A Visual Approach supporting the Development of MicroApps on Mobile Phones

Stefania Cuccurullo, Rita Francese, Michele Risi, Genoveffa Tortora
University of Salerno
Fisciano (SA), Italy
{scuccurullo, francese, mrisi, tortora}@unisa.it

Abstract — The definition of an approach supporting an End-User in the development of mobile applications is a hard task because of the characteristics and the limitations of mobile device interfaces. In this paper, we present a visual approach to enable End-Users to compose visually their own applications directly on their mobile phone. To this aim, we propose a touchable interface and an ad-hoc visual language, enabling the user to compose simple focused applications, named MicroApps. The user has not in charge the creation of the user interface that is automatically generated. Moreover, we present the results of a preliminary usability study that revealed a good satisfaction degree of all the involved subjects, whereas an empirical analysis highlighted that the MicroApp visual approach is effective and efficacy.

Keywords- Visual languages; Mobile End-User Development; Mobile Applications

I. INTRODUCTION

Mobile phone applications are becoming more and more popular, and Gartner research expects a market volume that can reach $30 billion in 2013 [6]. Indeed, new services and innovative interaction modalities, including gesture detection, device movement and context-based control are continuously proposed [9]. These innovations are mainly due to the novel and cheap equipments offered by the last mobile phone generation, such as on-board camera, accelerometers, compass, GPS, etc., their increased processing power and fast internet connectivity.

However, the more smartphones become smarter, the more the design of complex applications covering the various user needs becomes a not easy task. Indeed, few of today's available applications take into account that mobile users have different preferences and employ the applications in various situations. In many cases, it is impossible to foresee all the context of use and the actual requirements are often difficult to describe.

One possible solution is to provide to the End-User the means to easily compose and customize mobile applications, starting from a set of available simpler features. Some existing generation tools create the application on the Personal Computer (PC) and successively download it on the mobile device [4]. In addition, as in [1], the proposed composition approaches often require the user to compose the application user interface.

The main idea is to avoid writing the source code. In particular, a mobile application requires the knowledge of different technologies, such as a particular programming language (i.e., Java or Objective-C), different software development kits related to different mobile devices (i.e., Android, iPhone and so on). This involves a high learning curve with a subsequent high effort for End-Users or novice programmers. Moreover, several mobile applications can be modeled as a composition of pre-existing applications/services available on the different mobile devices. In fact, an appropriate handling of these services allows the user to define by himself more complex applications meeting different needs. To this aim, the composition of these applications can be visually modeled through graphical symbols, associated to a particular application behavior and to a specific user interface. By opportunely connecting these graphical symbols, the user can describe complex behaviors.

This paper aims at supporting an End-User in the creation of focused mobile applications, called MicroApps [3]. A naive End-User generates and uses a MicroApp directly on the mobile phone. A MicroApp is designed by graphically composing the functionalities offered by the various phone applications, such as taking an image from the Camera object and saving it (i.e., the Camera.Take and Image.Save actions), retrieving the contacts list from the Contacts object (i.e., Contacts.List action), and sending an email using the Mail object (i.e., Mail.Send action), etc. Each action exposes a description of its user interface that enables to generate automatically the MicroApp user interface. Moreover, the proposed tool assists the user in the composition of the functionalities provided by the mobile device. The result is a specific application based on the user needs that customizes the usage of the mobile device. Finally, a workflow engine enacts the composed process when the application is invoked. In particular, this engine, named MicroApp Engine, is a suitable mobile application that manages and executes modeled MicroApp specifications.

This paper is structured as follows: Section 2 discusses the related work, Section 3 presents the proposed approach for the generation of mobile applications, while Section 4 provides the results of a preliminary usability study to assess the proposed approach. Finally, Section 5 concludes the paper with future works.

II. RELATED WORKS

Mobile End-User development is at the beginning phase and presents new issues mainly due to the characteristics and the limitations of mobile device interfaces. The research interest towards this topic is mainly due to the growing preference revealed by the user towards the services offered by these devices and the need of customizing their applications [4].
In [7] and [9] an approach has been proposed supporting the user in the definition of context-action rules aiming at activating mobile phone functions when the rule conditions are satisfied. Differently from them, we adopt a visual interface to compose a MicroApp that can be more complex than the pattern event-action.

Jigsaw programming has been largely investigated in literature. In this kind of approach, program constructs are represented using icons that look like jigsaw pieces, and only icons that fit together can be composed to form legal programs, see as an example [2], [5] and [12]. More recently, Google has proposed AppInventor [1], that adopts a block editor to create simple programs on the PC that should be downloaded on the mobile device. This editor enables the user to program by using the OpenBlocks programming language [13]. Blocks enable the user to program repeating actions, conditions, information storing, etc. The approach we propose is very similar. It does not require the user to compose the interface and the PC usage, because MicroApps are directly created on the mobile device, and provides a form of assistance in the composition considering the compatibility of the input/output of the various actions.

Authoring tools have also been proposed to compose user-generated mobile services. In [4] Microservices composed by the user can be shared and downloaded. Microservices are created considering two user expertise levels: beginners, enabling a template-based development of a Microservice, and advanced, based on a XML-based language. We propose an approach that assists the user in the application composition, as better detailed in Section III.

An emerging End-User service composition technique is based on the Spreadsheet metaphor [8]: the process is modeled putting the service invocation in the cells. The HUSKY tool [14] enables the users to compose logic spatially arranging of component services within spreadsheet cells. Time progresses from left to right and from top to bottom in cell blocks. Thus, a set of adjacent cells makes a sequence of events. In this case, the information concerning the compatibility among input-output services is not graphically depicted.

III. THE PROPOSED APPROACH

In this paper, we focus on the creation of customized mobile applications, named MicroApp, that the user is able to compose directly on the mobile device [3]. The framework works into two main configurations: model and enactment. In the first configuration, the user composes the application using a Visual Editor. During the enactment configuration phase, a micro application is executed on the mobile device touching its icon or when a specific event occurs.

The proposed approach helps the users to manage the complexity of their activities performed with the mobile device by composing simple applications. The users do not concentrate in managing the dataflow and in the designing of the user interface, but only on the sequence of the actions needed to model the required MicroApp.

Composition approaches have been classified in three groups [15]: Control Flow-based, Data Flow-based and Assisted. The approach we propose is hybrid: on one side, the tool assists the composition by enabling the user to select an individual action from a wide range of actions available on the phone and, on the other side, it allows the user to compose the actions using an ad-hoc developed visual language. A Visual Editor and a Visual Language, adopting a Jigsaw-based programming approach, support the composition of a MicroApp.

In the rest of this section, we describe in detail the development process, the visual composition and the user interface generation of a MicroApp.

A. The Development Process

The overall process to develop and execute a new MicroApp mobile application is composed of three subsequent phases: MicroApp Definition, MicroApp Modeling and MicroApp Deployment, as shown in Figure 1, where the rounded rectangles represent process phases, whilst the rectangles represent the intermediate artifacts produced at the end of each phase.

![Figure 1. The MicroApp development process.](image)

The first phase, MicroApp Definition, allows the user to describe the scaffolding of the MicroApp to be developed. First of all, this phase enables the user to choose how the MicroApp is represented on the device. In particular, the icon representing the application and the visualized name are provided. To complete this phase, a user has to choose an activation modality from the Service Activator list provided as input to the process. In particular, examples of activators are: application menu, user gestures, location, etc. The output of the first phase is an empty mobile MicroApp expressed using an XML notation. An example of MicroApp definition user interface is shown in Figure 2(a). In particular, the camera icon of the developing MicroApp has been selected to represent the application which has been entitled “Take and Send”. The selected activation event is a circular finger gesture. When the device detects this gesture, the application is loaded and then executed.

The MicroApp Modeling phase defines the behavior of the mobile application. In this phase, the MicroApp components are taken from a repository and are composed
Connecting them, respecting the input/output parameter constraints. The output of this phase is a description of the static and dynamic aspects of the mobile application to be generated. The MicroApp Deployment phase stores the MicroApp description in the MicroApp Repository of the mobile device and registers it in the action list that are launched when the selected activation event happens.

B. MicroApp Visual Editor and Language

An ad-hoc developed mobile Visual Editor, designed considering the limited size of the device screen, supports the user in the modeling of the behavior of a MicroApp by composing its application logic.

The main idea is to eliminate all the textual components of a programming language, providing a suitable visual language easy to use and not restricted to simple functionalities. Users can select from a wide range of actions and do not have to define dataflow among them as these aspects are automatically managed. Indeed, to enable the user to appropriately compose a MicroApp, the selection of the actions is supported by an underlying computational algorithms that manage the compatibility of already selected actions.

The proposed approach allows the users to manipulate and connect the actions to build their mobile applications. Thus, the compatibility concept is related to the type and number of inputs provided (outputs received) by the application actions.

Graphically, an application action available on the mobile device is represented by a rounded square containing the application icon, such as Camera and Mail, and the name of the action, such as Preview and Send, respectively. The input/output parameters are represented by colored bullets. In particular, as shown in Figure 3, the input parameters are depicted in the higher part of the square, whereas the output parameters are shown on the bottom.

![Figure 3](image)

Figure 3. Action block examples.

The parameters are differently colored, depending on the type of the corresponding object. As an example, the pink colored parameter represents an Image object, while the cyan colored parameter represents a Contact object, containing the contact data (e.g., name, surname, address, email, cellular phone and so on). Similarly, the red bullet represents a text string and, finally, the yellow bullet represents an email object.

Figure 3 shows some examples of application actions. In particular, in Figure 3(a) the Preview action takes as input an Image object, displays it and returns the same object as output. The generated user interface is described in the next section. In Figure 3(b) the red bulletin parameter, corresponding to a text string, is used to fill the subject field of an email. Moreover, the circled bullet denotes a variable number of parameters of any type. The objects associated to these parameters will be used to compose the email body, and could be indifferently Image and/or Text objects. Once the mail is sent, it will be provided as output.

Let us note that in Figure 3(c) the Contacts action exposes a triangle parameter in the left hand side. This kind of parameter has to be assigned during the application composition, and the associated value will be fixed for each execution of this MicroApp. This means that at design time the user has to select a contact present in the contact list.

Figure 3(d) shows the Take action, that has no input parameter. The Image object provided as output is obtained
by pressing the *Take* button at execution time. In addition, each input parameter type has associated a default action that is able to generate its value. The default action is automatically invoked in case the user at design time leaves it unassigned. It is worth noting that it can be hard for a novice user to remember the meaning of each color parameter. Thus, the editor assists the user providing a description of all the parameters when the user performs a long press on the action icon.

Figure 4 shows some composition rules available to create a MicroApp. In particular, Figure 4(a) depicts an example of a successful sequential action flow. In fact, the output parameter of the topmost action is compatible with input parameter of the lower action. Figure 4(b) shows an example of a parallel action flow, whereas Figure 4(c) depicts an example of joint action flow. In particular, a new action is added to collect the outputs of the actions in the first and second columns. The user initially drops and drops the action on a particular column (i.e., the first column), and successively clicks on the other columns for associating the input parameters to the action, respecting the number and the type of the inputs. Note that if, as in the case of the second column, there is an empty space, the action *Contacts* is automatically lengthened. If the user provides no action in the third column, the text input will be provided by the default action associated to the text parameter (i.e., the *Text.Input* action), otherwise only an action with a textual output parameter should be successively added.

Note that, the association between the action parameters is automatically resolved taking into account their compatibility.

In case an action does not need to use a specific parameter, the editor allows to connect the input parameters of an action with the output parameters of actions that are not in consecutive columns. This case is shown in Figure 4(d), where the Image object provided as output by the *Map.Maps* action is not provided to the *Mail* action. This Image object could be used by other subsequent actions. Let us note that the Visual Editor implements the *undo* operation to remove the last action or input/output parameter association.

It is important to point out that the composition can be performed selecting the blocks in different orders. Indeed, the user could first add the block *Mail.Send* and then position the other blocks. In particular, if there exists a block in the first row of the Composition Area that has at least an input parameter the Visual Editor automatically adds a new empty row by shifting vertically all the blocks by one position, as shown in Figure 5.

Once the user terminates the MicroApp modeling phase, he selects the *Deploy* command. The Visual Editor verifies the inconsistency issues, such as triangle parameters not specified and if all the circled bullet inputs have associated an output parameter. In case of an inconsistency problem, the editor provides the appropriate warning messages; otherwise, if the verification phase terminates successfully, an XML description of the composition process is stored in the MicroApp Repository, ready to be enacted by the MicroApp Engine.

The translation of the MicroApp composition process into the XML description is performed by analyzing the actions in the Composition Area from top to bottom and from left to right directions. This allows to linearize the execution of the modeled MicroApp by sequencing the execution of all the actions. During this step, the forward and backward communication messages between the actions are checked and implemented. When the MicroApp Engine loads the XML description, it translates the description into an execution sequence by instantiating the action objects, and then running the process over them. The control flow is driven by a mechanism that synchronizes the execution of the actions.

Finally, after the end of the deployment phase, the application icon appears in the main application menu of the mobile device. In case a different Service Activator has been selected in the definition phase, the application is also executed when the selected activation event occurs.

C. The MicroApp User Interface

The composition of the user interface is not an easy task for a non programming user. Indeed, there is the need of model the user interaction in terms of GUI elements, such as windows, pull-down menus, buttons, scroll bars, iconic images, wizards, etc.

In this paper we partially follow the approach proposed
in [10] for service composition. In particular, each application action is annotated by a user interface. The MicroApp user interface is automatically generated by combining the annotated actions presentation frontends. When generating a MicroApp, the interface is created following the Microsoft PowerPoint presentation approach: each action is presented as a slide.

As an example, Figure 6(a) shows the interface generated for the Camera.Take action. In particular, the user is able to get an image, touching the Take button, or he can go Back in the control flow or eventually Exit the MicroApp by pressing the appropriate buttons. In practice, the Back button corresponds to an undo operation. Similarly, Figure 6(b) shows the user interface for the Mail.Send action. In particular, a preview of the mail is provided to the user that sends the mail by touching the Send button. The Back and Exit buttons have the same behavior as those of Camera.Take. In particular, in case the user touches the Back button the process is moved a step back to the execution of Text.Input action.

In case an error occurs during the execution of an action (e.g., no Internet connection is available), an error message alerts the user providing the cause that generated the problem, and successively the MicroApp Engine asks the user to perform an undo operation in order to repeat the action again or to exit the application.

The user interface associated to each action is specified using an XML description, that is automatically managed by the MicroApp Engine. As an example, Figure 7 shows the XML description used by the Camera.Take action to manage the layout of its user interface. In particular, each user interface component is specified by an XML tag that represents the graphical widget identified by a unique id. In fact, the TextView tag is used to inject the title of the performed action (i.e., “Camera Take”), the FrameLayout tag is used to visualize interactively the camera preview, and finally the Button tags are properly customized to implement the Take, Back and Exit behaviors.

However, the XML description in Figure 7 is also used by other MicroApp actions, such as Camera.Preview. Moreover, it is worth noting, that some actions do not require a user interface. As an example, the Image.Save action that is executed in background, directly stores the Image object taken as input in the image gallery of the mobile device, and provides the same Image object as output parameter.

![Image 7](image7.png)

Figure 7. An XML description used by the Camera.Take action.

IV. A PRELIMINARY EVALUATION

The proposed approach has been implemented in a prototype supporting users in the composition of MicroApps. In particular, the application is running on an Android based HTC device, by using the SDK version 2.2.

We conducted a preliminary usability study to assess the effectiveness and efficacy of the MicroApp development tool for End-Users during the composition of mobile applications. The satisfaction of the users has been also investigated here. To this aim, the presented study adopts the combination of two techniques: a questionnaire-based survey and an empirical analysis. The context of the user study was constituted of Bachelor students in Communication Science at the University of Salerno. In particular, data for the study have been gathered considering a group of ten volunteers. All the involved subjects had not procedural or object-oriented programming experience, but they are practical with touch mobile phones.

The study has been divided in three steps and performed in one-to-one session (i.e., a supervisor for each subject) using the think aloud technique. In the first step, a lesson of 15 minutes introduced to all the subjects the principles of editing MicroApps and the main features of the prototype. In the second step, we asked to the subjects to perform a task. The task concerned the creation of a MicroApp that exploited the Image, GPS, Calendar and Camera objects posting to a preselected Facebook profile information that states where the user is currently located, together with a photo and the current date. The application uses the camera actions that allow the user to take and save a photo, and successively enables the user to post the information to own Facebook personal area. In particular, the application posts the photo on the user wall, together with the GPS and Calendar data. In the third step, the subjects have to fill in a post experiment questionnaire to collect information on their satisfaction.

During the experiment the supervisor did not provided any help to the subjects to avoid biasing the experiment. He only wrote the comments and problems of the subjects, when they spoke aloud. For each subject the needed time to accomplish the experiment was annotated as well.

The questionnaire we used in this study contained 14 questions, shown in Table I, arranged in three categories: subject experience, MicroApp editing satisfaction, performed task. All the questions expected closed answers according to a five point Likert scale [11]. In particular, the values range from 1 (positive) to 5 (negative).
For the empirical analysis we have considered two variables: (i) the number of minutes required to perform the task, (ii) the number of undo operations performed by the user during the editing of the MicroApp.

In order to represent graphically the distribution of the size measures of the considered usability study, we have adopted the boxplots. This kind of diagrams is widely employed in exploratory data analysis since they provide a quick visual representation to summarize the data using: the median, upper and lower quartiles, minimum and maximum values, and outliers. In particular the answers are visually summarized in Figure 8, whereas Figure 9(a) shows the time performance results and Figure 9(b) the number of performed undo operations.

**Figure 8. Boxplots of the questionnaire answers.**

**Figure 9. Boxplots of the empirical analysis.**

All the involved subjects finished the task, but two subjects generated a wrong MicroApp. The survey shows a good satisfaction degree of all the involved subjects (i.e., question q11), even if they consider the task not simple to perform (i.e., question q9), but hard to complete without using the prototype (i.e., question q14). Moreover, the empirical analysis reveals that the MicroApp visual approach effectiveness and efficacy is appropriate. In fact, 4 minutes is the mean time to accomplish the task performing in average less than 10 undo operations.

**V. CONCLUSION**

In this paper we presented a mobile application and a development process that support the user in the visual composition of customized applications for mobile devices. The proposed Visual Editor was designed for naive users and does not require the user involvement in the specification of the user interface. Moreover, the results of a preliminary usability study revealed a good satisfaction degree and the effectiveness and efficacy of the MicroApp visual approach.

In the future, we plan to improve the usability evaluation of the proposed approach, also comparing it with other existing similar ones, like AppInventor [1]. We are also investigating how the proposed methodology can be extended to other mobile platforms and support Service Oriented Architecture.

**REFERENCES**


