Mobia Modeler: Easing the Creation Process of Mobile Applications for Non-Technical Users

Florence Balagtas-Fernandez
Media Informatics Group
University of Munich
florence.balagtas@ifi.lmu.de

Max Tafelmayer
Media Informatics Group
University of Munich
max@tafelmayer.de

Heinrich Hussmann
Media Informatics Group
University of Munich
heinrich.hussmann@ifi.lmu.de

ABSTRACT

The development of mobile applications has now extended from mobile network providers into the hands of ordinary people as organizations and companies encourage people to come up with their own software masterpieces by opening up APIs and tools. However, as of the moment, these APIs and tools are only usable by people with programming skills. There is a scarcity of tools that enable users without programming experience to easily build customized mobile applications. We present in this paper a tool and framework that would enable non-technical people to create their own domain-specific mobile applications. The tool features a simple user-interface that features configurable components to easily create mobile applications. As a proof of concept, we focus on the creation of applications in the domain of mobile health monitoring. In the future, we would like to extend our work to cover other domains as well.

Author Keywords

mobile application, modeling tools, domain-specific modeling, user-centered design

ACM Classification Keywords

H.5.2 Information interfaces and presentation: User Interfaces—*Graphical User Interfaces*; D.2.2 Software Engineering: Design Tools and Techniques—*User Interfaces, Evolutionary prototyping*; D.2.6 Programming Environments: Graphical Environments

General Terms

Design, Experimentation, Human Factors

INTRODUCTION

With the mobile phone gaining ground in being the most accessible computing device [5], applications for such devices are no longer limited to be created by mobile phone companies alone, but are now extended to anyone with the right skills and motivation (e.g. Android platform, iPhone platform). However, like any other programming task, development for mobile devices is a complicated job. Therefore, we proposed [1] to apply the model-driven development approach in creating such applications. In this approach, an application is generated by first creating a model of the application and later on transforming the model to platform-specific code through transformation tools. The main focus of this paper is to discuss the Mobia Modeler which is a tool that enables users without programming experience to easily build mobile applications through modeling. The Mobia Modeler serves as the front end of the Mobia Framework whose two major components are the modeler mentioned, and the Mobia Processor which is responsible for transforming the models into platform specific code. We followed a user-centered iterative design approach in the development of the modeler's user interface. There have been various designs in the past as mentioned in our previous paper [2], which were tested and lead to improvements that were finally applied to the current design. The design of the Mobia Processor and the process that the model goes through in order to generate the final code will also be presented.

IUI'10, February 7–10, 2010, Hong Kong, China.

RELATED WORK

Various systems allow people with no technical expertise to develop applications. Examples are, ESPranto SDK from Van Herk et al. [10], Vegemite from Lin et al. [7], UI Fin from Puerta et al. [9], and MAKEIT from Holleis et al. [5]. Our work is similar to the researches mentioned in a way that we want to create an environment that would help users with limited technical knowledge, create useful domain specific applications. However, the difference between the Mobia Framework from the mentioned related work is that, the focus is not only on the usability of the modeler for non-programmers, but in the design of the underlying framework as well such that it can be easily extended to support other types of applications for such mobile environments in the future.

MOBIA FRAMEWORK USE-CASE SCENARIO

In order to fully understand how non-technical users can benefit from using a tool like the Mobia Modeler, we present in this section an example scenario in the area of mobile health monitoring.

Increasing Quality of Life with Mobile Health Monitoring

There has been an increased awareness of people with regards to their health. Many technological devices are being created in order to aid people in their goals for better health. Some tools [3] aim to aid people in keeping track of their physical activities and at the same time motivating them through visual and/or audio feedbacks. Others tools [8] help people with existing health problems by monitoring and analyzing physiological signals. However, the problem with these existing applications and tools is that they already contain predefined displays and functionalities. It would be a big help if there was a way to customize these types of applications depending on the type of health related issues that are to be addressed by such applications without the need to do low-level programming. A tool like the Mobia Modeler would be useful in order to achieve this.

Giving the Power to Create: The Target Users

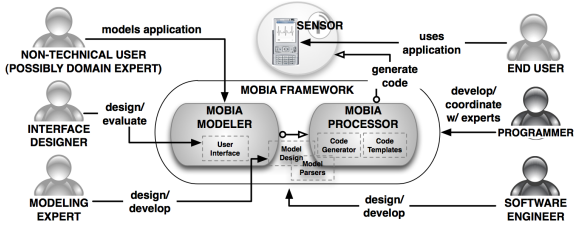

Figure 1. The Mobia Framework and its use cases

As we have mentioned in our conceptual paper [1] about applying domain specific modeling for mobile health monitoring, different types of people can benefit from a tool like the Mobia Modeler. *Domain experts in the field of healthcare* such as doctors, nurses and clinical technicians can create customizable applications for their patients. Other domain experts such as *people in the field of medical research* can also make use of the tool for their own purposes. Other types of users for the Mobia Modeler are *ordinary people* who are just concerned about their own well-being and would like to create their own applications for monitoring their health. For clarification purposes, we want to emphasize that the users we mentioned here are the people who create the application using the Mobia Modeler. They may be the user of the final application (e.g. ordinary people creating applications for themselves), or they may create the application for other users (e.g. healthcare experts creating the application for their patients). Figure 1 shows an overview of the whole Mobia Framework and its use cases.

Health Monitor: A Sample Application

The scenario discussed is adapted from Leijdekkers et al. [6]. A doctor monitors his clinically obese patient by keeping track of his nutrition, physical activities and heart rate. He wants to ensure that his patient is eating the right foods and doing the assigned exercises by getting a daily update on the patient's food intake and physical activities. Since the patient just recently had a heart attack, the doctor wants to ensure that the heart rate does not go over 120 bpm for the next 30 days. In case this happens, the doctor would like to be notified via his pager. He also configures the application to call an emergency number when the patient's heart rate goes up beyond 150 bpm [6].

THE MOBIA MODELER

The main design idea for the Mobia Modeler is *configuration over combination*. This means that instead of building mobile applications by assembling individual user interface elements on the screen, users can just add components to the model and configure these artifacts based on the application requirements. We call these components *Configurable Components*. The formal definition of configurable components in the context of Mobia is: it is a *logical container for multiple user interface elements that has a clearly defined meaning and acts as a whole, and which functionalities can be modified through simple configuration*. The approach of configurable component-based design has been applied to many areas in software and embedded systems. However, according to Fernando et al. [4], there are still issues that need to be addressed in the design of such systems. The key issues emphasized pertain to the attribute-dependent categorization of components, the development and storage of component configurations, and how to provide guidance to the developer/user to choose the right components [4]. We address the issue of categorization mentioned by Fernando et al. [4] by grouping configurable components in the Mobia Modeler into: Basic, Structure, Sensor and Special (see table in figure 2). The issues of development, storage and developer guidance will be addressed in the next sections.

A Step-by-Step Guide into the Mobia Modeler

The modeler starts with a wizard which helps the user configure the modeler's general user interface and supported functionalities. This is the time where the tool collects domain specific properties (e.g. domain, users) and adapts the tool base on the supplied information. The modeler's interface (figure 3) is composed of three main parts: the Main Area, the Menu Bar and the Side Bar. The *Main Area* is the only view used by the Mobia Modeler for modeling mobile applications. Previous studies [11][2] conducted supports this single view design since it increases the learnability of an application. The *Side Bar* contains elements that represent the different configurable components which are grouped according to the four types mentioned in the previous section, and can be added to the screens in the Main Area. To guide users, not all configurable components are available at a given time. Some are disabled depending on the current state of the model. The sidebar can be adapted indirectly through settings made in the configuration wizard during the creation of a new application. The *Menu Bar* is located at the top of the Mobia Modeler's user interface and contains additional functions. All the menu items shown in figure 3 are already supported in the current version of the Mobia Modeler except for the *Simulation* function. The created model can be saved and loaded either through the server or as a local copy. Components can then be added to the main area depending on the application the user wants to create. Each component can be configured by clicking on the pen symbol (see figure 3) on top of each component. Aside from tool tips, the Mobia Modeler also offers visual hints to the user such as the change in color of a config-

Type	Definition
Basic	Functionality common to mobile devices (e.g. notepad, recorder). General Purpose.
Structural	Adds new screen instances to the application (e.g. login, navigation). General Purpose.
Sensor	Represent real-world sensors used with the mobile device (e.g. ECG sensor). Domain Specific.
Special	Represent mini applications (e.g. fitness diary). May be general purpose or domain specific.

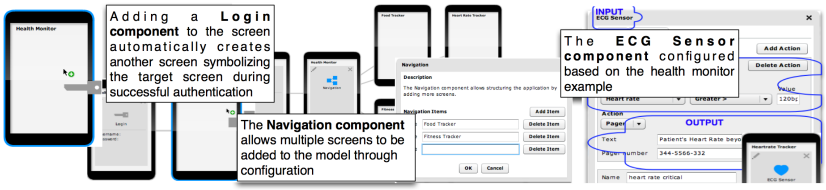

Figure 2. Examples of Component types

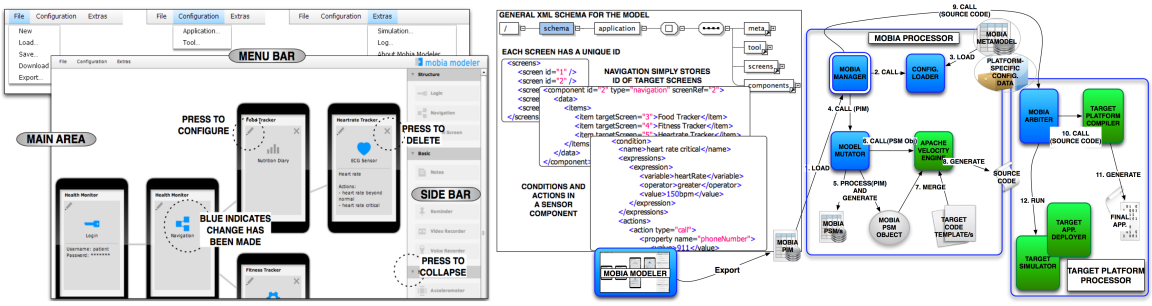

Figure 3. (left) The Mobia Modeler with the Health Monitor sample application model and its (middle) serialized form. (right) The design of the Mobia Processor and the whole code generation process.

urable component if the default values have been changed, or by disabling components in the Side Bar to prevent users from doing invalid moves. Finally, the graphical model can then be serialized into some XML format (see figure 3) that is an important data used for processing the model into code which is the topic of the next section.

From Model to Code: A Look into the Mobia Processor

In our previous paper [1], we have discussed an overview of the initial design of the Mobia Framework including its components. In this section, we will elaborately discuss the Mobia Processor component of the framework and the steps needed to achieve code generation (figure 3).

The process starts when the application model (*Mobia PIM (Platform Independent Model)*) is exported from the Mobia Modeler. The *Mobia Manager* then loads the information from the Mobia PIM into the runtime system of the processor. It then calls the *Configuration Loader* which loads platform specific information (e.g. target platform, code generation templates, *Mobia metamodel*) based on the information specified in the Mobia PIM. The *Mobia metamodel* which is in the form of an XSD file, contains a general description of a model in the Mobia Framework. The Mobia Processor relies on the Mobia metamodel to process the PIM file. For future work, we want to use the Mobia metamodel to easily extend the Mobia Modeler in order to support other domains, or add/modify currently existing components in the modeler's interface. The *Model Mutator* then processes the Mobia PIM and transforms them into *Mobia PSM (Platform Specific Model)*. The difference between Mobia PIM and Mobia PSM is that the Mobia PSM contains additional information that is targeted towards a specific platform. The terminologies PIM and PSM are borrowed from Model-Driven Architecture (http://www.omg.org/mda). The *Model Mutator* then passes the PSM object to the *Apache Velocity Engine* (http://velocity.apache.org/) which merges the information from the PSM and the code templates to generate the final source code. The *Mobia Manager* then calls the *Mobia Arbiter* which is responsible compiling and deploying the final application.

EVALUATION THROUGH QUALITATIVE USER STUDY

A qualitative user study was conducted in order to identify issues that arise from the current design of the Mobia Modeler and find ways to improve it. There were 16 participants in the user study (7 males, 9 females) with the average age of 30. All of the participants have experience in computer usage. 10 of them have no experience in programming, while the other 6 use programming in their respective professions. Although none of the participants have experience in using mobile health monitoring applications, they claim that they understood the concept. In order to *evaluate the different features of the Mobia Modeler*, we observed how the participants interacted with the modeler during the exploration phase and combined it with the participants' answers during the interview (see figure 4). Comparing the views from the two types of participants (programmers and non-programmers or the non-technical people), in terms of understanding the general concepts, usability and design approach of Mobia, it scored higher in the programmers' group as compared to the non-programmers. The variance between the answers of the people in the non-programmers group is also higher which correlates to the different experiences that the participants in this group have.

SUMMARY AND FUTURE WORK

We have presented in this paper the Mobia Modeler which aims to allow non-technical users to create domain-specific mobile applications through modeling methods. We also

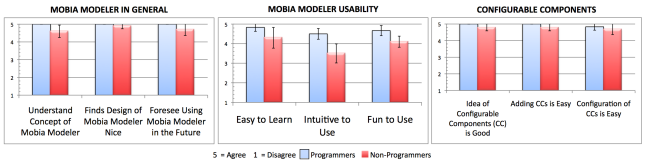

Figure 4. Comparing feedback from non-programmers (non-technical people) and programmers with regards to the Mobia Modeler and its concepts

briefly described the design of the Mobia Processor which works together with the Mobia Modeler in order to generate platform specific code. We combined the use of models and configurable component-based design is our approach. However, our work still has issues which need to be resolved.

- **Provide Richer User Experience and Help.** Users in general are more encouraged to work on a certain task if they see immediate feedback. One thing that is currently missing in the modeler is the simulator that shows the basic functionality of the application being modeled. Another thing that can be added is to provide templates or pre-made models that the user can explore in order to see the capabilities that can be done with the tool. Support for Plug-and-Adapt which automatically adapts the interface based on detected hardware components (similar to [10]) would also ease the initial configuration process.
- **Support for Other Domains and Components.** The underlying Mobia metamodel which is in the form of an XSD file, influences how the Mobia Processor deals with the generated model for code generation. The next plan is to automatically generate and adapt the modeler's interface based on the changes to the metamodel. To further simplify the task, a separate tool may be created such that new domains and its constructs can be added to the modeler without having to manually edit the XSD file.
- **Support for Multiple User Types.** The current target users of the front end (modeler) of the Mobia Framework are novice users. However, as they mature and become semi-skilled users, there should be a way to address their growing needs. More research has to be done in terms of adapting the tool to accommodate this change in expertise.
- **Experience Report.** Getting feedback from the target users/domain experts with regards to the practicality of using the tool for their respective fields is another important thing that needs to be done in the future.
- **Verification and Validation of Generated Artifacts.** Testing and verification of the models generated by the Mobia Modeler and the code generated by the Mobia Processor still needs to be done.

REFERENCES

1. F. Balagtas-Fernandez and H. Hussmann. Applying domain-specific modeling to mobile health monitoring applications. *Information Technology: New Generations, Third International Conference on*, pages 1682–1683, 2009.
2. F. Balagtas-Fernandez and H. Hussmann. Evaluation of user-interfaces for mobile application development environments. In *Human-Computer Interaction. New Trends*, volume 5610/2009, pages 204–213. Springer Berlin/Heidelberg, 2009.
3. F. Buttussi and L. Chittaro. Mopet: A context-aware and user-adaptive wearable system for fitness training. *Artif. Intell. Med.*, 42(2):153–163, 2008.
4. L. F. Friedrich, J. Stankovic, M. Humphrey, M. Marley, and J. Haskins. A survey of configurable, component-based operating systems for embedded applications. *IEEE Micro*, 21(3):54–68, 2001.
5. P. Holleis and A. Schmidt. Makeit: Integrate user interaction times in the design process of mobile applications. In *Pervasive '08: Proceedings of the 6th International Conference on Pervasive Computing*, pages 56–74, Berlin, Heidelberg, 2008. Springer-Verlag.
6. P. Leijdekkers and V. Gay. Personal heart monitoring and rehabilitation system using smart phones. In *ICMB '06: Proceedings of the International Conference on Mobile Business*, page 29, Washington, DC, USA, 2006. IEEE Computer Society.
7. J. Lin, J. Wong, J. Nichols, A. Cypher, and T. A. Lau. End-user programming of mashups with vegemite. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 97–106, New York, NY, USA, 2009. ACM.
8. N. Oliver and F. Flores-Mangas. Healthgear: A real-time wearable system for monitoring and analyzing physiological signals. In *BSN '06: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, pages 61–64, Washington, DC, USA, 2006. IEEE Computer Society.
9. A. Puerta and M. Hu. Ui fin: a process-oriented interface design tool. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 345–354, New York, NY, USA, 2009. ACM.
10. R. van Herk, J. Verhaegh, and W. F. Fontijn. Espranto sdk: an adaptive programming environment for tangible applications. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 849–858, New York, NY, USA, 2009. ACM.
11. M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, New York, NY, USA, 2000. ACM.