

Modeling Information From Wearable Sensors

Florence Balagtas-Fernandez
Department of Computer Science
University of Munich, Germany
florence.balagtas@ifi.lmu.de

Heinrich Hussmann
Department of Computer Science
University of Munich, Germany
heinrich.hussmann@ifi.lmu.de

ABSTRACT

The goal of this research is to create a modeling environment for non-expert users to allow them to easily create applications for mobile health monitoring. In this paper, we focus on the design and representation of the modeling constructs that represent information taken from wearable sensors. We want to find a way to represent sensor information such that it is not confusing and is easily recognizable by the non-expert user.

1. INTRODUCTION

Model-driven development is a widely accepted approach to simplifying the development of complex systems. This involves the development of applications by creating a high-level model of the application and automatically transforming this high level model to platform specific code [1]. In this research, we apply the model-driven development approach to simplify the creation of applications for mobile platforms. We concentrate on applications in the domain of mobile health monitoring, which basically involves wearable sensors [2] that collect biological information from an individual, and a mobile phone which stores and provides visualization of the collected information. We are in the process of designing and developing a modeling environment, which is suitable for non-expert users who are defined as individuals who have limited expertise in programming for mobile platforms. Our goal is to create a modeling environment which is easily learnable and usable, and would allow non-experts to model their own applications for mobile health monitoring.

End-user creation of mobile health monitoring applications can be beneficial to individuals, medical institutions and research groups that focus on collecting biological data from individuals. These types of people usually have little or no experience at all in programming such systems, and therefore need either the help of some programmer or have to learn programming themselves which is not an easy task. The following are some example scenarios that illustrate the benefits of end-user creation of mobile health monitoring applications. Suppose a person wants to have a health monitoring application that would keep track of how much he spends on physical activities with the use of a wearable device that records movements and transmits movement information to a mobile phone. However, he wants some customized view of the software and additional functionalities that are not present in the vendor-provided software of the wearable sensor. He could just use a modeling tool that would allow him to easily create this specialized application based on his wishes, and in turn, this tool will create the application for him without him having to worry about low-level programming. Another application is in clinics that provide patients with health monitoring devices that keep

track of each patient's health. Since every patient is a unique case, an application can be tailor-made for each individual needs and as instructed by the doctor. Our last scenario is in the field of medical research. Some research involves having test subjects wear some non-invasive devices to monitor a subject's activities or temperature for instance. The information are then analyzed by the researchers to prove their hypotheses and formulate conclusions based on the collected data. It would be helpful for these researchers to have some tool that they could use to create customized applications depending on the experiments they have devised, and not constantly rely on a programmer to create the application for them.

In the next section, we will discuss different applications and technologies that are involved in health monitoring. We will then move on to discussing a general overview of the modeling framework that we are trying to develop, and then discuss the possible ways to graphically model information from wearable sensors, which is the focus of this paper.

2. RELATED LITERATURE

The advancement of technologies has allowed the creation of small wearable devices that are relatively non-intrusive, but which can measure biological data in real-time and allows the transmission of the collected data through wireless means (e.g. transmission through Bluetooth). Some example commercial systems and researches which applies such technologies are the Nike+iPod Sports Kit [3], MOPET wearable system [4] and MPTrain Personal Trainer [5]. The mentioned systems are designed to help motivate people in their fitness routines with the use of visual feedbacks such as virtual coaches [4] or through aural feedbacks such as voice and music [3].

Aside from fitness applications, these wearable sensors are also used for monitoring the health of individuals with certain ailments, in order to avoid or prevent serious complications. One example is the Alive Heart Monitor from Alive Technologies [6] which is a device that measures the electrocardiograph (ECG) and acceleration signals of a person. It has Bluetooth capability that allows it to transmit data remotely to a desktop or a mobile device installed with the AliveECG software that comes with the kit. A lot of research work uses the Alive Technologies products and are shown on their website¹. One particular work of interest is the MobHealth Framework², which consists of a set of extensible APIs that abstracts the low level data sent by the Alive Tech sensors and allows creating J2ME applications for the sensors from Alive Technologies. The APIs abstract the low

¹ <http://www.alivetec.com/news.htm>

² <http://sourceforge.net/projects/mobhealth/>

level data sent by the device by providing interfaces that represent the information in a more human-understandable form. For instance, the BodyMovement class [7] contains codes that represent body movements such as BodyMovement.LYING, and BodyMovement.STANDING. This class abstracts the information taken from the acceleration signals whose information are represented as numbers such as 00 for lying down and 12 to represent standing. This helps programmers by ridding them the task of trying to decode the raw data themselves. Although frameworks such as MobHealth help programmers in easily creating their applications through method calls, it is still quite difficult to use especially for non-experts. In our proposed modeling framework, we would like to add a layer of additional abstraction by finding ways to graphically represent sensor information. For each type of wearable sensor, we need to know what type of information can be taken from it, and how this information be visually represented such that the non-expert user will know what they can expect in their applications.

For our research, we have taken the Alive Technologies devices as the example target wearable device and the Java Mobile Edition (J2ME) framework combined with the APIs provided by MobHealth as the target code. In the next section we will first give an overview of our conceptual modeling framework and briefly describe the modeling tool that we are currently developing. We will then discuss in section 4 how we propose to model sensor data.

3. MOBIA MODELING FRAMEWORK

The Mobile Applications (Mobia) Modeling Framework is a framework, which we are currently designing and developing in order to simplify the development of applications for mobile platforms. We introduce one module of Mobia which concentrates on the development of applications for health monitoring. These types of applications involve wearable sensors that collect information from a person, and a mobile phone which processes the data taken from the sensor.

Applications that involve separate devices for collecting information are typically difficult to develop, that is why we propose to apply the model-driven approach to the creation of such applications. However, since the target users of framework are non-expert users, we want to create a modeling environment that is easily learnable and features modeling constructs that are intuitive enough.

Figure 1 shows an overview of the components of our conceptual framework which consists of two major components: the *Graphical Model Component* and the *Model Mutator Component*. The *Graphical Model Component* is responsible for the visualizations of the modeling constructs. As seen in Figure 1, we have designed it to be extensible to support other types of modules for solving other problem domains in the area of mobile development. However, for this particular stage of our research, we focus mainly on mobile applications involved in mobile health monitoring. The *Model Mutator Component* is responsible for processing the model and code transformation. Figure 1 shows the possible components of the Model Mutator which are the *model processor* and the *model-code mapper*. The *model processor* handles the interpretation and checking of the model, while the *model-code mapper* is responsible for mapping the processed model to the different frameworks needed to produce code.

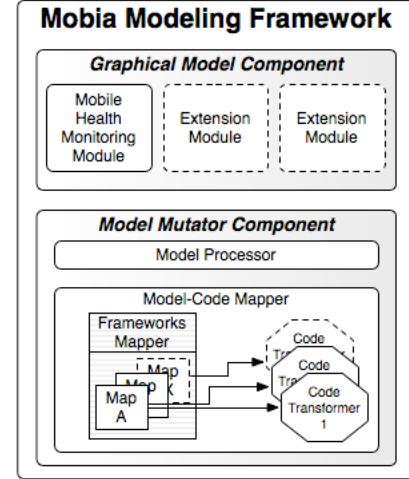


Figure 1: Overview of Mobia Framework

We are currently developing a prototype called Mobia Modeler Suite, which is a tool that allows the simple graphical modeling of mobile applications. Since we target non-experts to be the main users of this tool, we must find ways in order to present information in a clear manner. The need for developing several prototypes of the interface for the modeling environment is important in order for us to evaluate the different possible ways in presenting the interface to the user.

For this paper, we focus on discussing the design issues involved in visually representing sensor data in the Mobile Health Monitoring Module of the Mobia Modeler Suite. More about this will be discussed in the next section.

4. MODELING SENSOR DATA

The importance of trying to represent sensor data in a modeling environment is that, we want users to know what kind of information they can expect from a certain device and help them to be able to immediately visualize what they expect to appear on their applications. Too many technical details about a certain sensor should be abstracted from the user of the modeling tool. We must provide a way to give the user what they initially expect to get from a certain device, but at the same time provide additional information that the user has not initially thought about, but could probably use in their applications.

We have introduced the *medget modeling construct* to represent a certain device in the model. This represents a wearable sensor that can measure specific aspects of health. For research purposes, we focus on three types of medgets: the thermometer, the ECG meter and the actimeter, which measures temperature, ECG and movement respectively. We also assume here that each medget only functions as one device that measures only one specific type of bodily function, although in reality, a medget may measure multiple bodily functions (e.g. Alive Heart Monitor which not only measures ECG, but movement as well through accelerometer signals).

The following points are the things we want to identify in this research:

- How to visually represent *an individual medget* and the information it provides.


- How to show *all the available medgets* in the modeling environment in a manner that it is not confusing to the user.
- How to visualize the flow of information from a medget to the screen.

In order to find out the answers to the points mentioned, we made several designs for the medget constructs and a design evaluation in a form of a survey was made. There were 14 participants to the survey, 8 of which have backgrounds in Computer Science while the others are from Engineering, Mathematics and Physics. The evaluation made was more of a subjective and qualitative evaluation, relying on the participant's previous knowledge and expertise. Details on the approaches we made will be discussed in the three subsections.

4.1 Representing Individual Medgets and their Data

One of the concerns in the design of the Medget is that, what type of information should be available to the user in the modeling environment. We have identified three types of information that could be presented to the user: the name of the medget, a symbol representing it and the data it can provide. For ease of explanation purposes, let's take the Thermometer medget as a concrete example. Table 1 shows the three types of information that can be provided by a medget, and a certain instance of it, which is the thermometer.

Table 1: Example Medget (Thermometer)

Type of Information	Medget Instance
Name of Medget	Thermometer
Representation Symbol	
Available Data	Temperature in Celcius or Fahrenheit

We want to know here, which information matters most to the user. Figure 2 shows all the possible combinations of information that we have presented to the user, and the percentage of which the users have chosen. Take note here that few of the users indicated multiple choices.

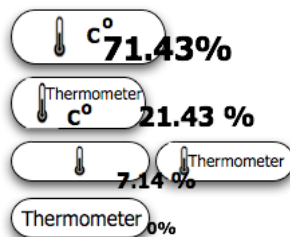


Figure 2: How much and which data does the user need?

A conclusion drawn from the said result is that, the user is not too concerned with the name of the medget as long as the symbols representing it is clear enough for the user. Textual labels were suggested to be shown during a mouseover event for instance. Another thing is the importance of showing the data that a medget provides e.g. Celcius for the Thermometer.

4.2 Displaying All Medgets

Another thing we wanted to find out is with regards to how we can show all the available medgets and the data they provide in the modeling environment. For this example, let us assume that

there are three types of medgets available: medget A, medget B and medget C. We will not name the medgets here since what is important is how they are positioned and displayed, not particularly the information they contain. Now, each medget can have 1 or more types of data that it can provide which are shown as cross symbols.

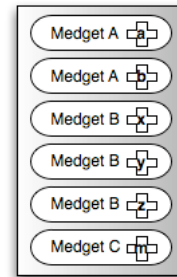


Figure 3: The Individual Display

For the *Individual display* shown in Figure 3, each medget and all possible data it provides is shown as one component in the medget palette. For instance, if there are three available data type for each medget, then three components for that medget are present in the palette.

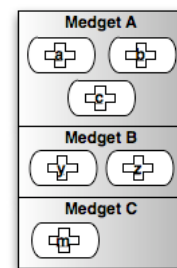


Figure 4: The Grouped Display

The *Grouped display* shown in Figure 4 shows all the possible data grouped according to the medget it is taken from. The medget palette is divided into three sections for the three types of medget, and each section contains all the possible data.

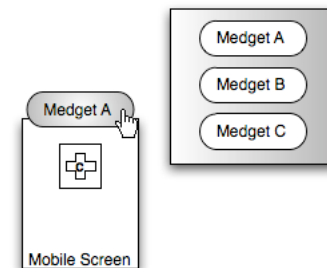


Figure 5: The Minimalist Display

The *Minimalist display* in Figure 5 shows a very minimal representation of medgets in the palette (box on the right). Later on, when the device is attached to a mobile screen (box on the left), double clicking on the medget icon on top of the screen will show a possible data representation for each double click.

Based on the user survey, 46% chose the Grouped display, while 31% chose the minimalist and the rest chose the individual display. The focus is first on the medget that sends out the information and then later on, to the type of data it provides. It is

also easier to look for what the users need when things are grouped.

4.3 Visualizing Flow of Information

One of the design questions that we wanted to answer when trying to visualize information for applications in mobile health monitoring is that, how can we try to represent the flow of information from one medget to the screen? For this one, we have provided 3 different types of techniques.

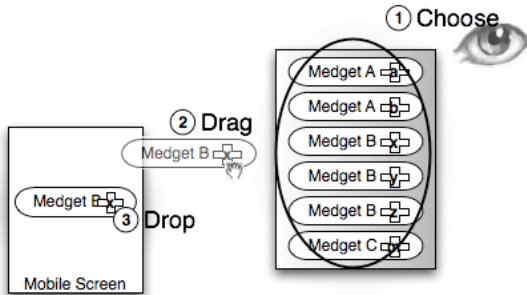


Figure 6: Choose, Drag and Drop Approach

The first one is the Choose, Drag and Drop Approach which is the common approach in programming or modeling environments. Given a set of components in the palette, the user chooses the component and then drags it to the target location.

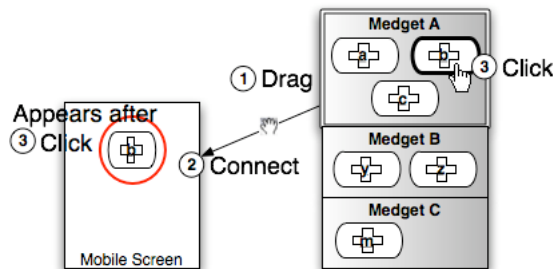


Figure 7: Drag, Connect and Click Approach

The second one is the drag, connect and click approach, in which the user drags an arrow from the medget to the target screen. This arrow will signify information would flow from the medget to the screen. After a connection between a screen and a medget is done, the user would then click on the desired data that would appear on the mobile screen.

Finally, the Drag, Drop and Click is an approach in which the user drags a certain medget to the target screen and the medget is attached to the top of the screen to symbolize connection. The default data provided by a medget will then be shown on the screen. Clicking on the attached medget will show a possible data representation for each double click. For instance, if a medget has two available data representations, the visuals will toggle between the two each time a double click is done.

Based on the survey results, majority of the participants preferred the Choose, Drag and Drop approach (46%). A big factor that may have resulted from this is because it is the concept that the participants were most familiar with. Second choice (38%) was the drag, drop and click approach. Although the least popular which was the drag, connect and click approach actually symbolizes some sort of connection with the medget in the presence of flow arrows in comparison to the real

world, the steps involved seems too deviate from existing approaches.

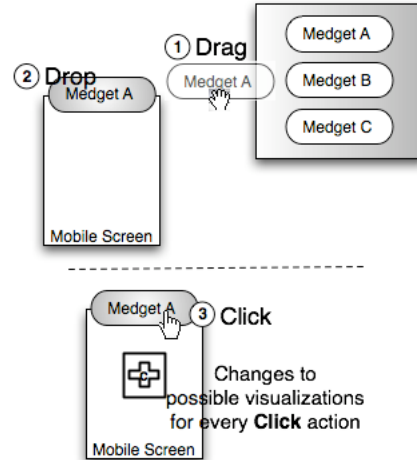


Figure 8: Drag, Drop and Click Approach

In conclusion, although the other methods can be learned easily, the most commonly encountered approach was still the most preferred among the participants.

5. SUMMARY AND CONCLUSION

We have presented in this paper our ongoing work on creating a user-friendly modeling environment for modeling applications for mobile health monitoring. The main core of this paper was to discuss how we can model sensor data taken from wearable sensors. We presented some design ideas that involves medget modeling constructs, how it is presented and how flow of information be shown. Evaluation of the designs were done through a form of a survey. Conclusions drawn from the user survey is that, the most commonly used approach is still the most opted approach by the users. Also, information should be presented in the least possible way, but should be as descriptive as possible whether it is in a graphical form or not.

6. REFERENCES

- [1] Kleppe, A., J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. 2003, Boston, USA: Pearson Education, Inc.
- [2] Korhonen, I., J. Parkka, and M. Van Gils, *Health monitoring in the home of the future*. Engineering in Medicine and Biology Magazine, IEEE, 2003. 22(3): p. 66-73.
- [3] Nike + iPod Sports Kit. <http://www.apple.com/ipod/nike>.
- [4] Buttussi, F. and L. Chittaro, *MOPET: A context-aware and user-adaptive wearable system for fitness training*. Artificial Intelligence In Medicine, 2008: p. 153--163.
- [5] Oliver, N. and F. Flores-Mangas. *MPTrain: a mobile, music and physiology-based personal trainer*. in *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. 2006. New York, NY, USA: ACM.
- [6] Alive Technologies Products. <http://www.alivetec.com/products.htm>.
- [7] Mobhealth Framework. <http://sourceforge.net/projects/mobhealth>.
- [8] Netbeans Visual Library Platform 6.0. <http://graph.netbeans.org>.
- [9] The WeP Project. <http://thewep.org>.