MobileBIT
A Framework for Mobile Interaction Recording and Display

Michel Cânovas[1], Hugo Silva[1], André Lourenço[1,2] and Ana Fred[1]
[1]*Instituto de Telecomunicações, Instituto Superior Técnico, Avenida Rovisco Pais, 1, 1049-001 Lisboa, Portugal*
[2]*Instituto Superior de Engenharia de Lisboa, Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisboa, Portugal*

Keywords: Android, Smartphone, ECG, Rapid Application Development.

Abstract: The proliferation of mobile devices and smartphones, together with their built-in sensors and the ability to easily connect to other peripherals using Bluetooth, is enabling new and promising signal processing applications. However, the development of mobile applications that use these features is still in the early days, requiring advanced development skills. In our work, we propose a generic framework for rapid prototyping in a mobile environment. The proposed framework enables real-time data acquisition, processing, recording, communication and visualization. As proof of concept we present an application used for electrocardiographic (ECG) signals monitoring.

1 INTRODUCTION

Nowadays, smartphones and tablets devices play a fundamental role, enabling ubiquitous access to real-time information in an always-on/always-connected approach (Wan et al., 2009). However, application development can be a thorough and time-consuming process for researchers, which often falls out of the scope of their core interests.

Considering this, the main motivation behind our work is to create a tool that addresses real-time biomedical signal acquisition, processing, recording, communication and display, using mobile or web platforms. Our ultimate purpose is to give biomedical researchers, physicians, psychologists or other health-related professionals, an intuitive tool to easily develop mobile signal processing and communication applications tailored to their needs. These can be used in telemedicine and mobile monitoring of health, improving quality of care and independent living (Wan et al., 2009; Silva et al., 2011a; Silva et al., 2011b).

2 SYSTEM ARCHITECTURE

The architecture is arranged in layers decoupling the presentation from the processing counterpart. It uses the Model-View-Controller (MVC) architecture, as proposed by Silva *et al.* (Silva et al., 2012). Moreover, the definition of the application is made by assembling a network of functional blocks that runs on top of a Workflow Manager (WFM). This network defines the behavior of the application, in such a way that each block performs a specialized task, including data acquisition from a source, signal processing, and display of results.

The WFM is the core of the framework. It deals with the instantiation and control of a previously designed system, described according to a structure defined in JSON. This structure uses a specification, called Data Processing Language, that is automatically loaded and instantiated at runtime by the framework. Figure 1 gives an overview of the architecture.

2.1 Functional Blocks

These are the operating units of the system; it is their configuration and combination that defines the ultimate behavior of the end-user applications designed with the proposed framework. All blocks must extend an abstract class, which includes methods to implement the observer pattern (Cooper, 2000) and other convenient methods which give handle to the WFM. The communication between blocks is made through an event-based Message Passing Protocol using the Android OS `Message.class`.

Conceptually, the framework uses three types of blocks: a) the normal blocks, which take input data, perform a specific task, and output results; b) the

Cânovas M., Silva H., Lourenço A. and Fred A..
MobileBIT - A Framework for Mobile Interaction Recording and Display.
DOI: 10.5220/0004247003660369
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2013), pages 366-369
ISBN: 978-989-8565-37-2

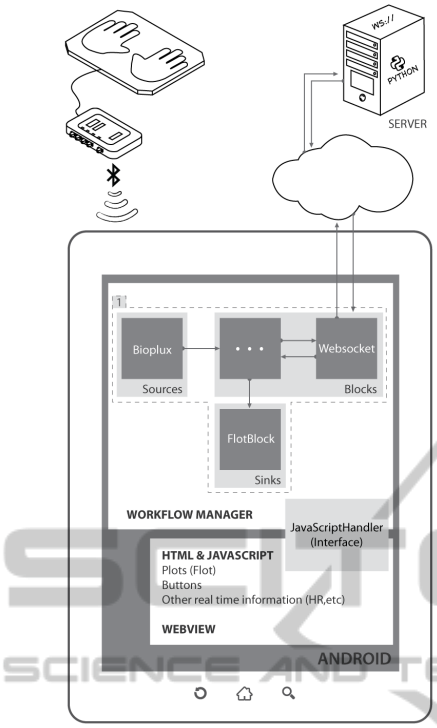

Figure 1: Architecture of our demo application, where biosignals are acquired from an external hardware device, displayed in a mobile device through a rich user interface, and sent to a remote server for efficient processing; the results of the processing stage is returned back to the phone for presentation to the user.

Sources, which are a special type of blocks that only have outputs; c) Sinks, that are a special type of block which only have inputs. Our framework has the underlying assumption that the functional blocks are triggered by the output data produced by the Source blocks, and thus, a normal system operational flow starts on the sources and generally ends on Sinks (as part 1 of Figure 1).

All the information about the type, input/output connections and configuration of the blocks, which, overall, characterize a realization of a system, is represented using a structure defined according to the Data Processing Language, which will be further detailed in the next section.

2.2 Data Processing Language

The description of a system to be instantiated by the WFM, is described using a JSON-based specification to which we called Data Processing Language (DPL). It has a simple organization as follows:

```
{"<CLASS:LABEL>":
   {"config": {"<PARAMETER>":"<VALUE>",...},
    "in": {"<CHANNEL>":
      "<CHANEL_LABEL>"|"<LABEL>/<CHANNEL>",...},
    "out": {"<CHANNEL>":"<CHANNEL_LABEL>",...}
   },
  "<CLASS:LABEL>":...
}
```

This structure can be seen as a collection of key/value pairs through which the blocks, that the end-user application will use, are identified, and their basic configuration information is provided. As long as there is a correspondence between the represented blocks (`CLASS` in `"<CLASS:LABEL>"`) and Java classes (`CLASS.class`, in Java), the WFM will be able to instantiate and build the defined system in runtime. This structure is not only easily read but can also be written or edited very simply. Future work will be focused on having this structure generated through a visual programming tool and fed to the application automatically.

2.3 Workflow Manager

The WFM is the core of the mobile framework, and it manages the execution and interaction between the different blocks. It contains a handle to every important instance of the system, and it is through the WFM instance that most of the others communicate.

The most demanding task performed by the WFM is the initial network setup from the DPL structure. As soon as the end-user application starts, a JSON structure, following the specification described in the previous section, is fed to the WFM. Then, using a the high performance JSON processing library, all blocks are instantiated sequentially, configured, and connected among themselves through their input and output channels.

Once the whole system is assembled, all that remains to be done is to trigger it, namely by starting the sources. Then, the arrival of the data itself to the input of the next block triggers its operation, and so on. Finally, the processed data will arrive to a sink which gets the flow to an end. Sinks end the flow either by saving, streaming or displaying their input.

Usually, this action of starting the sources, is triggered by an UI event. Since the UI is shown through a Webview, which, in turn displays HTML content, the native part of the application must be able to exchange events with the Javascript side. Hence, the `Webview` class provides a method called `addJavascriptInterface`, that binds a Java object to Javascript. This object turns out to be the interface between the Java (the native Android) and Javascript (in the HTML page) worlds (further explanation continues on the next section).

The WFM communicates with this interface (hereafter called `JSI`) through messages containing

commands to be executed. For this reason, the WFM has also the very important task, which is to parse and execute these command messages. Likewise, since the only way to the `JSI` is across the WFM, all data that needs to be displayed to the user has to arrive to the WFM to be correctly dispatched.

2.4 User Interface

The UI is, as in most applications, one of the most important parts of the platform. It needs to be able not only to accurately display the data to the user, but also to get his inputs, all this in intuitive and aesthetical ways (Tidwell, 2011).

Another concern taken into account in the design of our framework was the unification of the layout of the platform, between the mobile or other clients. This led to the use of the Android OS Webview component, that enabled the interaction and display of HTML content, which also allows the use of exactly the same UI both on the mobile phone, and on a standard PC interface.

The presentation layer is decoupled from the processing backbone. In the Android case, the Javascript interface is configured to receive commands from JSI; in a client running in a browser, it is configured to receive commands from a Websocket.

The mechanism behind this versatility is hidden in the script contained in the HTML file. This script is responsible for the seamless behavior of the UI either on the browser or in the Android: all the commands to be executed by the JavaScript world are handled by an object bound to the `JSI` instance in Android. So, if it does not exist, it is created as a JS websocket, linking the page in the browser directly to the server.

In order not to computationally overload the Webkit engine, responsible for rendering the HTML page with the layout (as it does in the built in browser of the OS) all the processing is done at the Java backend, before reaching the Webview, since, in addition, we believe that, in Android, native methods are faster and more efficient than web scripts.

3 MobileBIT PROOF OF CONCEPT: CardioWatch

To demonstrate the potential of our framework, a demo application was built, entitled ”CardioWatch”. The goal of this application is to show the subject heartbeat, measured using a bioPLUX, biosignal acquisition device connected to an 1 lead ECG sensor. The acquisition unit is interfaced with the Android environment via Bluetooth. On the other end, there is a remote server that performs the signal processing, which is accessible over the web.

Inside the Android-powered device this is materialized through a system composed by a sequence of four blocks: a `BioPlux`, a `Websocket`, a `FlotBlock` and an `HTMLfield`, which are, respectively, the virtual representation of: the acquisition device; the implemented socket connection to the server; the ECG line chart; and the heart rate (HR) numeric display. The Data Processing Language structure, corresponding to our application example, is the following:

```
{"BioPlux:Bio":{
        "config":{"MACaddress":"12:34:56:78:9A:BC"},
        "in":{},
        "out":{"0":"ECG"}},
 "Websocket:Wskt":{
        "config":{"Port":"9999",
            "Host_URL":"123.456.7.890"},
        "in":{"0":"ECG"},
        "out":{"0":"SV","1":"HR"}},
 "FlotBlock:Flot":{
        "config":{"FlotDiv_id":"graphHolder"},
        "in":{"2":"SV"},
        "out":{}},
 "HTMLfield:HR":{
        "config":{"FieldDiv_id":"hr"},
        "in":{"0":"Wskt/1"},
        "out":{}}
}
```

Here we can see how the connections between the input (`in`) and output (`out`) channels of the blocks are performed and also how the configuration parameters (fields under `config`) vary amongst the blocks. Each block extends the abstract `Block` class, although they perform very distinct tasks.

For instance, the BioPlux block implements methods for controlling a bioPLUX (Gamboa and Silva, 2007) device via Bluetooth. This device is connected to a pair of electrodes that gathers the ECG signal from the hands, at a configurable sampling frequency that can go up to 1000Hz, and sends them to the Android device, namely to the BioPlux block (which ends up being a virtual image of the actual device).

The Websocket is a block that implements a full-duplex single socket connection (Websocket, 2012) over which messages can be sent/received to/from a remote server. With this, the gathered signal can be remotely filtered, analyzed and returned along with some of its features. In this demo case, at the server, the signal is filtered, downsampled, and the instant heart rate is computed and retrieved. This functionality provides a rapid prototyping framework of complex algorithms in a mobile environment, that otherwise would require more time and effort to implement using more traditional programming approaches.

At the mobile device, the Websocket block re-

ceives the JSON message from the server, parses it, and notifies the FlotBlock and HTMLfield blocks, to present the data to the user interface, namely the real-time, filtered ECG signal and the heart rate. FlotBlock owes its name to the Javascript library used for plotting in jQuery, that produces graphical plots of arbitrary datasets on-the-fly on the client-side(Flot, 2012).

The application connects automatically to the server on start-up, and as soon as the start button is pressed, the Android device connects to the bioPLUX unit and starts to receive real time data. The bottom panel displays the processed data, namely the filtered ECG signal trace, and the heart rate information.

4 CONCLUSIONS AND FUTURE WORK

In this paper we presented a rapid application development framework for mobile devices. It can be used both for standalone or client/server applications.

The presented framework is based on a MVC paradigm for rapid application development, that cano be used by researchers and clinicians to facilitate the prototyping of mobile applications.

Alongside, we have devised a Data Processing Language, that can be used to easily define the structure of the end-user application.

Furthermore, the UI of our mobile framework is supported on web technologies, allowing easy configuration of the layout, look-and-feel, among other aspects of the graphical interface with the user, which is fundamental given the latest trends in mobile application development.

The proposed architecture has undoubtable benefits, namely the ones arising from its versatility, due to the fact that the user can define his application behavior by combining the different blocks, and reuse the layout of a web UI.

Due to its sophistication, we assessed the performance of our framework in terms of computation times and message exchange robustness. The profiling done to the demo version shows that the system components running in the native part of the Android have sub-millisecond performance. Since this is a real-time demo applications, and one that communicates with a remote server through the web, most of the time consumed in this communication.

In the overall, tests have shown that our framework performs adequately for rapid prototyping of mobile applications. The example application presented serves not only as a proof of concept but also as way to demonstrate the potential of the platform. The integration of the Websocket block to communicate with a remote server, provides a testbed for functionalities that can be supported by algorithms already featured on the server toolbox.

ACKNOWLEDGEMENTS

This work was partially funded by the IT - Instituto de Telecomunicações under the grant ”Android Biometrics Platform”, by the Fundação para a Ciência e Tecnologia (FCT) under grants SFRH/BD/65248/2009 and SFRH /PROTEC/49512/2009, whose support the authors gratefully acknowledge. The authors also thank the Institute for Systems and Technologies of Information, Control and Communication (INSTICC), the graphic designer André Lista, Prof. Pedro Oliveira, and the Instituto Superior de Educação e Ciências (ISEC), for their support to this work.

REFERENCES

Cooper, J. (2000). *Java Design Patterns: A Tutorial*. Addison-Wesley.

Flot (2012). Flot. http://code.google.com/p/flot/ (last accessed on 30/08/2012).

Gamboa, H. and Silva, H. (2007). bioPLUX wireless biosignal acquisition system. http://www.bioplux.com/bioplux.

Silva, H., Lourenço, A., and Paz, N. (2011a). Real-time biosignal acquisition and telemedicine platform for AAL based on Android OS. *Proc INSTICC International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications - AAL, Rome, Italy*.

Silva, H., Palma, S., and Gamboa, H. (2011b). AAL+: Continuous institutional and home care through wireless biosignal monitoring systems. *Handbook of Digital Homecare*, pages 115–142.

Silva, M., Guerreiro, T., Gonçalves, D., and Silva, H. (2012). A web-based application to address individual interests of children with autism spectrum disorders. *DSAI 2012: 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion, Douro, Portugal*.

Tidwell, J. (2011). *Designing Interfaces*. O’Reilly Media, second edition edition.

Wan, D., Mentzer, K., and Nash, R. (2009). Always on, always connected how technology will transform the future of chronic care. Technical report, Accenture.

Websocket (2012). Websocket. http://www.websocket.org/ (last accessed on 30/08/2012).