A Model-driven development approach to design software for embedded systems using the Android platform

L.V. Cobaleda¹, J. C. Villa², J. J. Yepes², J.F. Duitama¹, J.E. Aedo².
¹ARTICA, Software Engineering Research Group
²ARTICA, Microelectronic and Control Research Group
Universidad de Antioquia
Medellin, Colombia

Abstract— the complexity in embedded systems applications at the present time has been increasing considerably. This paper proposes an iterative and incremental approach to design software applications for embedded systems based on Model-Driven Software Development (MDSD). MDSD helps to develop software by enhancing its reusability and maintainability. The proposed approach is tested by using it to develop a self-monitoring home healthcare system, which gathers and processes information from a generic Wireless Body Sensor Network (WBSN) using Bluetooth technology. The prototype was developed using both the Android software development kit (SDK) and the development platform Eclipse, integrated with IBM Rational Rhapsody plugin.

Index Terms— Android, wireless sensor network, embedded system, methodology, model-driven development.

1. INTRODUCTION

In recent years, Wireless Sensor Networks (WSN¹) have gained worldwide interest, due to their potential for developing applications in a wide range of areas: health, environmental monitoring, control and monitoring processes. In consequence, it is reasonable to expect that many programmers will engage in the development of the next generation of software applications for embedded systems, mobile devices and WSN. Integrating a WSN with a cell phone through protocols is a common design practice in some applications due to the high capacity network connectivity and computational performance of the mobile phone [1]. Various companies have developed mobile devices with different computational capabilities according to people’s needs. In some cases, companies open their application programming interfaces (APIs), allowing anyone to develop their own applications for the company’s devices [2].

Software development for mobile devices is supported by several methodologies and some of these are specifically for embedded systems development [3]. This paper proposes a methodology to software development for embedded systems with the android mobile platform that considers a communication channel between Wireless Body Sensor Networks (WBSN) and mobile phone with Android.

This paper is organized as follows: Section II describes related work. Section III describes the application development for the case study. Section IV describes the proposed methodology. Section V shows platforms and the toolkit used to build an application following the proposed methodology. Section VI presents conclusions and future work.

2. RELATED WORK

The development of embedded systems that integrate devices and mobile solutions is gaining acceptance, especially in telemedicine. In [4] the authors study the possibility of implementing a logging application on a smart-phone to help diabetic patients with their daily lives. These patients have to keep track of measured blood glucose levels, as well as daily routines that affect their condition. Incorporating this functionality into the user’s cell phone gives easy access, mobility and communication capabilities. The intention was to provide tools that enable the patients to understand aspects about their condition and to simplify data entry. The results show that it is possible to create an alternative solution to conventional logging that includes features missing in today's systems.

The Clinication system [5] is a web- and cell phone-based Patient Adherence Management System. This system can send reminders via e-mail, or send text messages to a patient’s mobile. One of the modules of the system is called CellPly, and patients can use this module on a mobile device to control and monitor aspects of their treatment. The patient will receive messages to remind them of their treatment activities, informational messages about the therapy and questionnaires to monitor vital signs. CellPly can be used in addition to the Clinication portal or independently of it.

In the context of therapeutic adherence, another work [6] suggests some elements for working collaboratively to improve the levels of adherence, such as a monitoring system with wireless sensors to capture relevant patient variables like blood pressure, oxygen saturation, heart rate and respiratory

¹ WSN: The Wireless Sensor Networks are distributed devices, self-employed, which use sensors to cooperate in a common task.
rate; a smart pillbox that stores medicines and communicates messages through a wireless channel when it is opened; a call center where medical staff receive alerts indicating low adherence; and finally, an application for managing medical records.

Another work [7] proposes model-driven development of mobile personal healthcare applications. The authors developed an approach to modeling care plans for chronic disease using two domain-specific visual languages (DSVLs). The first allows healthcare providers to model complex care plans, health activities, performance measurements and sub-care plans. The second DSVL describes a mobile device interface for the care plan. A code generator synthesizes mobile device implementation of this care plan application. Unlike these works, in our application communicates with a wireless body sensor network which is managed from the communication and processing of physiological variables. All was developed with MDSD approach.

3. CASE STUDY

This section describes a software application for a cell phone designed for Android operating system that employs a communication channel with a WBSN to obtain and process the information gathered by nodes in a self-monitoring home healthcare context. Moreover, this case study is subsequently used to detail the methodology proposed in this paper.

This work was developed in the context of tele-assistance. It considers a home care scenario for pneumological patients (patients with respiratory disease) who regularly require the measurement of oxygen saturation and heart rate (among other variables) to allow severity assessment of a possible asthma attack. In this case, it would be useful if, after a measurement taken by the pulse oximeter, the personal mobile device not only shows the patient’s measurements, but also sends an alert to the health care provider if the measurements are considerably outside of the accepted range.

This application was developed to facilitate the monitoring of patients with respiratory diseases who are being treated at home by medical personnel. The patient regularly takes a measurement of oxygen saturation, heart rate and cardiac activity with a pulse oximeter node and an electrocardiogram (EKG) node, both connected point to point via Bluetooth to a mobile phone. In the mobile device the information received is analyzed according to the ranges of acceptable values for the patient and, if there is a risk, the mobile device sends a warning message to the health care provider. This scenario is shown in Figure 1.

![Figure 1 Case study scenario](image)

4. METHODOLOGY

The methodological approach proposed in this paper is based on MDSD. This section describes the proposed method and presents an example design of the case study: a mobile application for monitoring home healthcare patients.

A. Model-Driven Software Development approach

MDSD approach attempts to find domain-specific abstractions and make them accessible through formal modeling. It creates a great potential for automation of software production, leading to increased productivity, quality, maintainability, reusability, and manageability of software complexity. This approach is intended to promote the use of formal and abstract models which can be easily understood by knowledge domain experts.

MDSD separates the specification of the structure and essential system functionality from its implementation using platform-specific models [8] [9]. Thus, it is possible to use concepts that are much less tied to the underlying technology and closer to the problem domain. The domain specification is defined in Platform-Independent Models (PIM), and is completely separated from target platform implementation. The Platform-Specific Models (PSM) are created from the PIM and through model transformations. Finally, the application code is generated.

B. Methodological approach

The methodological model-driven approach presented in this work is iterative and incremental; the key issue is to define a number of short-time iterations, each of them having a group of similar steps; the goal is, after finishing one iteration, to refine the system specification and development, resulting in the final complete product. The aim of the first phase is to select and develop some suitable requirements to produce a stable increment. In the second place, it is to adjust the high-level and independent-platform models, in order to achieve a high reusability and maintainability of the software components, and finally the last phase consists of generating the code for a specific platform.
Therefore, the first step is to identify the fundamental functional (FR) and non-functional (NFR) requirements, which are used to establish the base architecture for the following development. The base architecture is set down in formal high-level platform-independent models. After that, the semi-automatic transformation to platform specific code is done and the first version of code is generated and tested. In the next steps, additional requirements are specified and detailed, the architecture is refined and the code generation and tests are completed. These activities are presented in detail in the following section.

![Use Case Diagram](image)

**Figure 2** Use Case Diagram

C. Development of requirements

In this phase it is important to select the suitable requirements that will guide the subsequent phases; i.e. the actors, their responsibilities, some functional features and constraints.

In the case study, several requirements were identified in the first iteration, but only those essential to establish the base architecture were selected. These are related to the communication with two WBNS nodes. Figure 2 shows the use case model in which the principal requirements are specified. Although the system requires communication with two medical devices, this application could require more devices, so the design is projected to be scalable to more medical devices.

D. Application Model

The second phase is intended to create the PIM. The application packages are organized according to the main responsibilities identified in the software system. Focusing on reusability and maintainability is a key issue in this approach. Another activity in this phase is to define the detailed design of classes that conserves the logical distribution made before. Here the design patterns are applied, and the sequence diagrams and state charts are created.

The logical approach in this work is based on the Five-Layer architecture Pattern [10] but applied in a simple way. Specifically, the Abstract Hardware and Operative Systems Layers are not clearly separated because the current requirements are not demanding this partition. Thus, this application is divided into three layers, shown in Figure 3, where the responsibilities concerning the display of information, data processing and technical communication are clearly grouped. This arrangement makes reusability and debugging control easier.
Figure 3 Application Packages

User Interface (UI) layer is responsible for showing the medical measurements to users and for capturing the configuration values. This layer contains specialized view classes corresponding to each medical device. In consequence, it has one view for the pulse oximeter and one for the electrocardiogram. There is also a sub package in this layer, called plot, which is responsible for drawing the electrocardiogram signals.

Application (App) layer is responsible for embracing the specific domain classes. The Devices package is the most important in this layer. It has a class structure where the extensibility of new functionality is easily allowable through the addition of subclasses. This is shown in Figure 4.

Figure 4 Extract of class diagram for Devices package

Classes inside of Technical services (TS) layer implement technical features of communication. In particular, the Bluetooth Serial Port Profile (SPP) was implemented at application level to establish a communication channel between the cell phone and WBSN.

Some of the design patterns adopted in this work are the Observer pattern to notify the Bluetooth discovery process; the Front Controller pattern to draw the electrocardiogram signs; and the Channel architecture pattern to transform the data from medical devices [10]. The sequence diagrams were created and the functionalities were assigned to the classes according to the logical distribution of packages. An extract of this diagram is shown in Figure 5.

Figure 5 Sequence diagram for verifying Bluetooth

E. Code generation

In this phase, the process of transforming the high-level platform independent models into code for a specific platform is done in a semi-automated way. Subsequently, the particular modifications are introduced in the code directly. The code generation for android 2.2 is done with Rational Rhapsody plug-in for the Eclipse software development kit.

The self-monitoring mobile application runs on Android 2.2 or higher in a context of WBSN. The application establishes Bluetooth communication with a WBSN which captures corporal signs such as oxygen saturation, heart rate and cardiac activity. If the patient measurements are outside of an acceptable range, the mobile sends a message to a health care provider.

The Java code is generated with the Rational Rhapsody tool where some parts were written manually. It is important to emphasize that it is possible to accomplish the iterative development process due to the Rational Rhapsody tool capability for keeping the consistency between models and generated code. Figure 6 shows the work with Rhapsody plug-in tool for Eclipse.

This proposed architecture has also been used in the development of an application for an embedded system with Android, named System Integration and Interoperability Medical monitoring for Tele-care (SIMMIT), which is developed by the Microelectronic and Control Research Group. The packages, the class structure and the patterns used were similar. However, the implementation of the channel architecture pattern was different due to the means of communication with external devices, which was achieved...
through a USB\textsuperscript{2}-serial interface. In this way, this architecture evidences a valuable reference framework for developing new embedded applications in this domain.

![Figure 6 Extract of code](image)

**Figure 6 Extract of code**

**F. Testing**

In the deployment process, two WBSN nodes gather data about some vital signs using the cell phone. The pulse oximeter node takes heart rate and oxygen saturation using Bluetooth communication with no difficulty. This is shown in Figure 7.

![Figure 7 Application Running](image)

**Figure 7 Application Running**

5. **Platforms and Tools Description**

In this section, a set consisting of a toolkit and platforms utilized to build the application are described.

This work was developed with the Android software development kit (SDK) and the development platform Eclipse, integrated with IBM Rational Rhapsody. These platforms and tools were useful for modeling use case diagrams, class diagrams, sequence diagrams and state charts; also for generating code for Android 2.2 (Froyo) platform, in order to interchange information with a generic WBSN node.

A. **Eclipse**

Eclipse is a software development kit (SDK) [11] consisting of the Eclipse Platform, Java development tools and the Plug-in Development Environment [12]. This Platform is a multi-language software environment composed of an integrated development environment (IDE) and an extensible plug-in system; it can be used to develop applications in various programming languages including Ada, C, C++, Java, Perl, PHP, Python, Ruby (including Ruby on Rails framework) and Scheme. Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

B. **Android Software Development Kit (SDK)**

The SDK for Android includes a set of development tools [13]. This SDK includes a debugger, libraries, a phone simulator, documentation, sample code and tutorials. The integrated development environment (IDE) is officially supported by Eclipse Plug-ADT (Android Development Tools plugin) and also it can control Android devices that are connected.

C. **IBM Rational Rhapsody**

Rational Rhapsody is a model-driven development environment for embedded systems based on UML [14]. It was primarily designed to accelerate development, manage complexity, enhance testability, reduce costs and improve quality by leveraging the Object Management Group’s (OMG’s) Systems Modeling Language (SysML) and Unified Modeling Language (UML) standards. Throughout the development process, Rational Rhapsody was built to manage complexity through visualization, and helps maintain consistency across the development life cycle to facilitate agility in response to ever changing requirements.

D. **Samsung GT-S5570**

The Samsung GT-S5570 (also known as Samsung Galaxy Mini) is a low-cost smartphone manufactured by Samsung [15]. It was released with the operating system Android 2.2 (Froyo) and Android 2.3 (Gingerbread). The Galaxy Mini is a 3.5G smartphone that offers quad-band GSM at 7.2 Mbit/s and also counts with GPS receiver with A-GPS, Bluetooth 1.0 and WIFI 802.11 b/g connections. The display is a 3.14-inch (80 mm) Thin Film Transistor-Liquid Crystal Display (TFT LCD), capacitive touchscreen of vertical QVGA (240x320) resolution. It also has an ARMv6 600 MHz processor and 384 MB RAM with 279 MB available.

E. **Pulse Oximeter and Electrocardiogram Nodes**

Pulse Oximeter and Electrocardiogram nodes are generic small nodes for health monitoring developed by ARTICA [16]; these are small WBSN systems designed for health monitoring applications. They are composed by MSP430 processor, for the ATMega128RFA1 radio transceiver chip, and the RN-42 Bluetooth chip. Each node communicates with

---

\[1\] USB: Universal Serial Bus
the other via 802.15.4 protocol. In this wireless sensor network, the sink node is the pulse oximeter, which is in charge of gathering information, and sensing the oxygen saturation and heart rate. The Electrocardiogram (EKG) node senses cardiac activity. Figure 8 shows pulse oximeter node and Figure 9 shows EKG node.

![Figure 8 Pulse Oximeter Node](image)

![Figure 9 Electrocardiogram Node](image)

6. CONCLUSION AND FUTURE WORK

This paper describes an iterative, incremental and model-driven methodological approach for embedded systems with the android mobile platform that considers a communication channel between WBSN and the embedded system. This approach is demonstrated through a case study of the design of a self-monitoring home healthcare software system that uses a communication channel with a WBSN. It enhances software reusability, extensibility and maintainability; a layered architecture is proposed to improve the reusability and the maintainability, a polymorphic structure is proposed in the application layer to support the extensibility, and the use of some design patterns are proposed to improve the reutilization. The architecture proposed in this work has been tested in other scenarios in the healthcare context and its usefulness evidenced as reference architecture for the development of applications for embedded systems. Future work will include improving the robustness and reliability in the methodological approach and integration with new Android versions.

ACKNOWLEDGMENT

We would like to express our thanks to the Excellence research Center, ARTICA and to the members of Microelectronic research group and Software Engineering Research Group from Antioquia University.

7. REFERENCES