A Prototype for EUD in Touch-based Mobile Devices

José Danado, Fabio Paternò
Human Interfaces in Information Systems
ISTI-C.N.R
Pisa, Italy
jose.carlos.danado@isti.cnr.it, fabio.paterno@isti.cnr.it

***Abstract*—Widespread usage of powerful touch-based mobile devices enables the creation of tools that will allow end users to create their own mobile applications. In this paper, we introduce the Puzzle framework, which supports a visual authoring tool for opportunistically creating mobile applications in touch-based mobile phones. Puzzle focuses on users without programming experience and enables them to playfully experiment and create mobile applications. Puzzle applications can include access to web-services, native phone functions and interactive objects. In the paper, we report on its current architecture and user interface.**

End user development; visual mobile editor; mobile computing; authoring tools

I. INTRODUCTION

The complexity of mobile applications is increasing because of the many possible preferences and contexts of usage. The possibilities are also increased by the ability to combine pre-existing applications or services, access to physical interactive devices and to native smart phone functions. Furthermore, professional developers are sometimes not able to address all the end users requirements. In addition, end-user developers outnumber the number of professional developers. For this purpose, Yahoo!Pipes, MIT App Inventor and Facebook are providing tools to support end user development of applications, but they can be used only in desktop systems. However, the mobile phone is a device that can be explored also in order to opportunistically create mobile applications due to its increasing capabilities and availability.

Our goal is to create a framework, named Puzzle, to allow users to visually create mobile applications directly in a mobile smartphone. End users can start with available applications or from scratch. Afterward, end users can extend their applications to create more complex ones that better meet their requirements. Puzzle intends to be easy to learn and use as well as increase the quality and relevance of the resulting applications for the users. Furthermore, the Internet, and widespread usage of mobile devices are potential elements useful to shift from the conventional few-to-many software distribution model to a many-to-many model. Puzzle can allow for development in fast dynamic environments where end users can easily customize their mobile applications to newer requirements.

Puzzle focuses on a high level approach not just mimicking a traditional language through a graphical metaphor, but providing visual blocks ready to be combined on the go, also decreasing the learning curve and motivating users to use it in daily life. Puzzle intends to decrease the conflict between complexity and learning effort. Reduced learning effort is also supported through tracking and suggesting possible connections to new jigsaw pieces being added to an application.

Puzzle and the associated applications are based on Web-based languages (e.g. HTML5, Javascript, or CSS3) and protocols (e.g. HTTP). Puzzle goes beyond a web mashup tool and enables integration of native smart phone functions as well as interaction with physical objects. One goal is to bridge the gap between what technologies can provide and what end users require to realize a full vision of ubiquitous computing [1].

In particular, the contributions of this work are: *a)* a framework to support creation, modification and execution of mobile applications in touch-based phones; *b)* a visual-based authoring and execution environment that allows end users to exploit different technologies and create applications that fit their requirements; and *c)* a set of graphical metaphors and interaction techniques to support end user development on mobile touch-based devices.

II. RELATED WORK

The metaphor in Puzzle was inspired by previous work in desktop EUD. MIT App Inventor represented the process of building applications in a way similar to Scratch [2], where a traditional programming style is supported by combining jigsaw pieces. However, there is a need for allowing end users to create applications opportunistically in their mobile devices, which motivated us to create Puzzle. Recent advances in smart phones have enabled the creation of mobile EUD environments. Contributions for mobile EUD address parameterization of the mobile terminal [3], frameworks to support mobile authoring and execution [4], mobile authoring tools [5] [4], creation of UI through sketching or by adding interactive techniques in the touch screen [6].

Puzzle leverages on top of these developments to obtain a mobile EUD environment where a jigsaw metaphor similar to Scratch is used. However, Puzzle focuses on a higher level approach not just mimicking a traditional language through a graphical metaphor. Furthermore, Puzzle includes a color help system inspired by the work of Cuccurullo et al. [5] to assist end users on the mobile application development.

III. THE PROPOSED APPROACH

The environment is designed to be accessible and usable for mobile users that do not use programming languages in their daily work. It is intended to support end users to playfully experiment and create applications to support their tasks. Thus, Puzzle uses: *a)* a jigsaw puzzle to convey a left-to-right flow of data; *b)* a color help system to convey possible connections between jigsaw pieces; *c)* a hint system to help users to overcome usage doubts; *d)* drag and drop interaction techniques for creation and modification; and *e)* sliding and popup menus for saving mobile screen space.

The visual environment allows end users to combine various types of functionalities, such as: web services, native phone features, and interactive physical objects.

A. The Environment User Interface

In order to support the description of the metaphors and interaction techniques in Puzzle, we introduce a sample usage scenario: it includes the ability to publish a quote on a Facebook user wall and afterwards be able to easily change that application in order to post that quote as a phone text message to a friend. In this section, we will discuss the metaphors and interaction techniques used in the visual environment to support this usage scenario.

Puzzle is based on touch interaction for object selection, dragging and sliding. The user can drag an object in the screen or slide a previously hidden object. The limited screen space and usage of the finger restricts the view of available tasks and also requires hiding unused objects on the screen. For such purpose, sliding mechanisms, such as scroll bars and sliding menus are used in the UI. A sliding menu is inspired by the physical metaphor of a drawer that once opened shows contained objects and hides them once closed.

In the example, users are initially able to create or execute applications within the Puzzle framework. Creation is illustrated with a jigsaw piece with a plus sign on the bottom right part and a "New" label at the bottom. Available applications are illustrated with a jigsaw piece and a label with the name of the application at the bottom (see Figure 1).

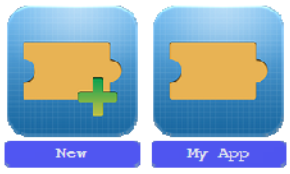

Figure 1. Left) New application - Right) Created Puzzle application

1) Jigsaw Piece

'Jigsaw piece' is a metaphor that evokes the intuitive suggestion of assembling pieces together. Essentially, we allow users to connect jigsaw pieces and compose various arrangements through a series of left-to-right couplings of pieces. Constraining connections in a left to right fashion also provides users with the sense of a pipeline of information flow.

Each jigsaw piece has connections, named inputs or outputs. Inputs are placed in the left side of the 'jigsaw piece' and evoke the ability to receive some information from another jigsaw piece. At the view level, inputs have an inner circular shape suggesting that they will receive an input. At the implementation level, inputs are values (e.g. text, images, or Booleans) received from connecting jigsaw pieces and such values are used during execution of the current jigsaw piece. Outputs are placed at the right side of the jigsaw piece and evoke the ability to connect them to other jigsaw pieces. At the view level, outputs have an outer circular shape suggesting that they will send information. At the implementation level, outputs are values resulting from the execution of the current jigsaw piece and can be used within connected jigsaw pieces.

In Puzzle, inputs and outputs are augmented with a color system that evokes the ability to connect two different jigsaw pieces only if the colors from the input and output of connected jigsaw pieces match. This color system was inspired by Cuccurullo et al. [5] where colors are also used to combine matching blocks. When compared to different connecting shapes, colors were selected as the implementation is easy and flexible within a growing system. Furthermore, we assumed that colors could be easier to distinguish when compared with a growing number of shapes to uniquely describe all matches. The functionality provided by a jigsaw piece is communicated to the user through a label at the top center of a jigsaw piece (see Figure 2).

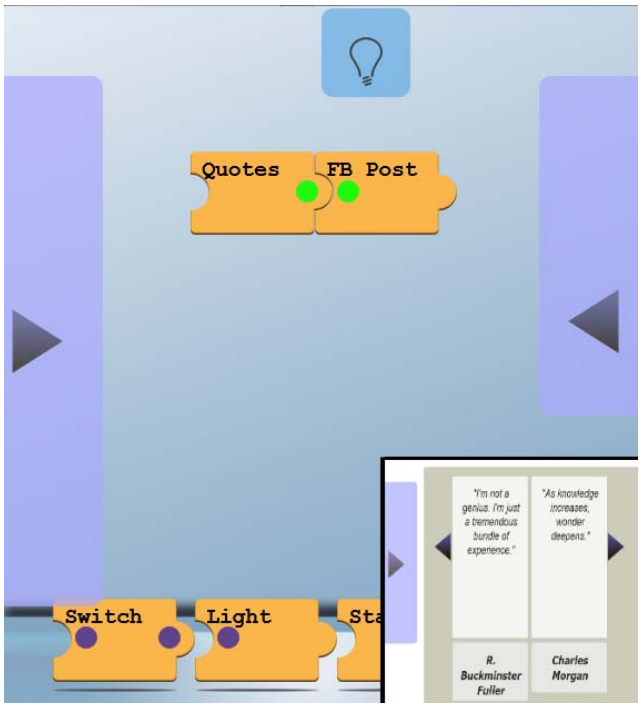

Figure 2. a) Authoring Tool b) Quotes executing (right-bottom)

2) Creation of a new application

Once an end user taps the "New" jigsaw piece, she enters the Authoring Tool (see Figure 2). On both sides of the screen there are sliding menus, a carousel with jigsaw puzzles is presented at the bottom, and a hint button is shown at the top. The button is only available when a hint exists. Sliding menus were selected to save screen space and convey the idea that further options are available. The sliding behavior is conveyed

through arrows suggesting opening and closing them. At the bottom, the carousel allows the user to scroll through available jigsaw pieces in a category also optimizing screen space.

Available jigsaw pieces to create an application can be added throughout the lifetime of the framework, they are divided into categories listed in the left menu. Therefore, the left menu is a list of buttons (categories) that the user can scroll and select. Selection of a category triggers an update of the jigsaw pieces displayed in the bottom carousel. Thus, jigsaw pieces are organized in a two layer abstraction model, where the user needs to firstly select a category and afterwards select a jigsaw piece. Such approach was selected to accommodate a growing number of available jigsaw pieces. The right menu lists the operations available for the Puzzle framework, namely execution, main screen and configuration. Likewise, the right sliding menu is represented by a list of buttons, in this case associated with operations.

At the bottom, a carousel is used to show jigsaw pieces of a particular category. A carousel provides support to easily scroll between available jigsaw pieces. In a carousel: jigsaw pieces are available to be combined; the user has control over viewed jigsaw pieces (e.g. the user is able to view and select available jigsaw puzzle pieces); the carousel prevents users from making mistakes, and is visually appealing.

3) Interaction techniques

Creation of an application relies on *object selection*, *dragging* and *sliding*. In our example, first the user slides the left menu to select the "Gallery" button. As a consequence, at the bottom carousel the related jigsaw pieces are displayed. Next, the user is able to click on the left and right arrows of the carousel to scroll through available jigsaw pieces and find the "Quotes" jigsaw piece.

Visible jigsaw pieces on the carousel can be *selected* by tapping and *dragging* to the center of the screen in order to combine them with other jigsaw pieces. After dragging the "Quotes" jigsaw piece to the center of the screen, the user needs to go to the left *sliding* menu, select the "Social" button and the related jigsaw pieces become visible in the bottom carousel. Next, the user *selects* the "Facebook Post" jigsaw piece and *drags* it to the center of the screen connecting it to the right side of "Quotes" (see Figure 2).

In such scenario, the suggested information pipeline is from "Quotes" to "Facebook Post", and the colors on the connection match, thus allowing both jigsaw pieces to snap and connect. After combining both jigsaw puzzles, the application is ready to be executed. The execution is conveyed through a double tap, or by sliding the left menu and selecting the execution button. In case of development of a new application, a popup dialog is raised to type a name for the application (see Figure 3). Once done, the execution starts.

4) Execution

Execution in Puzzle follows the left to right pipeline defined in the creation. Thus, the UI for each jigsaw piece is rendered. During execution, users are also presented with a sliding left menu with two options "Edit" and "Main" (see Figure 4). "Edit" directs the user to edit the current application and "Main" directs the user to the initial screen were the newly created application is added and listed.

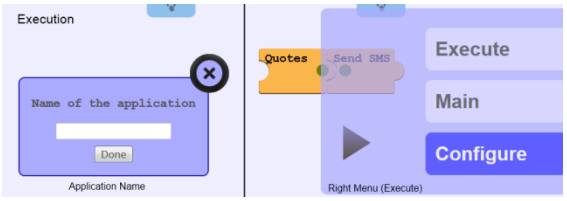

Figure 3. a) Popup dialog to name an application b) Right menu to start execution of the application

Figure 4. Left Sliding Menu (Execution)

The UI of the generated applications is defined in the development of the jigsaw piece. Alternatively, the jigsaw piece could allow for the user interface to adjust to a user configuration. In Figure 2.b), we show a brief view over the "Quotes" jigsaw piece UI during execution.

5) Editing

In order to change the current application to send a text message instead of sending the quote to a Facebook Wall, the user needs to remove the "Facebook Post" jigsaw piece. For that, the user taps on the "Facebook Post" jigsaw piece and a pop-up dialog is activated with a button that allows the user to delete it. Once deleted, the jigsaw piece disappears. Alternatively, users can disconnect the jigsaw piece by dragging it to a free space in the screen.

Afterwards, the user slides the left menu, clicks in the button "Phone" and the "Send SMS" jigsaw piece appears in the carousel. The user can thus drag it and connect it to the "Quotes" jigsaw piece. Testing the edited application is similar to execute a newly created application.

B. Architecture

Puzzle is a Web-based framework and relies on web languages to support execution of the framework and the created applications. The motivation to use web-based languages relates to: *a)* elimination of administrative tasks such as software installation and update; *b)* ability to be used in a wider number of mobile devices; and *c)* reuse of currently available web frameworks within Puzzle.

Puzzle allows the user to integrate information and features that are usually available in the mobile phone but not accessible to web-based applications e.g. access to text messages, or contacts. Interaction with physical objects allows end user developers to create applications that interact with embedded sensors and actuators and also foster user exploration of the possibilities technology can provide e.g. connection to Arduino board to control house appliances and/or their energy consumption.

Puzzle allows users to use, combine and remix data from multiple sources while exploiting contributions from other users. Thus, we can expect Puzzle to benefit from an "architecture of participation" to which end users can contribute with applications and building blocks. Such building blocks are the functions available within Puzzle for building new applications.

Figure 5 describes the Puzzle architecture. The authoring tool, applications and building blocks are stored and assisted by the server managing the framework. The mobile side of the architecture contains a native application including an HTML viewer and a native module.

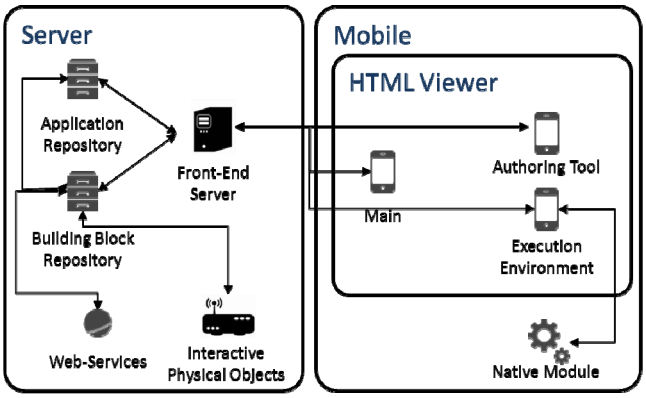

Figure 5. The Puzzle Architecture

The user is introduced to Puzzle through the Main module, which provides the option to create a new application or execute an existing one. The first option directs the user to the Authoring Tool, and the second option directs the user to the Execution Environment. In Main, the Front-End Server is queried to gather a list of applications from the Application Repository. While creating a new application, the HTML viewer is redirected to the Authoring Tool where different requests are made to the Front-End Server to gather information from Building Blocks Repository.

When the application is finished, the user is directed to the Execution module by the Front End Server where applications can request access to the native module (e.g. send a text message), web-services (e.g. post on Facebook) or interactive physical objects (e.g. turn off the lights). Currently, the native module is available for Android devices and integrates native functions, such as: sending a text message, list received text messages or list contacts. The Front End Server allows access to web-services, such as Facebook, or Flickr. Lastly, the Front End Server also includes a connection to an Arduino board able to control an electric plug, house appliances or monitor its power consumption.

IV. Conclusion and Future Work

Puzzle targets users without programming skills to start creating mobile applications on their touch-based phones and further execute such applications in the touch-based phone. In order to address this goal, interaction within Puzzle focuses on usage of a jigsaw piece metaphor, sliding menus and a drag and drop interaction technique. When compared with tools such as Scratch [2], the approach used allows for a higher-level abstraction reducing the learning effort while creating an application in a touch-based mobile device.

In preliminary user tests focusing on end users without an IT background, the acceptance to the metaphors and interaction techniques was positive with users able to create or modify applications through this process. Puzzle allows users to easily create mobile applications from scratch without a previous learning step. The jigsaw metaphor was correctly interpreted by users, the drag and drop technique was easily used, and the abstraction model to select jigsaw pieces to create a defined application was found adequate.

Furthermore, Puzzle relies on technologies such as Javascript, HTML, CSS or Arduino. Motivations to use these technologies include: *a)* allowing the framework to be easily extended and used in different mobile devices, and *b)* allowing the framework to easily grow with contributions from a community of end users and developers willing to support Puzzle and further add, develop and customize jigsaw pieces according to their requirements.

Future work includes analysis of the Puzzle user tests to further improve the current prototype. Namely, address suggestions from users to improve Puzzle, including features such as iteration; selection of a group of jigsaw pieces; zoom/pan, and overall view for easy navigation during development.

Acknowledgment

This work was carried out during the tenure of the ERCIM "Alain Bensoussan" Fellowship programme of the first author.

References

[1] S. Holloway and C. Julien. The case for end-user programming of ubiquitous computing environments. In Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10). ACM, New York, NY, USA, pp 167-172.

[2] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. 2009. Scratch: programming for all. Commun. ACM 52, 11 (November 2009), 60-67.

[3] U. Tuomela, I. Kansala, J. Hakkila, and J. Mantyjarvi, "Context-Studio - Tool for Personalizing Context-Aware Applications in Mobile Terminals," pp. ISBN 1-8649-9738-9, p. 292, 2003.

[4] J. Danado, M. Davies, P. Ricca, and A. Fensel. 2010. An authoring tool for user generated mobile services. In Proceedings of the Third future internet conference on Future internet (FIS'10), A. Berre, A. Gomez-Pérez, K. Tutschku, and D. Fensel (Eds.). Springer-Verlag, Berlin, Heidelberg, 118-127.

[5] S. Cuccurullo, R. Francese, M. Risi, and G. Tortora, "MicroApps Development on Mobile Phones," in End-User Development, M. Costabile et al., Eds.: Springer Berlin / Heidelberg, 2011, vol. 6654, pp. 289-294.

[6] J. Seifert, B. Pfleging, E. Bahamóndez, M. Hermes, E. Rukzio, and A. Schmidt. 2011. Mobidev: a tool for creating apps on mobile phones. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11). ACM, New York, NY, USA, 109-112.