS allows six types of result charts: XY, AREA, BAR, PIE, HISTOGRAM and DOTS. In XY and AREA types, results series are plotted as lines, with X and Y values. The difference is that AREA type fills the areas under curves. In BAR and PIE types, the results are represented, respectively, by bars or a pie. The HISTOGRAM type represents results in bars form, but the distribution of series is calculated automatically. Only the number of bins (levels) must be specified. Finally, DOTS type represents XY values as single points, not associating any ordering to them.

## III. APPLICATIONS

There is a bar at the top of JBioS graphical interface which has buttons that allow to select one of the JBioS panels: Signals, Analyze, Results and About.

In Fig. 3, a screenshot of Signals panel is shown. The left list is the signals list and contains all the signals loaded into the system. Each signal in the list is associated with a different background color, regarding its type. At the panel bottom, "plus" and "minus" buttons are respectively used to select and unselect signals for further analysis. Selected signals are identified by a blue circular check icon. The "x" button is used to remove signals from the list. By clicking on one signal in the list, its attributes and graphical representation are displayed on the large right panel. Just above the signals list, there is a menu bar. Signals are loaded from the Signal menu, while preprocessing tools are available from Tools menu. A multichannel viewer is also available in the menu bar, in case users need to visualize several signals at once. Moreover, there is also a refresh button, for reloading plugins without needing to restart JBioS. The root folder for simulated signals plugins is 'signals' and for preprocessing tools plugins is 'tools'.

Fig. 4 shows a screenshot of JBioS analyze panel. The table shows signals that were previously selected in the signal panel. The right panel contains all the analysis methods that were recognized and loaded by JBioS. There is a check box associated to each method, used to select them. By checking a method, a dialog box appears, where its parameters can be set. In this Fig. 4, for example, the available methods are: fourier and wavelet transforms, autoregression, correlation analysis, time domains parameters, return map, multiscale

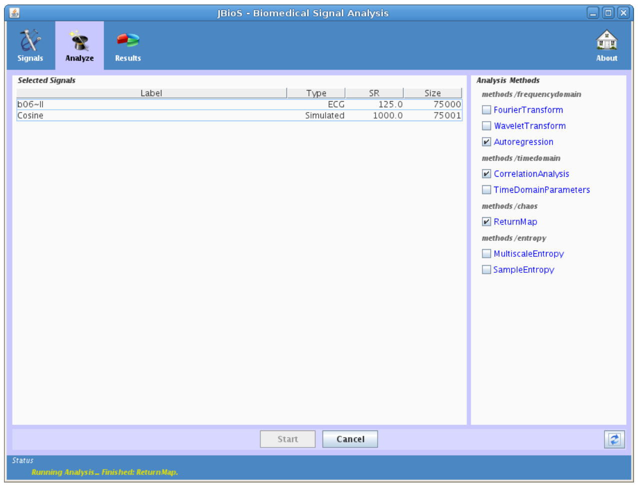


AER~rest
CSS~rest
b06~PLETH
b06~RESP
b06~ABP
● b06~II
b06~V
b06~AVR
● Cosine
Gaussian White Noise
Logistic Map (a=4.0)
Sinc
f(x) = 1/x + x*x

| Label: | Type: | Size: | Samp. Rate: |
|---|---|---|---|
| b06~V | ECG | 75000 | 125.0Hz |

Ready.

Fig. 3. JBioS Signals panel. The left list contains all the signals loaded, and the right panel shows the selected signal information.



| Label | Type | SR | Size |
|---|---|---|---|
| b06~II | ECG | 125.0 | 75000 |
| Cosine | Simulated | 1000.0 | 75001 |

methods /frequencydomain
- ☐ FourierTransform
- ☐ WaveletTransform
- ☑ Autoregression

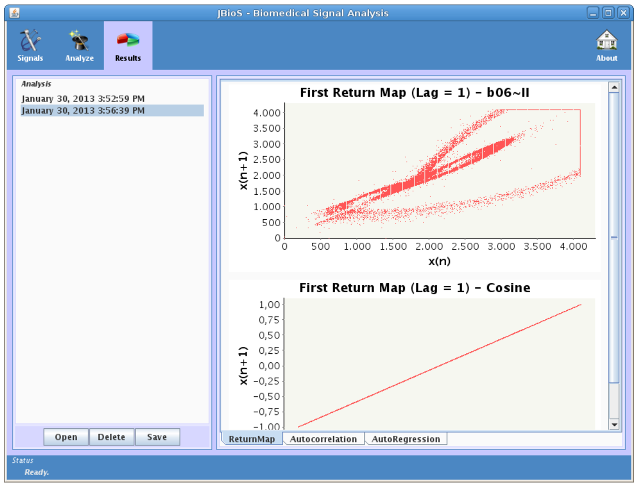
methods /timedomain
- ☑ CorrelationAnalysis
- ☐ TimeDomainParameters

methods /chaos
- ☑ ReturnMap

methods /entropy
- ☐ MultiscaleEntropy
- ☐ SampleEntropy

Running Analysis... Finished: ReturnMap.

Fig. 4. JBioS Analyze panel. Selected signals are displayed in a table. The right panel contains all the analysis methods recognized and loaded by JBioS.


JBioS - Biomedical Signal Analysis
Signals
Analyze
Results
About
Analysis
First Return Map (Lag = 1) - b06~II
x(n+1)
x(n)
First Return Map (Lag = 1) - Cosine
Open
Delete
Save
ReturnMap
Autocorrelation
AutoRegression
Status

January 30, 2013 3:52:59 PM
January 30, 2013 3:56:39 PM

Ready.

Fig. 5. JBioS Results panel. The left panel shows already performed analysis. Results of each method are shown in right panel, separated by tabbed panels.

and sample entropy. Each of these plugins were placed in different subfolders (”frequencydomain”, ”timedomain”, ”chaos”, ”entropy”), inside the root folder for analysis methods plugins, namely ”methods”. Once all the desired methods were selected, Start button should be pressed to run the analysis. During the execution, the Cancel button can be pressed to stop it.

When the analysis running has finished, users must pass to result panel, illustrated in Fig 5. Its left panel shows all analysis were already executed. Each analysis may contain different results, depending on the number of methods previously selected. Results of each method are separated by tabbed panels in the right panel. In Fig. 5, return maps of the selected signals in Fig. 3 are shown. Analysis can be saved in a file (.anls) and stored on hard disk. After, the analysis file can be reopened in JBioS for posterior evaluation.

Our group members have been applying JBioS to several studies with biomedical signals. Table I shows some examples.

TABLE I
EXAMPLES OF JBIOS APPLICATIONS

| Signal Data | Implemented Method | Results |
|---|---|---|
| Physionet heart rate variability databases | Multiscale entropy [11] | We reproduced results published in [11] |
| Physionet challenge 2010 database | Reconstruction of multichannel ECGs [12] | Signals were reconstructed using artificial neural network |
| Heart rate variability (HRV) signals obtained from university hospital | HRV time domain parameters [13] | Results are consistent with HRV task force [13] |

## IV. AVAILABILITY

JBioS is available free of charge, as well as some plugins examples and other resources. More details can be found at http://dcm.ffclrp.usp.br/csim/jbios.

## V. CONCLUSIONS

The software platform presented here is a proposal for an extensible tool with potential applications in research field and also clinical practice. Signal simulators, preprocessing and analysis methods can be easily incorporated in the platform as plugins, requiring a minimal development by users. It can promote an increasing number of plugins available for the platform, which could be shared by users through a website. Also, JBioS usefulness have been demonstrated with some applications performed by our group.

Furthermore, we intend to improve some features of JBioS, such as chart customizations as well as extend plugins for signal loading. The latter expands possibilities of data format importing. Moreover, the more plugins types JBioS can handle, the more comprehensive it will be, once plugins could be shared by community.

We also intend to make JBioS an open source project. Hence, it is expected that JBioS features can be progressively improved by the users community, which can contribute significantly in biomedical signal analysis field.

## REFERENCES

[1] R. Rangayyan, *Biomedical Signal Analysis*. IEEE Press / Wiley, New York, 2002.

[2] J.-P. Niskanen, M. P. Tarvainen, P. O. Ranta-aho, and P. A. Karjalainen, “Software for advanced hrv analysis,” *Computer Methods and Programs in Biomedicine*, vol. 76, pp. 73–81, 2004.

[3] D. Penteado, “CardioSeries,” 2012. [Online]. Available: http://sites.google.com/site/cardioseries

[4] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, “Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[5] C. Vidaurre, T. H. Sander, and A. Schlögl, “Biosig: The free and open source software library for biomedical signal processing,” *Computational Intelligence and Neuroscience*, vol. 2011, 2011.

[6] W. Rasband, “Imagej,” U. S. National Institutes of Health, Bethesda, Maryland, USA. [Online]. Available: http://imagej.nih.gov/ij/

[7] C. Zywietz, “Scp-ecg and vital signs information representation - two examples of successful transcontinental cooperation in medical informatics standardization,” *International Journal of Medical Informatics*, vol. 48, pp. 195–199, 1998.

[8] C. Zywietz and R. Fischer, “Integrated content and format checking for processing of scp ecg records,” *Computers in Cardiology*, vol. 31, pp. 37–40, 2004.

[9] D. A. Clunie, “Pixelmed java dicom toolkit,” 2011. [Online]. Available: http://www.pixelmed.com

[10] D. Gilbert, “JFreeChart,” 2005–2012. [Online]. Available: http://www.jfree.org/jfreechart/

[11] M. Costa, A. L. Goldberger, and C.-K. Peng, “Multiscale entropy analysis of complex physiologic time series,” *Phys. Rev. Lett.*, vol. 89, no. 6, p. 068102, 2002.

[12] L. E. V. Silva, J. J. Duque, M. G. Guzo, I. Soares, R. Tinós, and L. O. Murta, “Medical multivariate signal reconstruction using recurrent neural network,” in *Computing in Cardiology, 2010*. IEEE, 2010, pp. 445–447.

[13] M. Malik, J. T. Bigger, A. J. Camm, R. E. Kleiger, A. Malliani, A. J. Moss, and P. J. Schwartz, “Heart rate variability: Standards of measurement, physiological interpretation, and clinical use.” *Circulation*, vol. 93, no. 5, pp. 1043–1065, 1996.