Automatic Generation of Multi Platform Web Map Mobile Applications

Marta Cimitile¹, Michele Risi², Genoveffa Tortora²
¹Unitelma Sapienza University
Rome, Italy
marta.cimitile@unitelma.it
²University of Salerno
Fisciano(SA), Italy
{mrisi, tortora}@unisa.it

Abstract — The development of current mobile applications is a challenging task because mobile devices are characterized by a variety of advanced services respect standard computers. In particular, these services as an example Map visualization, Web access, GPS localization, Camera, Accelerometer, and so on, interact with the applications in different ways, depending on the used mobile device platform. These challenges are further increased by the fact that each platform needs a different development process and provides a different framework to implement these mobile applications. In this paper we propose a common architecture and an unified development process implementing portable applications based on mobile services. This architecture is based on the Model-View-Control design pattern and provides a framework that generates the code starting from a formal algebraic specification. This specification integrates different formalisms such as LTL formulae and functional programming, permitting the description of structure and dynamic aspects of a mobile application. Moreover, we show how the proposed framework allows to generate Web map applications integrating different mobile services.

Keywords- Mobile Computing; Platform-Independence; Formal Specification; Web Map Applications.

I. INTRODUCTION

Nowadays the increasing and intensive use of mobile devices can be compared to a technological explosion that obviously represents a determinant factor for the success of this segment. The fast diffusion of mobile communications according to [1] is determined by increasing technology and competition. Organizations use new technologies to offer a highest number of services and functionalities to their customers in the way to obtain competitive advantages. This goal is mostly and quickly achieved considering the wireless or mobile communications. Consequently we assist to the proliferation of software applications for mobile devices. These applications must be developed considering and exploiting the several and different characteristics of mobile devices.

According to this aspect, we can observe that differently from a computer, mobile application must be adaptable to the device operational systems and characteristics. In the last years, different systems have been proposed, wherein each one provides different characteristics and functionalities. The more diffused operational systems are: iPhone OS X, Android, Windows CE Mobile and Symbian (Nokia). Mobile devices characteristics that may influence the functionalities implementation can be:

- Screen: The screen dimensions and resolution for different devices can change. It means that developer must design a user interface adaptable to all devices screens;
- Memory: It is a critical resource, changing dependently from device capability. For this reason application developers strive to reduce the application’s memory footprint by, for example, eliminating memory leaks, making resource files as small as possible, and loading resources lazily;
- Application switching: Different devices can switch applications in different way. For example in iPhone only one application can run at a time, and third-party applications never run in the background. This means that application developer must consider switching device characteristics in the way that users can leave their phone application and returning to it later, easily.
- Internet connection: The research browsers, information broadcasting, advertisement, etc. are managed in different way by mobile devices. All the applications that use internet connection must be adaptable to the different protocols, browser, and so on.
- Geographical Positioning System (GPS): Applications must be adaptable to the device Web map management and GPS system to perform geocoding and/or reverse geocoding operations.
- Webcam: Video, biometric recognizing and identification applications are influenced by device webcam features.

From the user’s point of view, the problem of dependence between mobile device characteristics and application implementation is mitigated by the availability of an high number of device applications. In fact they can choose the application that permits to obtain the required service and is easier to be implemented. For the developers, there is a high difficulty to develop some applications that must be adaptable to the several devices. In particular developers are more focused to application adaptability topic than to the creative aspects.
Moreover, each platform uses a different programming language, such as Objective-C for iPhone or Java for Android, and needs a different development process and a different framework to implement mobile applications. This highlights a high degree of challenge in development of mobile native application, especially for novice programmers. A solution provided in literature [6] [7] is to develop a mobile application as a Web application. The advantages regard the portability. This approach allows exploiting the current Web design and development skills. Basing on the proposed approach developer doesn’t use a specific operating system or a specific programming language and his application will run on any device having a Web browser. The main disadvantages regard the impossibility to access directly to hardware features of the mobile device and the difficulty to achieve sophisticated user-interface effects. Furthermore, programmers are in charged to manage and implement all the aspects of the application, such as the data structure, the data flow, the control flow and the objects behavior, and finally the visualization and the interaction control. During the maintenance activity the application will be changed and enhanced for several reasons [9]. It could affect different aspects of the application, so the programmers have to modify the whole implementation. Moreover, some parts of the code hardly can be reused in different applications with the same context.

In this work, we proposed an unified architecture based on a suitable design pattern which captures the essential aspects and the specific characteristics of mobile devices permitting to adapt in an easy and rapid way different mobile applications to specific device systems. Such a proposal is based on the Model-View-Controller (MVC) design pattern, which is a well-established and compelling approach to develop software. Moreover, we provide a framework that enables to integrate within an algebraic specification different formalisms such as LTL formula, permitting the description of the structure and the dynamic aspects of the mobile. This framework allows specifying a mobile application independently from a specific device or operating system and supports programmers managing and controlling all the aspects of the mobile application easily. The proposed approach is particularly suitable for implementing interactive Web map applications.

The rest of the paper is organized as follows: related works are described in Section 2, whereas Section 3 illustrates the development process and Section 4 presents the algebraic framework; a Web map case study is presented in Section 5 and finally conclusions and future works are discussed in Section 6.

II. RELATED WORKS

The proposed framework consists of an architectural model based on a design pattern and algebraic specification formalism aiming to obtain a rapid mobile applications development process.

This topic is largely explored in the last years according to the increasing diffusion of mobile devices and mobile applications [1]. In [1] a code generator to develop mobile applications is proposed. The approach allows generating the code independently from the required language. Although the code brings several benefits, its design has been limited to creating applications that can perform http connections and database interactions. In [11] a framework to face the problem of device fragmentation of mobile applications is presented. Even if the approach is of interest, it is very new and there are not applications in real context. This framework generates a client application that communicates with a remote server to run the developed application. Respect our approach, the generated application requires a net connection to run correctly.

For this reason own proposal aims to generalize the development process starting from a more high level. In fact the proposed approach supports the development of adaptable applications from the architectural level. The main architectural models adopting an interactors based view are the MVC [13].

MVC software pattern is widely used in web and mobile application [15][8]. It is a three-layer application; each layer corresponds to the following components: Model, View, Control. The Model component includes the core of application data and logic domain functionality. The View obtains data from the Model and displays them to the user. The Controller receives and interprets input into the requirements for the Model or the View. MVC decouples data access code from business logic code, and presentation code, allowing us for greater flexibility and possibility for reuse. Moreover, MVC provides a powerful way to organize systems that support multiple presentations of the same information.

The most common algebraic specification formalisms (temporal/modal logic) are widely described and compared in [2][5] on the base of their expressivity, readability and efficiency capabilities. We used Linear Temporal Logic (LTL), an algebraic model incorporating the concept of transaction. This logic is the basis for formalizing dynamics. It constitutes, together with an algebraic part (which serves as semantics for structural aspects), a firm mathematical framework to give semantics to object specifications including transactions [3]. In own approach, the use of simple version of LTL allows to define events and possible user interactions occurring during the execution of the application.

Finally, in the way to work with Web applications, we need to a Web application framework that implements the MVC design pattern. This topic is largely discussed in literature. An interesting solution is the Struts framework [15]. Apache Struts is a free open-source framework for creating Java Web applications. The Struts framework is designed to help developers to create Web applications that utilize MVC architecture. In particular, Apache Struts exploits different technologies to implement these applications, such as Java Server Pages, XML configuration and Java programming language. We chose to refer to this framework because it allows implementing Web applications by separating the control flow with the main business logic of the application to be developed. Differently from the Apache Struts, our approach defines a mobile application...
exploiting an integrated algebraic specification.

In the literature, different approaches have been proposed to integrate Web maps into mobile applications [10], or into Web applications [14] [17].

In [10] a mobile digital maps system based on Google Maps is defined through system topology architecture and key realization technologies. The approach allows deploying and using a digital map on the mobile device by using an ad-hoc network structure.

A rapid software prototyping approach based on new Web tools has been presented in [14]. In particular, the approach uses AJAX to integrate Google Maps into a Web application. Differently from our approach, the application functionalities have to be completely implemented by the developer.

III. THE DEVELOPMENT PROCESS

The main process that we have defined to model and develop a mobile application is shown in Fig. 1. It is composed of three phases: Definition of the Application, Platform Configuration, and Integration and Deployment. The rounded rectangles represent process phases whilst the rectangles represent the intermediate artifacts produced at the end of each phase.

The first phase allows defining the object data and behavioral aspects of the application. In this phase the views of the application are defined on the object data. The output is a description of the static and dynamic aspects of the mobile application to be developed. This description enables to integrate algebraic specifications with different formalisms such as linear temporal logic and state diagrams, permitting the description of static and dynamic aspects of the application. The algebraic framework enables the specification of both business logic and user interface of a mobile application.

The second phase takes as input this specification and generates the business logic and the user interface of the application described using scripting languages. In particular, the first scripting language is used to manage the objects and the states of the mobile device whereas the latter allows managing the user interface widgets (i.e., the view). This phase takes as input a specific platform configuration file that specifies the available objects and the visualization constraints for a specific mobile platform. In particular, we consider Android, iPhone and Symbian platforms.

In case of inconsistency issues, the developer can repeat the definition phase in order to eliminate the inconsistency modifying the description of the objects and views. Finally, the latter phase integrates the generated code with the application framework. This framework specifies a generic mobile application customized with the objects, views and behavioral aspects defined through the algebraic specification. The output of the integration phase is a customized mobile application framework.

Moreover, this phase is in charge to select the definitive platform architecture. The customized application framework is linked to the specific components of the selected platform using an abstract layer that maps the application objects and widgets into the specific platform objects and widgets. The platform dependent architecture is as follow in Fig. 2.

The object data and behavioral aspects of the generated application are defined in the first layer through the business and presentation scripts and the Application Framework. These aspects use the user interface widgets, objects and APIs defined in the second layer.

Successively, the abstract layer, represented by the Platform to Objects/Widgets Bridge, maps the objects defined in the above layer into the dependent objects of the specific mobile device (i.e., Android, iPhone, Symbian). The proposed architecture enables developers to produce mobile applications which are independent from the underlying platform.

The mobile development process.

In this paper we describe the topmost two layers of the architecture. The third and the fourth layers are implemented using the PhoneGap framework [12]. It is a standards-based, open-source development framework for building cross-platform mobile apps with HTML, CSS and JavaScript for iPhone/iPad, Google Android, Palm, Symbian, BlackBerry, Windows Mobile and more.

In the following we provide details about the phases of the development process. In particular, we propose the

<table>
<thead>
<tr>
<th>Business and Presentation Scripts</th>
<th>Application Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface Widgets</td>
<td>Objects and APIs</td>
</tr>
<tr>
<td>Platform to Objects/Widgets Bridge</td>
<td>Android/iPhone/Symbian platforms</td>
</tr>
</tbody>
</table>

Figure 2. The platform dependent architecture.
algebraic framework to specify the Definition of the application. Moreover, we describe the Platform Configuration phase and finally we introduce the Application Framework component that is based on the MVC design pattern.

IV. THE ALGEBRAIC FRAMEWORK

This section presents the algebraic framework. It integrates different formalisms in order to define the structure of the mobile application together its dynamic aspects. The proposed framework allows specifying in a complete and rigorous manner the mobile application. In fact, the structure of the application is modeled with state diagrams and a behavioral description, whereas events can be expressed by using the temporal logic formalism. Finally, the user interface is automatically obtained from state diagram and arranged using a constraint solver. We used a constraint solver algorithm that is in charge to displace the graphical widgets with respect to the limits of the screen device. The algorithm uses the simple list layout that visualizes the widgets in the order they are defined.

A. The Algebraic Specification

The state of a mobile application is described using the algebraic framework specification. This framework allows the developer to:

- Describe data types, objects and events that the device can manipulate through their communication functions.
- Describe the control flow and the behavioral aspects of the application.
- Describe the user interface.

In a mobile application, an object is the fundamental unit of program. It is made up of three main parts: the structure, the behavior and the events. The first is a collection of named data (attributes) identified by a unique qualified name, the second is a set of methods working on these attributes whereas the events represent the messages that occurs when a particular condition is satisfied. The methods are defined using a functional loosely typed programming approach.

A mobile program is defined using objects and actions. The former is specified when specific functionalities are needed. As an example, we consider the Map object that is in charge to manage the map visualizations. However, an action is a particular object that implements and exports the execute and view methods. The execute method describes the action behavior and how the containing objects are used and managed. The view method provides the objects’ list that will be shown and arranged in the user interface.

The attributes are divided into internal (private) and external (public) type. The former type is invisible from the external and is not accessible outside the object that defines it, whereas the latter is accessible for reading and writing from other objects/actions. The public attributes are automatically saved and restored by a common storage. This storage provides a persistence of objects structure over the changing actions. In particular, these attributes are stored when an action is disposed whereas they are restored when the same actions are recreated, or when the attribute is referenced by another action. The consistency of these attributes is obtained taking into account a qualified unique name.

The mobile application is defined arranging the actions by specifying a state diagram. The state diagram allows to describe the dynamic behavior of the application. In particular, it describes all the reachable states that implement the application and how the events affect each state.

The state diagram is implemented by a state machine that executes the states and the state transitions. We can consider a state executing whereas the state machine waits until an event occurs for that state. In particular, an event occurs in the case a specific triggered condition is satisfied or when user interacts with the application. An event is a relation that links a starting with an ending state, which are not necessarily distinct. If more than one event occurs in the same time, they are serialized using a priority scheme in order to execute one event at time.

The definition of the application also needs a definition of the user interface, which is generated as shown in Fig. 3.

Inside an action the proposed framework allows to define the controls which have to be displayed in the user interface. This is performed using a special keyword as view. Thus, these controls representing graphical widgets are exported by the method View, implicitly provided by the action. Successively, the constraint solver displaces these widgets in the graphical user interface by using a list layout visualization providing an XML description of the layout. This XML description is used to generate the final HTML user interface.

```
abstract action actionbase
{
    data section: //defines attributes
    behavior section:
        init(); //initializes the action
        prepare(); //prepare data before the execution
        execute(); //executes the action business logic
        dispose(); //deinitializes the action
        loadObjects(); //restores the object saved in the repository
        saveObjects(); //saves the objects in the repository
    view(); //exports the graphical controls
    event section: //defines events by exploiting LTL formulae
}
```

Figure 4. The actionbase definition.

Respect the MVC design pattern, the actions represent the Model of our application, whereas the state machine represents the Control, and the XML descriptions are used to implement the View of the model. The communication between the actions becomes through a message passing technique.
Each action is defined by inheriting and implementing a specific abstract definition, shown in Fig. 4. Generally, only the first three methods of the behavior section are overridden. The other methods are automatically implemented by the framework.

A state machine is defined by providing the definition of the actions and a set of messages that link each action with the others. In particular, the state machines are initialized by calling the init method of all the considered actions. When an action is invoked, the loadObjects method is invoked in order to restore all the public attribute values of the current action. Successively, the prepare method is invoked to set up the action and then the view method is called to show the user interface widgets organized by the Constraint Solver component. Then, the action waits for an event. When an event occurs the method execute which implements the business logic of the action is invoked, and finally the saveObjects method saves the public attribute values in a common repository. During the execution of the business logic, a message is set for the current action, thus the state machine can perform the proper transition to another state.

There are two kinds of events: local event and global event. The former is an event that is defined for a specific action. It can occur only when the defined action is active and is running. The latter, can occur independently from the active executed action.

B. Managing the events through the LTL formula

The proposed framework uses LTL formulae to define and manage events. In particular, we consider a simple temporal logic for quantitative reasoning about mobile events. This temporal logic is an event-based linear-time logic; a time frame is considered to be an infinite, discrete and linear-ordered set of events, where an event corresponds to the occurrence of an execution action. There are several temporal operators for linear-time logic that can be used for the algebraic specifications. We consider the following temporal operators:

- $\square$ always, it is an event that has to occur.
- $\diamond$ eventually, it is an event that can occur.
- $\triangleright$ nexttime, it is a chaining of two actions that are executed consecutively. Generally, this event disables the user interaction.
- $\cup$ until, it repeats the current action until an event occurs.

The combination of these operators allows defining a complex events management.

V. CASE STUDY

This section intends to clarify possible usages of the proposed approach by means of an example, trying to cover all the major features of the proposed framework. We propose a simple mobile application that uses the Map, GPS, Geocoding, Calendar and Camera objects. The application shows a Google Map in the main screen of the mobile application and triggers the GPS change events in order to trace a red polyline on the visualized map. In case the user clicks the Photo button, the application allows to take a picture by using the Camera widget. The user can come back to the initial state by clicking the Back button or accept the picture. In this case, other information are computed, the current date is taken from the Calendar widget, and the street name (where the user is located) is computed by using a reverse Geocoding object. These information are successively stored in a database in order to make persistent the computed data. Moreover, the data are used to build a marker and an associated tooltip that the user choose to add on the map. The state diagram of the proposed example is shown in Fig. 5.

![State diagram of the Web map application.](image)

The example has been implemented by using the proposed algebraic framework. Fig. 6 shows the formal specification of the main action Map State.

```java
action MapState extends actionbase
{
    data section:
        public Map map as view;
        public Button takePhoto as view;
        public Button exit as view;
        private Geolocation gps;

    behavior section:
        init {
            map = Map.addMap("m_id"); //creates and adds the map to the view
            map.setCenter(gps.getCurrentPosition()); //sets the center of the map
            map.takePhoto.text = "Photo"; //sets the button labels
            map.exit.text = "Exit"
        }
        execute {
            map.addPolyline(gps.getCurrentPosition(), "red");
            map.setCenter(gps.getCurrentPosition());
        }
        event section:
            $gps.change \rightarrow MapState$
            $takePhoto.click \rightarrow CameraState$
            $exit.click \rightarrow $DoneState;
            $map.marker.click \rightarrow map.marker.showTooltip();$
            //no message is set, thus no change state is performed
}
```

![The implementation of the Map State action.](image)
map by exploiting the current position and then centers the map.

The event section defines the behavior when the Geolocation object indicates that the GPS position is changed. In this case, the Map State is executed again without call the init function. The click events defined on the two buttons allow to perform a transition in different states by setting the CameraState or the DoneState messages. For the latter message, the state machine performs a chaining by executing the Done State and terminating the application without user interaction. Finally, the click event defined on a marker shown on the map, visualizes the tooltip associated to the selected marker. In this case, no message is set thus the state machine does not perform any transition.

Once, the formal specification is completely defined, the developer chooses a specific platform (i.e., Android) to implement the Web map application. In case, the objects used in the formal specification are compatible with the selected platform, the framework generates the business and presentation scripts. These scripts are embedded into a single HTML file.

Finally, the deploy phase is performed by using the PhoneGap [12] framework and installing the generated application on an Android device. The screenshots of the generated Web map application are shown in the Fig. 7. With respect to the other methods, our approach does not need the knowledge of a specific programming language and the Google Map API specifications.

![Web map application screenshots](image)

**Figure 7.** The Web map application screenshots.

VI. CONCLUSIONS AND FUTURE WORK

Starting from the consideration of the necessity to find new solutions to the problem of dependence between mobile device characteristics and application implementation, this paper presents a framework based on an unified architecture, which captures the essential aspects and the specific characteristics of mobile devices. It permits to adapt in an easy and rapid way different mobile applications to specific device systems.

This architecture is based on the Model-View-Control design pattern to describe interactive applications and provides a framework that generates the code starting from a formal algebraic specification. This specification integrates different formalisms such as LTL formulae and functional programming.

A case study has been presented in order to highlight the main functionalities of the proposed approach. The case study shows the applicability of the framework and its capability to manage and control all the aspects of the proposed mobile application easily. In particular, we implemented a Web map application based on the use of Google API features.

Future work will aim at improve the proposed approach and apply it in several applications and contexts. Moreover we will evaluate the advantage deriving from the adoption of the proposed approach in real contests. For this reason we are available to support other research groups in applying the framework and exchanging research results.

REFERENCES


