Journal of Visual Languages and Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Puzzle: A mobile application development environment using a jigsaw metaphor

Q1 Jose Danado, Fabio Paternò

Q2 *CNR-ISTI, HIIS Laboratory, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy*

ARTICLE INFO

Article history:
Received 20 May 2013
Received in revised form
14 February 2014
Accepted 22 March 2014

Keywords:
End user development
Ubiquitous computing
Mobile computing
Authoring tools
Metaphors
Interaction techniques

ABSTRACT

Objective: Create a visual mobile end user development framework, named Puzzle, which allows end users without IT background to create, modify and execute applications, and provides support for interaction with smart things, phone functions and web services.
Methods: Design of an intuitive visual metaphor and associated interaction techniques for supporting end user development in mobile devices with iterative empirical validation.
Results: Our results show that the jigsaw is an intuitive metaphor for development in a mobile environment and our interaction techniques required a limited cognitive effort to use and learn the framework. Integration of different modalities and usage of smart things was relevant for users.
Conclusion: Puzzle has addressed the main objective. The framework further contributes to the research on mobile end user development in order to create an incentive for users to go beyond consuming content and applications to start creating their own applications.
Practice: Usage of a mobile end user development environment has the potential to create a shift from the conventional few-to-many distribution model of software to a many-to-many distribution model. Users will be able to create applications that fit their requirements and share their achievements with peers.
Implications: This study has indicated that the Puzzle visual environment has the potential to enable users to easily create applications and stimulate exploration of innovative scenarios through smartphones.

© 2014 Published by Elsevier Ltd.

UNCORRECTED PROOF

1. Introduction

Hardware and software play an important role in supporting professionals, such as architects, doctors, engineers, designers, mathematicians, film directors, and many others [1]. Such professionals exploit complex and powerful functionalities within a set of applications to achieve results for the task at hand. The development of such applications requires considerable effort, and the procedures, methods or approaches used often require adaptations to improve work practices, made necessary, for example, by new regulations or new capabilities being introduced in the applications considered.

In addition, a current technological trend is the increasing availability of smart things that can help us in different tasks. Smart things are physical objects able to interact and communicate with each other and/or with the environment to exchange data and information 'sensed' about the environment, while reacting autonomously to events in the 'real/physical world', and influencing it by running processes that trigger actions and create services. These smart things are networked together; they are able to

E-mail addresses: danado@isti.cnr.it (J. Danado), fabio.paterno@isti.cnr.it (F. Paternò).

http://dx.doi.org/10.1016/j.jvlc.2014.03.005
1045-926X/© 2014 Published by Elsevier Ltd.

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

access Internet services, interact among themselves and with human beings. However, applications and smart things are not directly interconnected, being created by different stakeholders and using heterogeneous protocols [2].

This plethora of applications and smart things often requires application customization and interoperability. Furthermore, constant changes in our world often force people to improvise, evolve and innovate. Such demands are challenges, and users would like to address them creatively while adapting solutions to the problems at hand [3]. The complexity of the problem is further increased with people often facing these challenges on the move, outside of office environments with easy access to their smart phones. In many cases, the mobile phone is becoming the main platform that users are working with to support their daily activities. Thus, there is a need for new tools to support these demands.

One main challenge is to identify how to design application development environments able to support integration of such technologies through intuitive mobile interactive environments. The challenge is further complicated by the limitations presented in mobile platforms with limited screen sizes, usage of touch-based interaction and heterogeneous contexts of usage. The creation of such environments can explore the usage of pictorial metaphors, which are representations of real world objects that can ease the creation of mental models to make the UI intuitive. In addition, a common and agreed upon architecture to support such vision is also lacking. The architecture should be flexible and enable the framework to be interoperable and easily extendable.

This paper discusses a framework named Puzzle, which considers these trends and *enables end users without programming experience to develop or customize their mobile applications*. There are various motivations for the proposal of this framework. Professional developers lack the domain knowledge that end users cannot easily articulate when transmitting requirements for a new application, and regular development cycles are too slow to meet users' fast changing requirements [4]. End user developers outnumber professional developers, thus it is important to develop End-User Development (EUD) tools that are easy to learn and use, and to increase their quality and relevance for the users [5]. The Internet, and wide spread usage of mobile devices are potential tools to create a shift from the conventional few-to-many distribution model of software to a many-to-many distribution model. Lastly, the reason smart environments are still largely unrealized is because research is technology-centric, with inadequate focus on user needs. Thus, creating tools that allow users to develop what they want from smart environments will expand the possibilities where technology can be used to intelligently support user's tasks [6].

Puzzle can connect applications to Web services, native phone functions and existing smart things to start exploring new and innovative scenarios. Puzzle also allows users to dynamically customize their applications in an intuitive and opportunistic mode. A user-centered approach has also been followed, with users providing feedback in some evaluation studies.

This EUD approach was previously introduced in some preliminary work related to the underlying architecture [7] and the possible metaphors and interaction techniques [8]. This paper discusses a new underlying architecture as an enabler for interacting with web services, phone functions and smart things; and the new UI of the environment as an enabler for creating complex applications through a simple process, and reports on new user studies.

The research reported on herein intends to address the following research issues:

- How end-users users can be supported to create services and applications in a touch-based mobile device?
- What technologies can be used to support heterogeneous mobile devices and smart things?
- What level of programming granularity would be suitable for the development of end-user mobile applications?

In Section 2, we summarize the state of the art related to academic and industrial projects that were used as a basis and inspiration to create Puzzle. In Section 3, we introduce a scenario providing an overview on how an end user would interact with the system. In Section 4, we discuss the Visual Environment, metaphors and interactions techniques and how end users are able to interact with it. In Section 5, the architecture is discussed to describe how we have integrated the different technologies within Puzzle. In Section 6, we discuss the evaluation performed including a description of the participants and the methodology used. Section 7 reports the results obtained in our evaluation and, Section 8 includes a discussion of the results. Lastly, we draw some conclusions and discuss the benefits of Puzzle and future work to further improve the framework.

2. Related work

The development platform mainly used in EUD environments for creating mobile applications has been the desktop. The application domains considered range from support through a set of template applications for tourism [9,10], domain-related content management to support guided tours [9–11], collaboration of different stakeholders [11], up to EUD design environments using concepts such as event-based workflow rules [12]. Namoun et al. [13] introduced design recommendations for lightweight service composition environments, considering aspects such as providing guidance to users, balancing difficulty and motivation, graphical development and level of abstraction, previewing of the application, language and terminology, usage of templates, system help and aesthetics. In Puzzle, we have considered similar aspects and further tailored them in a concrete solution to be used in mobile devices. In addition, Puzzle differs from the work done in the ServFace project by Namoun et al., since the target development platform is different and the framework is able to support not only web services.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

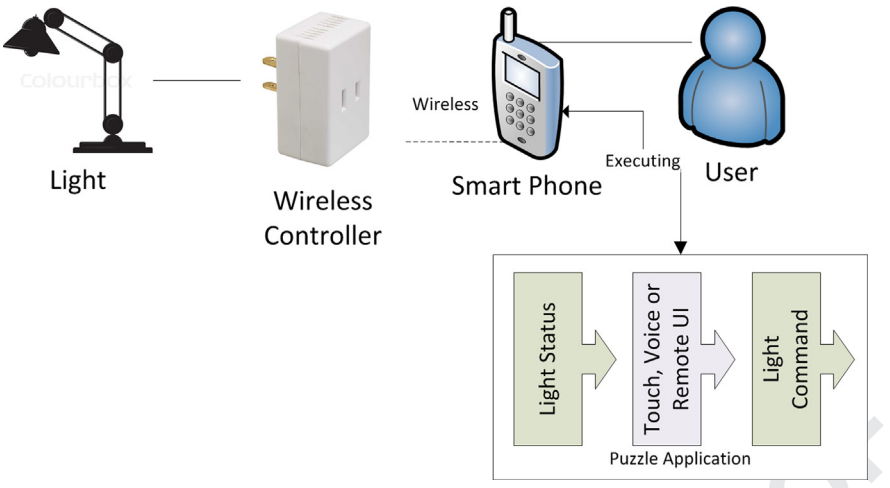

Fig. 1. Schematic view of scenario components.

MIT App Inventor is a general approach supporting building applications in a way similar to Scratch [14], where a traditional programming style is obtained by combining jigsaw pieces. Taking inspiration from such work, Puzzle also uses the jigsaw metaphor and provides support specifically for the development of applications on mobile devices. When compared with Scratch and MIT App Inventor, Puzzle hides programming constructs within the implementation of the functionalities associated with jigsaw pieces and focuses on suggesting to users combinations of interactive functionalities. Consequently, users are not required to know or understand low-level programming constructs to develop an application, but can focus on the functions they would like to put together to achieve the application goal.

When considering EUD designed for mobile platforms the contributions of the current state of the art focus on parameterization of the mobile terminal [15], frameworks to support mobile authoring and execution [16], mobile authoring tools [16,17], creation of UIs through sketching or by adding interactive techniques in the touch screen [18].

Limitations of such environments range from exploitation of only a limited set of web services, if any, to a limited set of smartphone functions available only on one specific operating system. Puzzle focuses on how to support an intuitive metaphor such as the jigsaw in a mobile device and extends these contributions through an architecture that allows users to use several technologies and integrate them in ways that better fit their needs in order to create new and innovative applications. The flexibility and extensibility of Puzzle relies on the usage of widely deployed Web languages (e.g. Javascript, HTML5, CSS3) and protocols (e.g. HTTP), not requiring plugins to access native and external functions, and enable users to customize or reuse existing powerful platforms [19].

TouchDevelop [20] mimics a textual programming environment for the smartphone without introducing interaction modalities more suitable for mobile devices. Atooma [21], Tasker [22] and Locale [23] are examples of Android apps targeting the creation of basic applications that perform one or more simple commands if one or more events occur (e.g. if I am in a meeting then turn off the sound of my phone). When compared to TouchDevelop, Puzzle uses a graphical metaphor instead of a textual approach and hence does not force the end-users to learn even the basic constructs of a programming language. The other tools focus on customization of the mobile phone functions, while Puzzle allows users to intuitively explore available jigsaws representing various types of functions and enabling them to explore and exploit interaction web services, phone functions or smart things in order to enable them to realize their needs. Puzzle is able to exploit various emerging technologies (e.g. RFIDs, Arduino, IEEE 802.15.4 [24], NFC, or 1D/2D image codes). Namely, Arduino is a single-board microcontroller created to ease usage of electronics in multidisciplinary projects and IEEE 802.15.4 boards integrate a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios that can be easily integrated in multidisciplinary projects. Such technologies are getting easier to use, requiring only basic user programming skills and, through user exploration, they can be innovatively integrated to increase the possibilities of use (e.g. a user can benefit from an Arduino to water her plants automatically or activate it through her phone).

Consequently, end-users can be empowered with new building blocks and tools, analogous to those that were emerging during the early phases of Internet growth (e.g. blogs, or wikis) [25]. Integration of these technologies with Web technologies through new tools can allow end users to participate in the Internet of Things in the same manner as in the Internet through Wikis, or Blogs. Puzzle leverages on existing Web technologies and open hardware platforms to allow users to easily create applications for the internet of things as well as to access their own smart things.

3. Example usage scenario

Davide has just bought a wireless light controller that he can manage through his phone (see Fig. 1). Through the usage of Puzzle, he accesses the authoring tool and creates

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

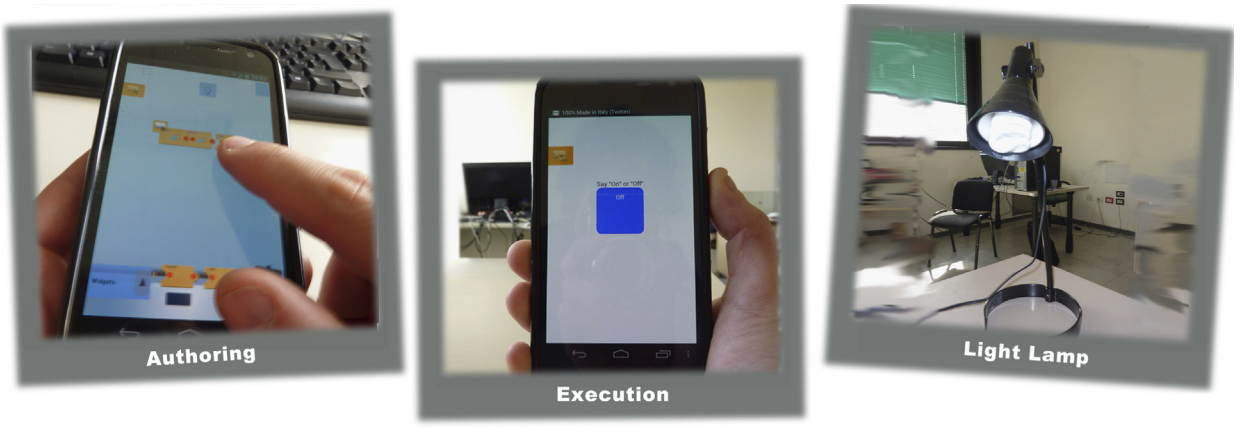

Fig. 2. Authoring, execution, and the light lamp used.

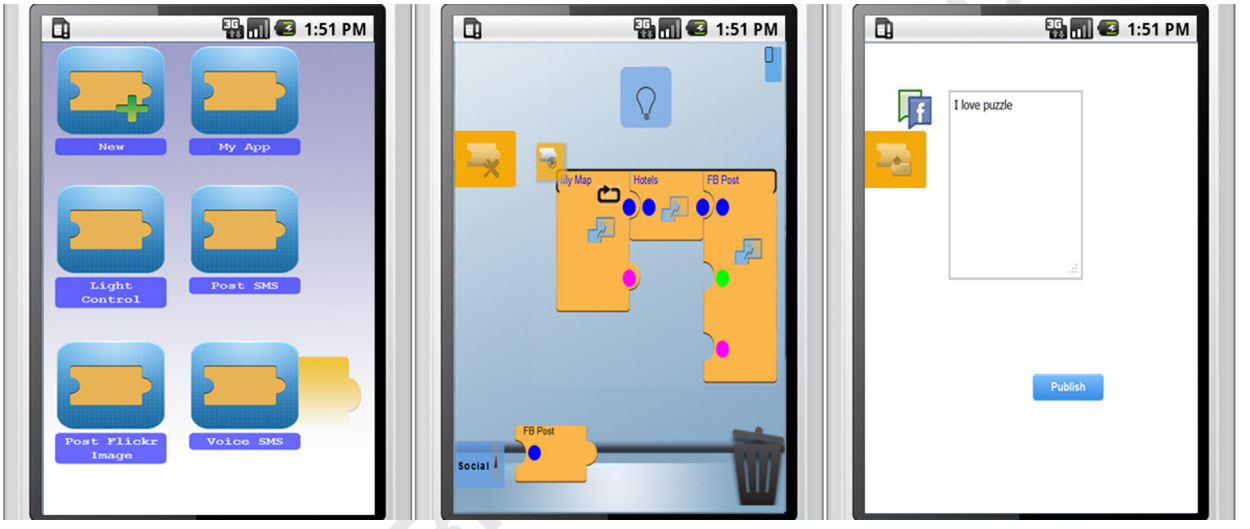

Fig. 3. (a) Start (b) authoring (c) execution environment.

an application that allows him to administrate that controller through his phone (see Fig. 2). He takes his phone to add three jigsaw pieces into the center of the authoring tool canvas. Namely, a first one to check if the lights are on or off, a second one where Davide can send a command to change that status, and a third one to send the command to the controller he just bought.

Davide also likes to change the status of the light using his voice on the mobile phone, or a remote control. In Puzzle, Davide is able to edit the application that he is using tapping on a button that will further open a menu with an option to edit the application. Once in the authoring mode, Davide drags the second jigsaw to a trash can in the right corner to remove it. Afterwards, Davide selects and drags a voice control jigsaw to the position of the previously removed jigsaw in order to be able to test the modified application. In addition, Davide wants to measure the consumption of the house and post a graph of his good usage on electricity on Facebook, and send an alert if a threshold was reached. For that, Davide taps the "New" button on the start screen and starts creating a new application. In his new application, Davide selects and drags a jigsaw that shows his latest consumptions in an image and, at the right of this jigsaw, drags and adds a jigsaw to post on Facebook. Tapping on the execute button allows Davide to start using his application. Furthermore, Davide can edit this application to send the alarm through the same process he has used to change control of the light with his voice. As a result, Davide is able to create and modify his applications through a simple process.

4. Visual environment

The visual environment has three components: start, authoring and execution environment (see Fig. 3).

Start is the initial component presented to users. In this component, users can create a new application from scratch, or execute one of the existing applications. At the beginning, an icon is shown to enable users to create an application and the remaining icons allow users to execute the associated application (see Fig. 3a). The first icon differs from the others because it contains a plus sign in the top left corner to indicate the creation of a new

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

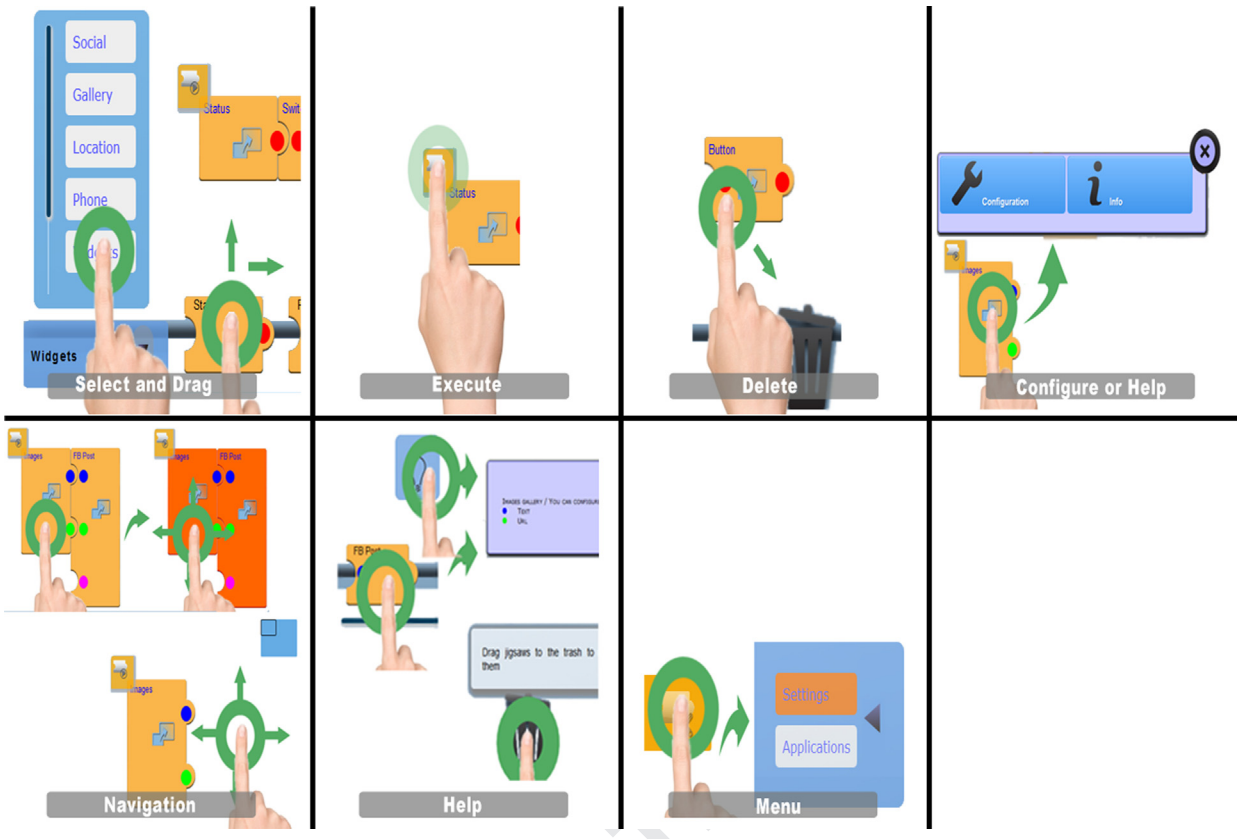

Fig. 4. Possible actions to create or edit a Puzzle application.

application. Moreover, tapping on the icon directs users to the Authoring part.

In the Authoring part, users are able to create or modify an existing application (Fig. 3b). A Puzzle application is developed by drag-and-drop of jigsaw pieces to the center, and connecting them in order to obtain the composed functionalities. The Execution Environment enables the execution of a selected application (Fig. 3c). In the Execution Environment, the UI presented is defined in the implementation of the jigsaw pieces in execution. Additionally, a left menu is overlaid so that users are able to go back to the Start or edit the current application in the Authoring.

Through a user centered design, we have created an authoring tool exploiting the usage of seven key actions that enable users to create, modify and execute mobile applications. The key actions are: selection and drag of jigsaws into the canvas, execute a created application, remove a jigsaw/function from an application, configure parameters within a jigsaw or get help for that jigsaw, navigate within the canvas or move jigsaws within canvas, get help related to the current task/jigsaw or UI object, and access to menu to configure the setting of the authoring tool.

Through a combination of such actions, the user is able to create a mobile application in Puzzle and get assistance on how to proceed (see Fig. 4). This set of actions and the visual representations were designed to take advantage of visual cues and resemble real life metaphors, such as the jigsaw.

The visual environment was designed so that users can be introduced to the framework without explicit training and be able to reason about application development through analogy on how jigsaw pieces are combined to build a Puzzle [26]. The creation of a Puzzle application is based on the usage of the 'jigsaw' metaphor. It takes advantage of the user's familiarity to join jigsaw pieces and foster users to explore combinations of jigsaws. User's easily start joining jigsaws and understand how to plug them together. The pictorial metaphor creates a mental picture that helps users to interpret how the framework works, namely creating an application combining pieces of a jigsaw.

Affordance of an object relates to how the object suggests to be used by the user. In Puzzle, the usage of a center canvas is also included as to afford jigsaw pieces to be dragged, explored and connected in the center of the screen. Once in the center of the screen, jigsaws can be connected. Such connections relate to the association of building blocks' inputs and outputs and describe the data flow. Therefore, inputs afford an inner connection and outputs afford an outer connection. Moreover, a top-down and left-to-right flow of execution is suggested by having inner connections at the left and outer connections in the right side of the jigsaw pieces. Jigsaws inputs and outputs represent what can be connected to them through the number of instances and color. Therefore, a jigsaw is represented through the number of inputs or outputs and also includes a color depending on the type of data

UNCORRECTED

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

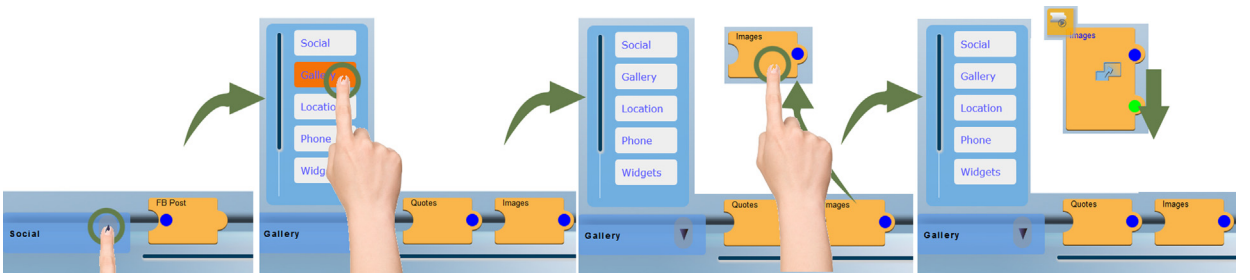

Fig. 5. Method to select a jigsaw from the bottom area and its expansion in canvas.

Fig. 6. Short and long tap on the screen trigger a click or help.

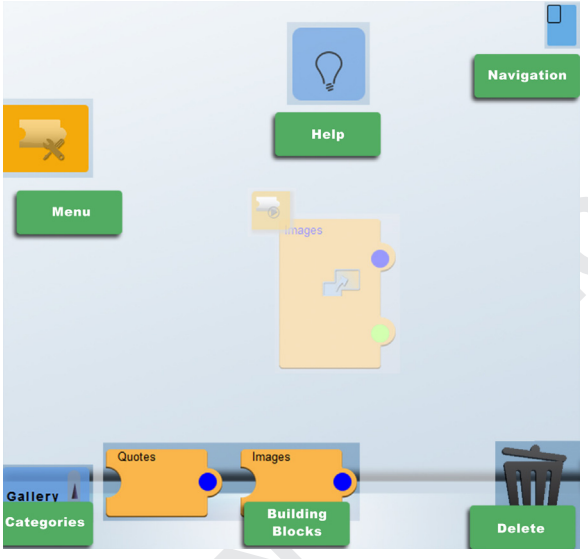

Fig. 7. Authoring tool UI components.

that can be exchanged. The definition of the number of jigsaw's inputs and outputs and their data types is performed by the developer of each jigsaw before the jigsaw can be used within Puzzle.

In order to save screen space, when a user is browsing through possible jigsaws to use at the bottom, jigsaws are represented with only one input and output connection. However, the user is able to get information related to the actual functionality of the jigsaw and its input/output types by taping on the jigsaw.

When moved into the canvas, the jigsaw expands to show all its inputs and outputs. In such a way, the user is able to correctly connect further jigsaws (see Fig. 5). The last picture on the right shows an enlarged view of the jigsaw shown in the previous illustration.

Actions within the framework mainly explore touch-based interactions. Touch durations and dragging actions are used to allow the user to create and modify the applications. Furthermore, the limited screen space requires techniques to show relevant tasks and hide unused objects on the screen. For such purpose, sliding mechanisms, such as sliding menus are often used in its UI. Users are able to touch an object that they are manipulating and activate different actions. A short tap event triggers a click, and a longer tap event triggers help on the usage of the object selected (see Fig. 6).

The layout used for the authoring tool includes the following main interaction spaces placed around the canvas: a menu, help, navigation, building block categories, building block selector and a trash can (see Fig. 7). The menu includes settings for the framework as well as a link to navigate to other applications. Help provides hints that the users can follow if they have problems during creation or modification of an application. Navigation allows the user to move the viewport across canvas. Due to the reduced screen size, the user is only able to see a portion of the canvas.

Users can select and browse categories, which are used to group jigsaws of common characteristics and reduce the amount of available building blocks in the building block selector. Existing categories are named social, gallery, location, phone, widgets, voice and operators. Naming and grouping of building blocks in categories is dynamic and can therefore be arranged differently. The building blocks scroll bar is at the bottom so that users can navigate through the options in the selected category and drag an option to the canvas. Lastly, delete removes a jigsaw from the application when a jigsaw is dragged on top of it.

One construct that is useful even in this high-level development approach is iteration. Iteration is represented through a wireframe around looped jigsaw pieces and highlighted with a loop icon. An end user can include iteration in her application by dragging an iteration jigsaw into a set of jigsaws to be looped. Afterwards and tapping

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

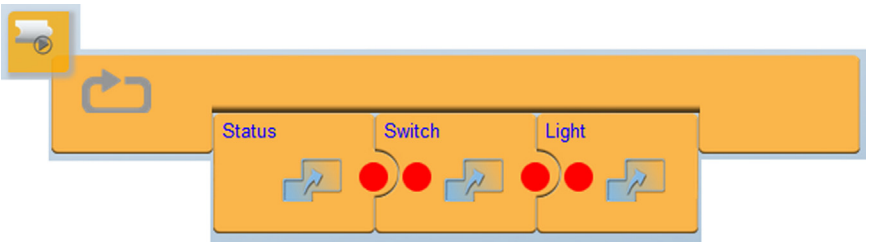

Fig. 8. Iteration jigsaw.

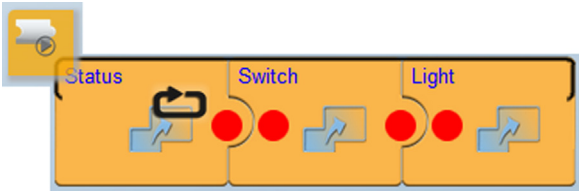

Fig. 9. Iteration wireframe.

on the loop icon superimposed over the looped jigsaws, a dialog is shown where she can define the number of times that the iteration is going to be repeated. The selection is performed through the usage of a slider informing the number of iterations.

This solution was selected from two possible visual representations in order to use less screen space and better represent the corresponding functionality. One option uses an additional jigsaw that embraces the jigsaws that will be looped (see Fig. 8). A second solution does not introduce an additional jigsaw in order to save screen space and uses a black wireframe around the jigsaws that will be looped (see Fig. 9).

Execution is enabled when the user adds a jigsaw to the center of the screen. In our preliminary work, this function was performed through a menu button hidden in the right side of the screen. Thus, a common function was hidden from the user. Furthermore, a user test showed that it was not clear where the execution could start. To address such issue, we included a button that overs on top of the first jigsaw to be executed in the application so that users can immediately activate and test the application and also identify where the execution starts (see Fig. 9, left square).

User participation has been an important aspect while developing the visual environment, since their feedback has also been useful in shaping the framework [27]. Through observation of end users developing applications with Puzzle, we have improved the pictorial representations provided.

4.1. Execution environment

Applications are executed by considering the jigsaws top-down and left-to-right. During execution, the environment selects the jigsaw to execute, and stores and retrieves the necessary associated input/output values. As an example, the UI of an application to send a text message to Facebook shows first the UI to select a text message, corresponding to a first jigsaw, and then the UI to post on Facebook, related to the final jigsaw. The latter UI includes the text message received as output from the previous jigsaw.

Associated with each individual jigsaw is one specific interactive component with UI and logic (building block) that is independent of the platform. Its execution is similar to the execution of a web page in a mobile browser. However, the framework provides access to native phone features without requiring the installation of plugins. An important feature of the framework is the ability for the user to change an application during its execution at any point. This is done through access to a tab icon that once clicked allows users to go to the Authoring part or to the Start screen (see the left orange button on Fig. 3c).

Furthermore, it is possible to change the interaction modality of an application by changing the jigsaw, as in the example shown in Section 3 (Example usage scenario). On a mobile device with a small visual interface and keypad, a word may be difficult to type but very easy to say. In Puzzle, different modalities are supported with potential benefits for the executed applications, such as: making the application easier to use in different contexts; requiring less training; making the application more flexible; considering user preferences; making the task more efficient; or supporting new functionalities.

The framework uses: speech recognition, graphical interaction or remote physical buttons. The first prototype initially supported only graphical interaction but considering mobile scenarios, we found out that users can benefit from other modalities. The user may simultaneously be engaged in other real world activities, thus the interactions need to be minimally disruptive and minimally demanding of cognitive and visual attention [28].

5. Software architecture

Fig. 10 illustrates the implemented architecture for the Puzzle framework. The authoring tool, applications and building blocks are stored in the server managing the framework. The mobile side of the architecture contains a native application including an HTML viewer and a native module accessible through HTTP requests.

At the beginning users access the HTML Viewer. Once Puzzle starts, the Start component requests the list of available applications (connection 1) from the Front End Server. Such list is stored in the Application Repository and delivered to the Front End Server through an SQL request (connection 2). When an application is created from scratch, end users are directed by the Front End Server to the Authoring module (connection 3). Next, the Authoring

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

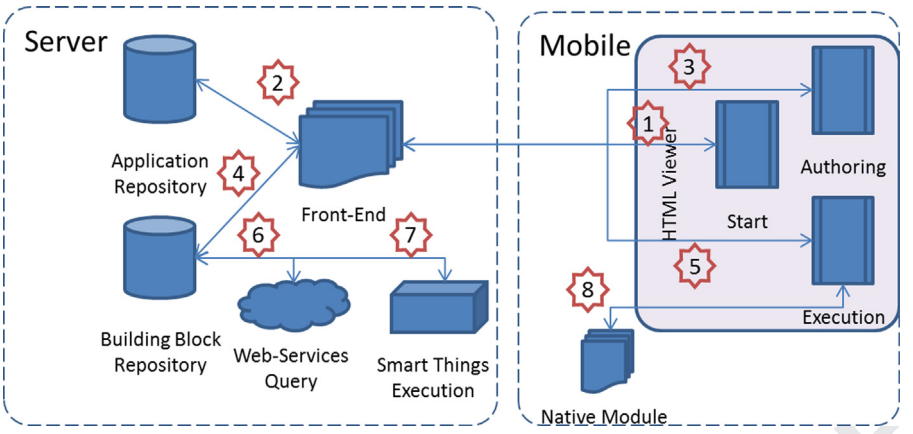

Fig. 10. Puzzle architecture.

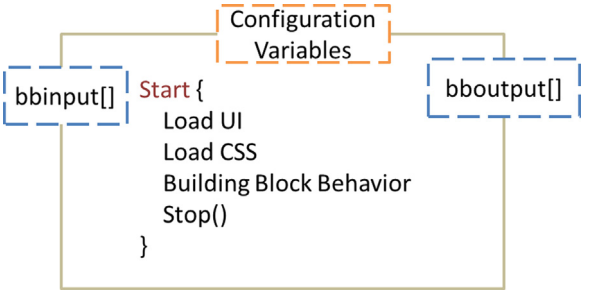

Fig. 11. Basic implementation components of a building block.

module requests the available building blocks from the Front End Server. Such information is stored in the Building Block Repository (connection 4). Afterwards, the Authoring module can request configuration details and further information from the Building Block Repository. When the application is created the Authoring module sends the information about the application to the Front End Server (connection 3) and stores that information in the Application Repository (connection 2).

When the application is created, the end user is directed to the Execution module by the Front End Server (connection 5). During application execution, the executing building block can further request some external services, for example Flickr or Facebook (connection 6). Additionally, a building block can require interaction with physical objects, through connection 7, or request the Native module to execute native functions (connection 8).

Considering the example usage scenario in Section 3, Davide would start creating his application through the Start component. Then, the framework will request the list of applications (connection 1) and receive it (connection 2). Once the list of applications is available to Davide, he can switch to the Authoring mode (connection 3). Once in this mode, the Authoring automatically requests building blocks from the platform so that Davide can start creating his application (connection 4). Composition of an application is performed on the device, allowing Davide to freely and easily explore combinations of jigsaws in an efficient manner. Once Davide has combined jigsaws, he can store (connection 3) and execute (connection 5) the application. At this point, Davide's app is in execution and each jigsaw is executed autonomously. In the example, a first building block requires information about the state of the light (connection 7), a second building block uses voice input and the internal speech recognition system (connection 8), and the l ast building block sends the command to the light (connection 7). All connections are managed through Javascript and transparent to the Puzzle framework. Thus, a building block can also use a web service (connection 6), e.g. Flickr.

5.1. Implementation

This architecture is based on Web technologies (JSP, XHTML, CSS, Javascript, …) for the implementation, thus avoiding the need for specific software installation and update, and enabling reuse of the framework in different mobile platforms. Furthermore, open hardware such as Arduino, or IEEE 802.15.4 boards were also used to enable users motivated to contribute to the framework to learn about such open hardware technologies and further extend the framework with additional functions that can be re-used by everyone. The basic element of the environment is the building block corresponding to a jigsaw, including its UI and logic.

Combinations of building blocks occur through connection of their inputs and outputs. Inputs enable the building block to receive data from another building block. Outputs enable the building block to send data to a following building block. Data exchange also requires a building block to check its pre-conditions and adjust its behavior accordingly. In the scope of Puzzle, we define a pre-condition as a condition or set of conditions that must always be true prior to the execution of a building block. If an input does not support an expected value for the execution of a building block then it violates its pre-condition. Currently, in case of wrong input a building block should perform an action to adapt the value to verify the pre-condition and/or warn the user through a warning message.

A building block is implemented as a Javascript file that performs the following operations (see Fig. 11):

- Dynamically load UI elements to support user feedback or user interaction.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

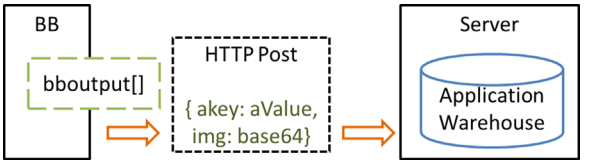

Fig. 12. Data exchange protocol between mobile client devices and the framework.

- Dynamically load required cascade style sheets to adapt the UI to the device.
- Access inputs through a special array.
- Store outputs on a special array.
- Define and access configuration variables.
- Implement a "start" method from which the building block execution will be launched.
- Invoke a "stop" method to relinquish execution to the next building block.

Building blocks are registered in the Building Block Warehouse, including description of inputs, outputs and configuration variables. The role of configuration variables is to enable at development-time customization of values to be used at run-time (e.g. a jigsaw implementing a gallery of images can accept keywords to define the type of images to be used, such as "wild life"). Support of special operations, such as iteration is also provided at the building block level. Such building blocks are statically added to the framework and have a special interpretation at run-time.

An application is an ordered set of building blocks registered in the Application Warehouse. The Application Warehouse is responsible to store all applications and its executing sessions; and manage the execution of applications. In the current implementation, an application has a description shared between users to allow everyone to benefit from existing applications and foster customization of existing applications. In addition, an application can also have sessions associated with it. A session is a unique identifier for an application in execution that links outputs from previous building blocks to the inputs of following building blocks, and manages which jigsaw to execute within the framework using related inputs. For this purpose, the framework previously stores outputs between the client device and the framework through HTTP Post requests. This communication is achieved through encapsulation of data in JSON structures. In case of binary data, such as the case of images, data is encoded in Base64. Base64 encoding was selected as Base64 encoding/decoding is natively supported in browsers making data ready to be used in building blocks through Javascript (see Fig. 12).

A contribution of the framework is the integration in a single environment of the ability to interact with *web services, native phone functions and smart things*. Smart things play an important role in connecting our everyday physical objects with Web services or even the creation of new and innovative services.

Established on the idea that providing the right tools based on understood Web protocols to end users will allow them to explore and create their own smart environments, Puzzle further extends this concept to other technologies that can merge the gap between what end users want and what technology can provide [6]. Indeed, the architecture is based on JSP, Javascript, XHTML, CSS, and implementation of building blocks relies on Javascript. This combination of technologies allows an embedded jQuery plugin to detect the platform where the environment is being executed and even adjust the environment toward that particular platform, resembling a responsive design approach. Consequently, during authoring or execution of an application, the UI is adapted to the current platform. Adaptation occurs through selection of the appropriate CSS classes and the plugin is also used to: (a) enable execution of an appropriate running sequence for that platform, and detect if the screen is in portrait or landscape mode. Therefore, information provided by the plugin can be further used within the execution to execute code according to the context. Access to *phone native functions* is performed through two methods. A first method relates to a module that answers to HTTP requests, performs a native function and returns a result. For this purpose and if parameters are required, a JSON structure is posted to the module with required parameters. In case of binary data, the required data is further encoded in Base64 encoding. This method benefits from the ability to be supported in current and older mobile platforms. A second method relates to the usage of Javascript to access native functions. The Android SDK and the iPhone SDK support this function.

To conclude this point, our developed *smart things* use open hardware such as Arduino or IEEE 802.15.4 boards. Arduino allows developers to access composition of sensors and actuators without a specific knowledge on electronics. Furthermore, communication with the Arduino board can rely on an Ethernet or WiFi connections as a first step, or also considering low consumption through the protocol IEEE 802.15.4. In Puzzle, the last option was implemented and HTTP GET requests from building blocks to sensors and actuators are performed through a Web service that forwards these requests through a serial port protocol using an IEEE 802.15.4 connection. The Web service accepts a JSON request with an operation, parameters and last parameter indicating whether the web service shall wait for an answer. Afterwards, the information is forwarded through the IEEE 802.15.4 connection to the hardware. The JSON request is translated into a String to reduce the amount of characters exchanged.

6. Usability evaluation

The goal of the framework is to allow end users without a programming background to create mobile applications opportunistically in touch-based devices. The framework contributes with: (a) a pictorial metaphor (the jigsaw) and interaction techniques that support the process of end-user service composition in mobile devices, (b) technologies to support creation of applications, able to access smart things, in heterogeneous mobile platforms, and (c) ability to create mobile applications that can be used through various interaction modalities. The evaluation aimed to assess how such framework is able to answer the research questions stated in the introduction.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

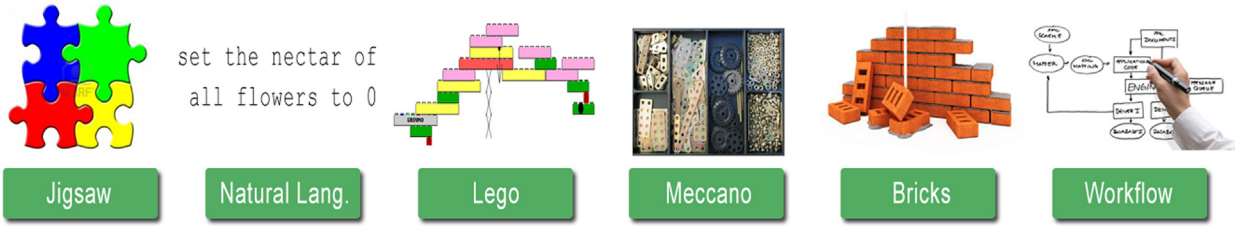

Fig. 13. Evaluated pictorial metaphors.

Through an iterative process, Puzzle has been developed using intermediate prototypes to evaluate and develop design solutions and to gradually build a shared understanding of end users' needs as well as their possible future practices [27]. The evaluation includes a *comparison between the UI in our preliminary work* [8] *and the current UI*. Through an empirical and iterative process, we first identified limitations of the UI, namely visibility of actions, illogical sequences of steps or a high number of actions required to reach a goal. The identified problems targeted: usage of side menus without hints as to their functions, the number of actions to execute an application, inability to easily identify the selected jigsaw category and change it, inability to provide help on objects and available actions in the authoring, a non-intuitive process to remove a jigsaw, inability to navigate within a canvas, or non-intuitive process to configure a jigsaw in authoring.

In order to address the identified limitations of our previous prototypes [8], we have designed a new set of interactions (see Fig. 4) and support them in the current UI. In particular, side menus have been removed and jigsaw selection converted into an always visible combo box showing the current category. Execution has been removed from the right menu and replaced by a button on the first jigsaw of the application to be executed. The delete operation has been modified from tapping on a jigsaw and further tapping on a menu to dragging the jigsaw into a trash can. Configuration of a jigsaw was initially performed by a tap on any part of the jigsaw, followed by a tap on a menu button and having to type in the value. This sequence is now a tap in a center jigsaw icon, a further tap on the configure button and configuration through a set of controls. Furthermore, navigation, help on actions and objects, and alternative and simpler menus have been included in the new UI.

Our last evaluation of the framework consisted of a pre-test questionnaire, a set of tasks to accomplish, and a post-test questionnaire. In the following sub-sections, we provide details on the evaluation participants, methodology, test setup and procedure.

6.1. Participants

Subjects were recruited through e-mail or personal invitation and were asked: (a) to have familiarity with mobile touch-based devices, and (b) not to have knowledge on using programming languages. Related to their job occupation, 9 volunteers were from the institute administration, 1 volunteer was an intern and the remaining volunteer was a university student. All subjects were novices with respect to usage and knowledge of the Puzzle framework (before this test, they had never used it). Volunteers received a small gadget for their participation.

Users were first introduced to the study and the observer/facilitator asked them a set of demographic and IT usage profile related questions collected in the pre-questionnaire, from where researchers calculated average (A) and standard deviation (SD) of the collected data. Eleven volunteers (7 females) aged 22–55 ($A=35.55$; $SD=10.72$) took part in the evaluation. All subjects had experience with touch-based devices and had moderate-to-high experience using web browsing, e-mail and social networks. Daily, volunteers also interact with computer-based devices; in terms of usage percentage per day they interacted ($A=67.55$, $SD=11.50$) with a computer or laptop, ($A=5.67$, $SD=4.01$) with a tablet, ($A=33.34$, $SD=12.50$) with a touch-based smart phone, or ($A=11.67$, $SD=2.89$) with another type of phone.

Most used applications were also analyzed and divided between desktop and mobile usage. In the mobile, 63.64% of users reported that, in average, they browse on the Web and check social networks (mainly, Facebook). In addition, 54.55% of the subjects reported that they also read e-mail in the mobile device or play a game. In the desktop, word processors, spreadsheets, mail readers and web browsing were the applications that all users reported to use. We also asked if subjects had previously considered or were interested in creating their own mobile applications. 27.27% of the subjects interviewed had considered to create their own applications. Remaining subjects had not considered that hypothesis due to: (a) lack of interest, (b) lack of knowledge on the tools to do it, or (c) the cognitive effort required to develop an application.

6.2. Methodology

In the pre-test questionnaire, we asked the users to assess different visual metaphors. For this purpose, we considered visual metaphors used on child construction kits (e.g. Jigsaw, Meccano, or Lego) and approaches used from other end-user languages (e.g. natural language [29], or workflow [30]) (see Fig. 13). A table was presented to subjects showing a visual description of the metaphor in one column and a five point Likert scale (not relevant=1, residual=2, neutral=3, relevant=4, very relevant=5). Subjects had to rate each visual metaphor for creating an application through a mobile device.

Following the pre-test questionnaire, the subjects had to accomplish three tasks while using a touch-based smart phone. The tasks were identified in order to assess specific

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

aspects in the composition of applications in a touch-based mobile device, specifically: (a) verify if it creates confusion to the user when a jigsaw is dragged into canvas and changes its shape by expanding, (b) check the effectiveness of the execution operation, (c) verify if users find access to smart things relevant, and (d) check which iteration visual representation to use.

Task 1 targeted the evaluation of points (a) and (b), task 2 focused on point (c) and task 3 focused on point (d). Tasks were as follows:

- Task 1—The subject had to post a text message or gallery image (from the phone or Web) on Facebook. The task was repeated both with the current UI and the UI used in the preliminary version.
- Task 2—The subject had to control a power outlet, check its consumption and interact with different modalities.
- Task 3—The subject re-used the application from task 2 and had to add some iterations to evaluate the visual representation of such construct.

During task performance, an observer was present to clarify any doubts subjects could have and took notes on the methods used by subjects to complete tasks. In task 1, subjects were able to use native phone functions (e.g. list text messages) and web services (e.g. post to Facebook). Usage of such functions was seamless to users as they were adding them through a jigsaw in their applications. During execution, the framework was also responsible for handling execution of such functions, thereby reducing the cognitive effort for users. In tasks 2 and 3, subjects were able to interact with smart things (e.g. a custom made Arduino system).

After task completion, a post-test questionnaire was filled in. A first question was whether it was important for subjects to have a channel that would allow them to communicate with developers and rank or request newer jigsaws to be added to the framework. Afterwards, we evaluated the granularity of the building blocks that users would like to manipulate in order to create their own applications. For example, an application for posting photos on Facebook, which is divided into two building blocks for selecting a contact and taking the picture has coarser granularity than one divided into authenticating the user, opening camera, taking picture, previewing the image, selecting a contact, and posting image.

In the granularity evaluation, a piece of textual programming language was first shown to subjects. The evaluation covered four levels of granularity (textual, visual GUI, medium and high) and each level was depicted with an image and a five point Likert scale (not relevant=1, residual=2, neutral=3, relevant=4, very relevant=5). Users had to rate each level. In order to not bias test results, all images were created from scratch and were not based on any existing development environment. From the finest level of granularity to the coarsest, it started with a sample of textual code. A first image represented a snippet of programming language code (textual). A second image represented a visual language similar to develop an application with MIT App Inventor. A third image represented a visual language where building blocks where at the level of perceived actions (e.g. posting a message to Facebook would be possible by combining a list of contacts, a text message or image to send, the type of sender and a send building block to integrate all outputs from previous blocks). At the coarsest level, only high level functions were listed (e.g. only a building block was able to receive input from a previous building block and post on Facebook).

Table 1
Sentences to evaluate Puzzle framework.

Number	Question
1	It is easy to use
2	It was easy to learn how to use it
3	It is easy to combine jigsaw Puzzle pieces
4	It is easy to understand the flow of the data
5	The icons and symbols are easy to understand
6	It is easy to add a jigsaw Puzzle piece
7	It is easy to delete a jigsaw Puzzle piece
8	It is easy to understand the result of the application
9	It is easy to get the result from the application that I want
10	It is easy to remember how to combine jigsaw Puzzle pieces to create an application
11	I have problems discovering what a jigsaw Puzzle piece is doing
12	It is easy to make errors or mistakes
13	It is easy to find mistakes in an application
14	I am satisfied with it
15	I would recommend it to a friend

Subsequently, the post-test questionnaire included a quantitative analysis to evaluate usability through six variables: learnability, efficiency, effectiveness, memorability, errors, and satisfaction. For that purpose, 15 sentences were presented to subjects and they were required to rate them in a five point Likert scale (strongly disagree=1, disagree=2, neutral=3, agree=4, strongly agree=5). Sentences 1–5 were designed to evaluate learnability, sentences 6–7 were designed to evaluate efficiency, sentences 8–9 were designed to evaluate effectiveness, sentence 10 was designed to evaluate memorability, sentences 11–13 were designed to evaluate errors, and sentences 14–15 were designed to evaluate the satisfaction. It should be noted that questions were positively and negatively constructed to check if the subject perceived its meaning. Formulated sentences are also related to the actions that can be used in Puzzle. Table 1 describes the list of sentences used.

Satisfaction levels while using the tool can further describe on whether a similar tool could be adopted in a product. Thus, we further evaluated satisfaction levels through the use of the Microsoft Reaction Card Method [31]. We used a subset list from the original 118 words with the following: *Expected, Friendly, Attractive, Professional, Sophisticated, Creative, Slow, Exciting, Difficult, Innovative, Consistent, Inconsistent, Engaging, Clear, Irrelevant, Time-consuming, Stimulating, Empowering, Confusing, Complex, Illogical, Flexible, Approachable, Rigid, Boring, Ambiguous, Faulty, Annoying, Fun, Predictable, Intuitive, Frustrating, Unconventional, Relevant,*

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

Table 2
Scenarios where Puzzle could be used.

Scenario
A set of sensors (LED, textile resistor, vibrator engine, temperature, scent dispenser, heart rate monitor, …) is available for you to use. Would you see yourself combining them and integrating their values within Puzzle possibly to interact with your mobile phone or social network. Possibly to reflect your emotions or that you have an incoming message/call (Could be used in Fashion industry to prototype garments)
A set of sensors/actuators (blood pressure monitor, textile resistor, vibrator engine, temperature, heart rate monitor, …) is available for a medical professional to select and use in a patient to monitor his/her health. This set of sensors could also be combined in the Puzzle authoring tool to use the values in different visualization methods or even to alert that professional of a threshold reached in some situation (could be used in medical industry)
A set of sensors/actuators (power plug/power management, surveillance/alarm system, window controller, temperature controller, …) is available for you to control your house. You could use all the controllers as blocks that you can integrate with other blocks in a Puzzle application. This would allow you to control your house ad reconfigure application for your own requirements. Furthermore, you would be able to monitor power consumption, water or gas. This set of sensors could also be combined in the Puzzle authoring tool to share your outcomes with friends in social networks. (Could be used in Smart homes)
A set of sensors/actuators (blood pressure monitor, textile resistor, vibrator engine, temperature, heart rate monitor …) is available for a coach to select and use in an athlete to monitor his/her performance. This set of sensors could also be combined in the Puzzle authoring tool to use the values in different visualization methods or to recommend new practice schemas for the athlete to improve his/her performance (could be used in sports)
You could integrate available services with phone features so that you could share and/or register your everyday life according to the different services that you are used to use. Share your Flickr images in Facebook or twit them, or even share a message from an email or SMS/MMS into a recipe service. (Could be used in Leisure)

Convenient, Compelling, Misleading, Effective, Clean, Entertaining, Efficient, Powerful, Easy to use, Hard to Use. The set of words was selected to understand perception of the overall experience and find out details about the experience.

In detail, this exercise intended to evaluate the functional aspects of the framework to ensure that the overall solution offers a compelling value and benefit to users. On top of a functional satisfaction evaluation, we also evaluated emotional satisfaction that stems from aesthetics, look, and feel. Functional satisfaction can be evaluated from the requirements of users to create an application, or from faults in the framework. Emotional satisfaction can be evaluated from the desirability and pleasure using the framework. Indeed, the pictorial metaphors can support the framework interaction design, but they can also elicit an emotional response from users. Understanding the functional responses is important to find optimal ways for a function to be completed. On the other hand, understanding and exploiting emotional responses can help design the framework to keep users interested in using it. Through analysis of words characterizing the subjects' experiences, we planned to evaluate both functional and emotional satisfaction [32].

Equally relevant for the development of the framework is to understand how users would foresee its usage in a real scenario. Through the usage of an open question, there is a chance to acquire unbiased answers but there is also a risk for the question to be left unanswered. Therefore, the approach was to first ask the subjects which application scenarios they foresee more suitable for the framework. If a subject was not able to provide a precise answer then a list of 5 scenarios was presented to the subject asking to rate them in a 5 point Likert scale (not relevant=1, residual=2, neutral=3, relevant=4, very relevant=5).

Table 2 provides the description of the presented scenarios.

Finally, an open question was asked to subjects for further comments on the framework.

6.3. Test setup

The test was carried out with a laptop computer, a touch-based smart phone, and a set of custom electronic components. The smart phone and the computer were connected through an IEEE 802.11b/g wireless router while the custom hardware was connected to the laptop through an IEEE 802.15.4 set of antennas. The laptop computer, Dell Vostro V130, was responsible to host the Puzzle framework and answer to HTTP requests from clients. In addition, communication between the laptop and the custom hardware was performed using a Serial Port protocol. The smart phone, Google Nexus S, had installed an Android application to access Puzzle.

Finally, a hardware component was built for the purpose of this evaluation. The hardware was able to control a light lamp and measure consumption of that light lamp. The main hardware components used were: 1 Arduino Duemilanove, 2 XBee Antennas, 1 SeeedStudio Relay Shield, 1 Light Lamp, 1 Current transducer, and 1 Power extension interfacing with the Relay Shield and the light lamp (see Fig. 2).

6.4. Procedure

Individual one hour interviews were conducted. The pre-test questionnaire was filled in by the researcher while asking background information questions (e.g. age or IT related experience) and afterwards, subjects filled in a form related to a number of questions. During completion of each task, a researcher acted as observer and introduced the framework to subjects describing its main functionalities. Afterwards, the observer acted only when users were stuck on some task and to indicate the next task. During this part of the evaluation, the researcher took notes on the usage of the tool. The post-test questionnaire was filled in by each subject and the researcher acted as a helper to clarify doubts.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

7. Results

In this section we summarize the results gathered through analysis of the data collected during the evaluation. Mainly, we based the evaluation on the filled questionnaires and notes from the observer. In addition, we also report on additional feedback/remarks/suggestions provided by users. During the discussion, we highlight the sub-goals that we are evaluating, even if there are questions and tasks that address multiple goals.

In order to answer the research question "How end-users users can be supported to create services and applications in a touch-based mobile device?", we first evaluated the visual metaphor used as a way to indicate how the environment conveys its functions. In a five point Likert scale, we aimed to understand how subjects would foresee an intuitive way to combine or join building blocks in order to create an application. The following boxplot graph includes: minimum and maximum values registered as lower and upper tickers, first and third quartile as the lower and upper limits of the box and median is represented in the middle of the box. First quartile, median and third quartile divide the population into the lower 25%, 50% and 75% of registered values, respectively.

The metaphors were ranked from top to bottom jigsaw ($M=3.91$), workflow ($M=3.64$), lego ($M=2.82$), natural language ($M=2.55$), bricks ($M=2.45$), and meccano ($M=2.36$).

Often, users based their rating on the approaches they use at work, or the toolkits they used in their childhood. The ability to easily connect functions with a jigsaw and lego, or familiarity with workflow methods are the basis for the higher ratings of these approaches. A natural language was found more abstract and thus requiring a more cognitive effort when compared with the other pictorial metaphors (see Fig. 14).

We have considered the limited screen space of a mobile device, how it restricts what the user is able to see and increases the interaction steps in order to select the right building block to use and still display essential information that conveys its functionalities. Therefore, the user has to open a combo box to select a category, select the right jigsaw and drag it to the canvas. Jigsaws in the scrollbar at the bottom only show the top input and output. When the user finishes dragging the jigsaw in the work area, it changes its appearance in order to show all its inputs and outputs. The method is depicted in Fig. 5.

Furthermore, we were interested to know if the changing representation of the jigsaw would confuse users. Six subjects reported that the changing representation was not a problem and it was easy to follow while combining jigsaws. The remaining users reported that they did not paid much attention at first but once they were queried by the observer to think loud about the issue, they found it confusing. After an explanation about the purpose of the changing representation of the jigsaw, stating that it would show more information about the inputs and outputs, subjects quickly understood the method.

Help on the jigsaw supported function could be obtained by clicking a jigsaw in the scrollbar or even in the canvas, but the possibility of also clicking on jigsaws was not clear. Fig. 15 shows the dialog box shown when a user clicks on a jigsaw in the scrollbar. Furthermore a jigsaw dragged to the canvas also provides access to help (Fig. 15) on its behavior and may (or may not) allow the user to configure it. In case the jigsaw supports configuration, a click event popups up an additional dialog asking the users whether they want to get help on the jigsaw or configure it.

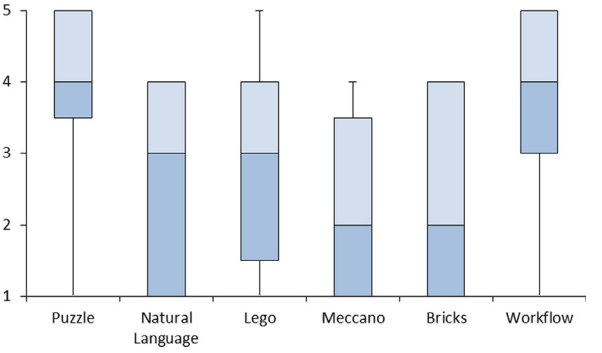

Fig. 14. Pictorial metaphors evaluation.

Based on suggestions from our preliminary work, the visual environment used in the test was already improved to address some user requirements and consider newer interaction methods. Changes range from: (a) changing sliding menus to a single menu on the left, (b) adding a combo box to allow for category selection and show the selected category, (c) including canvas navigation, or (d) exploiting the drag and drop interaction technique. Fig. 16 shows the preliminary authoring tool UI and Fig. 17 shows the current UI.

Seven subjects preferred the current UI. The reasons involved personal satisfaction since they felt more desirability and pleasure using the current UI and functional satisfaction as the functions were clearer and easier to use.

We also introduced ways to allow users to partially test and execute an application. In our preliminary work, one user suggested to provide hints on where the execution would start. In such evaluation, execution was triggered through access to a sliding menu in the right side of the screen (see Fig. 18). As consequence, we added a button to the first jigsaw to execute in order to ease its identification and provide a visible option to immediately execute or partially test an application (see Fig. 19).

All subjects preferred the approach depicted in Fig. 18. This could be due to two reasons: whether the proposed button was not clear on its function or users preferred the initial method due to legacy reasons using desktop applications. Afterwards, the observer explained the functionality of the button to subjects and they started to use it without problems. Nevertheless, they were suggesting that maintaining both approaches would ease novice users to get familiar with the button approach.

The research question "What technologies can be used to support heterogeneous mobile devices and smart things?" was addressed through the implementation of Puzzle and how the modules are interconnected to support the creation and execution of mobile applications and services. To envision requirements for realistic usage of

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

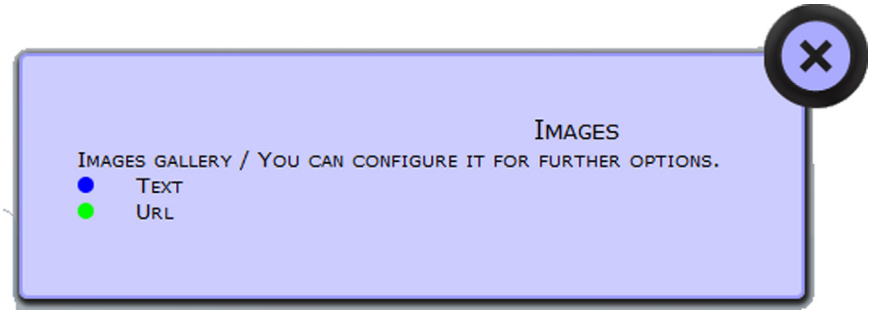

Fig. 15. Jigsaw help dialog.

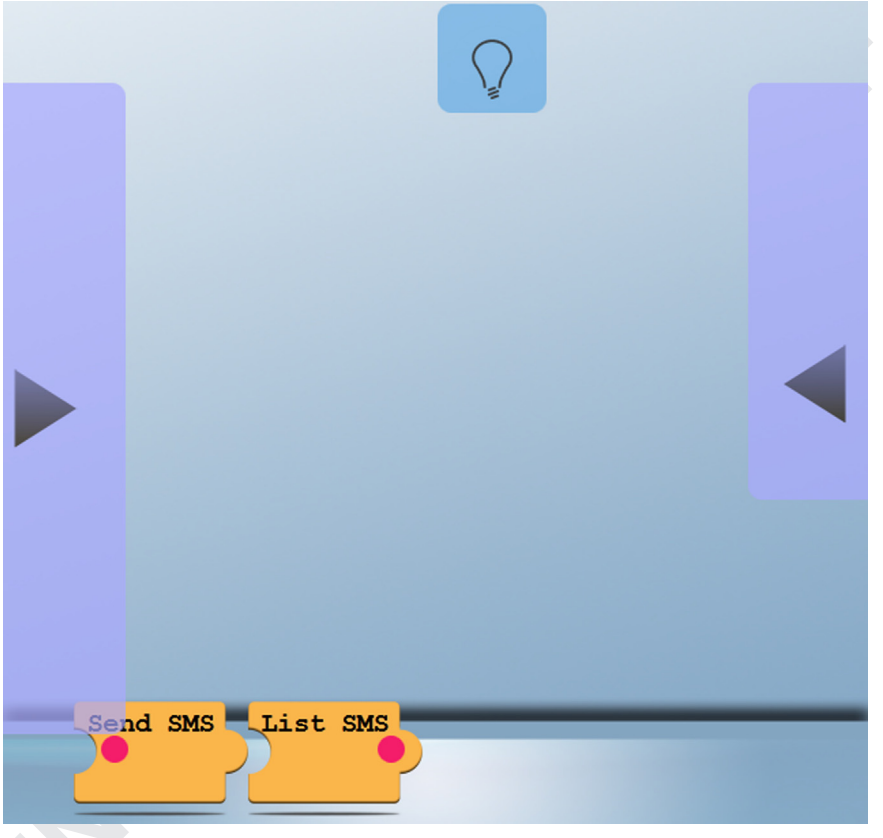

Fig. 16. Previous authoring tool UI.

smart things and smart spaces, customization is required to adjust the application to user's needs as well as expectations in hardware and software. Due to the large and varying individual needs for these applications, the traditional software paradigm, where users buy an application developed by a professional and limited to the embedded functions, will not scale [6]. This framework also makes a contribution by allowing the users to adjust their applications to be used through a selected modality and combine these modalities with their expectations on hardware and software through access to a set of web services, native phone functions and smart things. In an example, a user is able to select an interaction modality (e.g. touch screen, voice and physical button) and select with which smart thing to interact with. The framework is flexible to include and be extended to different hardware components. To exemplify its usage, we have used Arduino and IEEE 802.15.4 boards. Usage of such boards reduces learning process to create custom electronics and we argue that it will motivate users to further explore potentialities of our framework and create hardware components that would fit their needs.

An Arduino board was used to prototype the integration of IEEE 802.15.4 to control home appliances and allow users to create applications that would interact with such hardware prototype. For this task and following the previous example, we created a set of building blocks to control the Arduino and additional building blocks that could interact with them through different modalities. All users valued the approach of being able to control

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

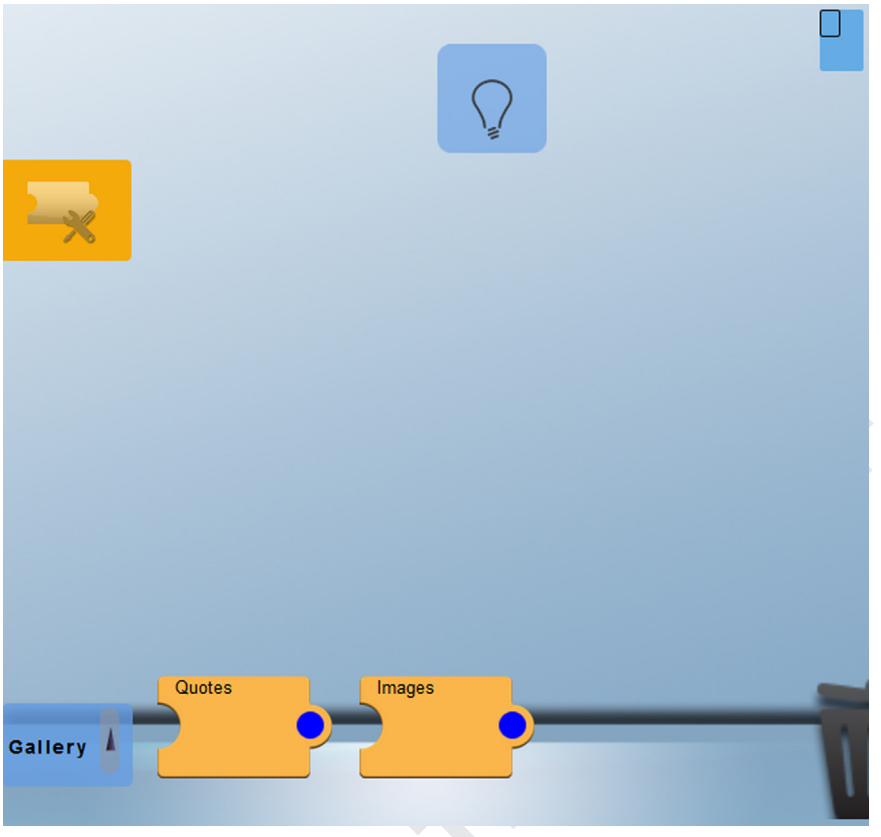

Fig. 17. Current authoring tool UI.

Arduino through a different interaction modality. The possibility to remotely control smart things in the house (or even outside the house) with different modalities was interesting for them. All users were pleased that it was easy to change the modality while keeping the same functionality of the application. Changing the modality was enabled through the exchange of the building blocks that were related to the interaction. Moreover, the fact that they were seeing the result of their actions reflected on a real scenario was motivating for all subjects.

In order to further support the research question "How end-users users can be supported to create services and applications in a touch-based mobile device?", we included and evaluated the use of iteration within the framework. We were interested in how users prefer the representation of iteration. The user actions required to support iteration were outlined by: (a) combining jigsaws for the required behavior, (b) select a jigsaw supporting the iteration function, (c) drag the iteration jigsaw to the top of the jigsaws comprising the selected behavior, (d) the iteration jigsaw would redraw itself to adjust to the connected jigsaws. Two implemented pictorial representations were shown to subjects, see Figs. 8 and 9.

Nine subjects selected the approach in Fig. 9, one user mentioned that it was important to have the feature supported and he would adjust to the supported representation and 1 user selected the approach in Fig. 8. Main reasons to support the approach in Fig. 8 were due to a less intrusive representation. The approach in Fig. 8 was cumbersome for 9 subjects and the approach in Fig. 9

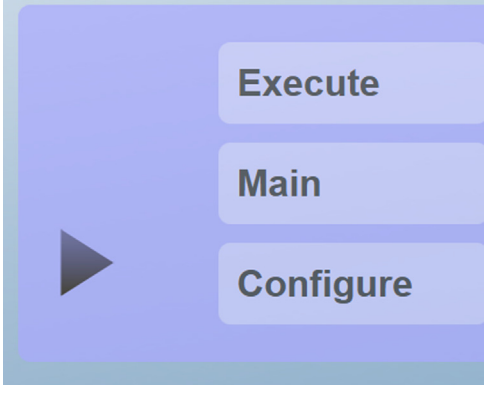

Fig. 18. Previous execution method.

was more desirable and pleasurable to use, thus supported by an emotional satisfaction since functional satisfaction was similar from both approaches.

In the evaluation and motivated by the fact that the framework would benefit from experts contributing with additional functions, we also considered ways for end users to communicate their needs to platform developers and enable the framework to be extended by other expert developers. A feature that could enhance acceptance, support and extension of such framework would be a communication channel between end user and expert

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

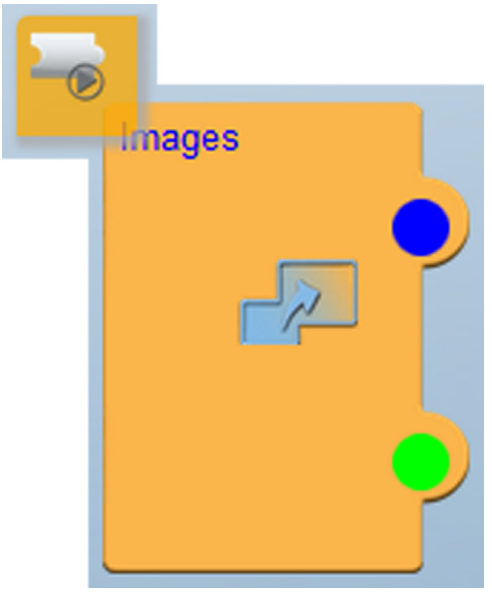

Fig. 19. Current execution method.

developers, namely developers of additional jigsaws. The argument used to describe such communication channel was that it would enable them to express their requirements for additional building blocks and at the same time provide developers with an overview of the required jigsaws to include and support developers' decision on which jigsaws to implement. Benefits could be mutual as developers would be able to get a good overview of users' requirements and users would be able to influence developments to fit their needs. Support for such channel of communication was highlighted as positive by all subjects.

An answer to the research question "What level of programming granularity would be suitable for the development of end-user mobile applications?" is of key importance to understand how end-users envision mobile EUD. After introducing users to our end user environment, we were also interested at which granularity level they were willing to create an application. Four options discussed in the methodology section were then presented to subjects.

All subjects did not have knowledge on programming languages and therefore were not very keen to use a programing language ($M=1.36$). For the remaining levels, the average was the same ($M=3$). However, the subject's evaluation distribution is different between a Visual GUI (lower granularity of Visual End User Development), Medium and High (medium and higher granularity of Visual End User Development). The Visual GUI and Medium granularity levels have a higher dispersion of values between the lower 25% of the subjects, and a median of 4. The High granularity level has an equally dispersed distribution and a Median of 3 (see Fig. 20). A solution could be to support the ability for users to create and modify an application at different levels. Research on this topic was carried out [15], however a more generic solution on how to support modification of applications created at lower granularity levels on higher levels is required. Mainly, flexibility provided in lower levels of granularity

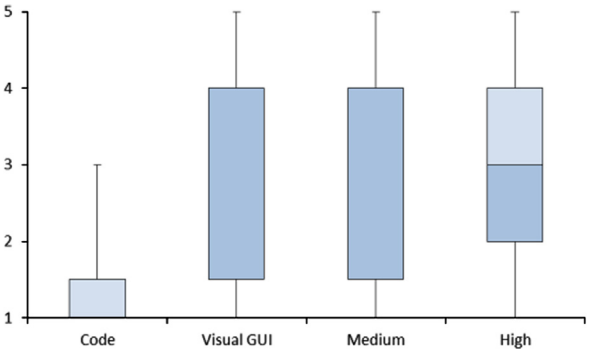

Fig. 20. Evaluation of end user development granularity level.

compromise the ability of such application to be edited in higher granularity levels.

In general, we evaluated the current UI and operations asking subjects a set of sentences targeting: learnability, efficiency, effectiveness, memorability, errors and satisfaction. Sentences for memorability and errors were negatively written to remove bias on answers.

The results showed that the interface is easy to learn, moderately efficient, effective, easy to memorize how operations work, reduces the number of errors a user can make and the framework is functionally and emotionally satisfactory (see Fig. 21).

Satisfaction was further evaluated through the use of Microsoft Reaction Cards. Within the current evaluation, it was important to evaluate whether functional satisfaction or emotional satisfaction was playing a higher role in subjects experiences. In the analysis of results, we analyzed each selected word and defined whether its meaning is related to a functional or emotional satisfaction factor. Fig. 22 shows a graph with occurrences of the words that got more than 2 occurrences.

"Easy to use" was the most selected keyword from subjects, followed by "Efficient". Both keywords relate to a functional satisfaction and show that subjects were pleased with the functions provided by the framework. "Stimulating" and "Intuitive" relate to emotional satisfaction and, therefore, provide insights on how the aesthetics, look, and feel could invite subjects to further explore the framework. Keywords such as "Professional", or "Clear" provide hints that the framework could target domain experts to help on their tasks, and still is complex for some subjects. On the other hand, "Friendly" or "Innovative" express that subjects would be able to start using it in the current level of complexity. These somehow controversial results can be explained by the experiences each subject has toward usage of technology. Nevertheless, further research is required to make the framework desirable to those subjects.

The evaluation also considered the scenarios where subjects would foresee the framework to be used. First, subjects were queried with an open question on which scenarios they would foresee the framework to be used. Answers started by subjects highlighting the home automation scenario or were blank. Since home automation could be a scenario biased by the tasks subjects enrolled during the evaluation, we presented further scenarios that

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

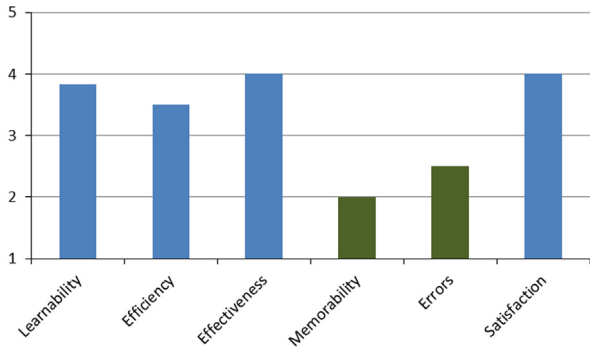

Fig. 21. Learnability, efficiency, effectiveness, memorability, errors and satisfaction levels using the framework.

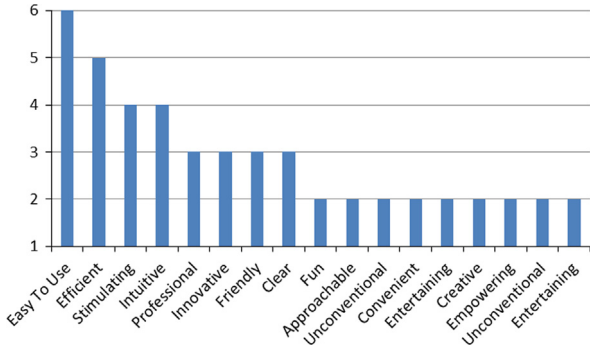

Fig. 22. Evaluation for microsoft react cards.

they could rate with a five point Likert scale. A medical related scenario was the scenario that scored the most relevant. Users were pleased that the framework could be able to allow medical staff to combine sensors to monitor patients without requiring the assistance of a technical person. However, relevance of the usage of Puzzle on such scenario could also be biased by the relevance of health to subjects. Afterwards, home automation was the scenario that gathered the interest of evaluated subjects (see Fig. 23).

8. Discussion

The results of this evaluation suggest that subjects were interested in the benefits of the Puzzle framework and recognized its potentialities. Mainly, subjects recognized the benefits of being able to create applications that could interact with web services, phone functions and smart things, focusing on the home automation scenario.

Puzzle considers a mobile framework, since usage of mobile platforms are increasing and it creates new opportunities for users to create applications when required. The main contributions of this research range from: (a) metaphors and interaction techniques to support a mobile EUD environment, (b) framework based on widely used web technologies and end-user hardware prototyping tools to foster its broad application, or (c) Analysis of user's mental process in the development of mobile applications.

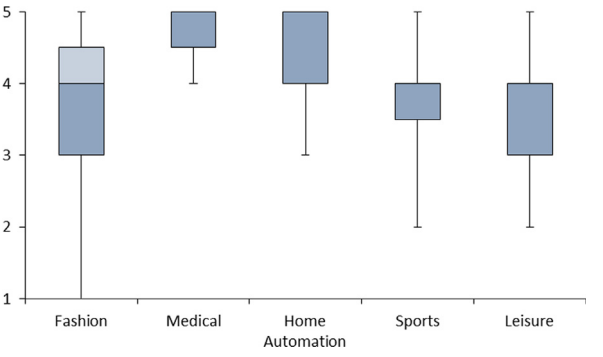

Fig. 23. Scenarios were to apply Puzzle.

8.1. Metaphors and interaction techniques to support a mobile EUD environment

The corpus of knowledge related to mobile EUD is still in its infancy and Puzzle contributes with knowledge on metaphors and interaction techniques that users can identify and use to create a mobile application. Our evaluation showed that a jigsaw is a familiar metaphor to use and subjects were keen to use it. However, further work may be required to go beyond paper prototypes as we have used in our evaluation to evaluate alternative metaphors. Our evaluation is limited to the use of images of the metaphors lacking possible interaction effects, namely with a workflow metaphor. The workflow metaphor could that could be adequate for scenarios were users share their attention between the tasks they are working on and the mobile screen.

The jigsaw metaphor was the preferred metaphor and it was further extended with a dynamic behavior to accommodate different inputs and outputs. The behavior is discussed in Fig. 5 and our results show that users were able to use it and understand its meaning. Users found it useful in order to have a more clear view on how the information flows. Limitations were still identified and further hints should be provided to users on how they can get further information about a jigsaw and its operations, inputs and outputs.

Two ways of representing iteration were also evaluated. The preferred representation is not completely consistent with the jigsaw metaphor. However, such representation allows a more compact representation and therefore was preferred by subjects in the evaluation. As a general comment, subjects found the method less intrusive and more adequate for mobile UIs.

In general, the new UI layout for the authoring tool was preferred by the subjects in the evaluation. Reasons relate to functional and emotional satisfaction using the new version. A majority of subjects found the current UI to work better (functional satisfaction) and were more willing to use it (emotional satisfaction). The improvement of the UI relate to the usage of direct representation and interaction techniques that avoid a change of context. In concrete to execute an application, users did not have to go to a menu. Execution was presented in front of them. Representation of the jigsaw metaphor follows a similar approach which allows users to understand what can be

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

connected to each jigsaw. Iteration also benefits from immediately state its purpose without requiring the user to resort to dialogs and change context to add iteration to her application.

8.2. Framework based on widely used web technologies and end-user hardware prototyping tools to foster its expansion

A key feature of the framework is the possibility to allow integration of smart things and use of different modalities in the created applications. Usefulness of this functionality was immediately identified by users and it was easy for them to wonder through possible scenarios were they could use the framework in their daily lives. Use of different interaction modalities in the resulting applications was also relevant for the subjects in the evaluation. This would enable subjects to control devices with different modalities depending on the task they were executing. Beyond the usefulness for users, the ability to build the framework based on web-technologies shows its flexibility and ability to expand with an increasing corpus of APIs and other frameworks specifically targeting web technologies that also go beyond its usage on a single platform.

8.3. Analysis of user's mental process in the development of mobile applications

Our research also analyses how users are willing to create mobile applications on mobile devices and provides hints for future research in the area of mobile EUD. Our evaluation showed that users may seek to use more complex approaches in the mobile device as they get more familiar with the tool. Our prototype focused on the usage of a higher level of granularity at the cost of hiding implementation details that could be used for further customization. Through this approach, users were easily able to create their applications without programming language skills. However, the framework can be extended to allow access to a more detailed view, mainly focusing on pictorial metaphors that would allow access to further details of the implementation.

In the pre-test questionnaire, a majority of subjects stated that they were not considering developing mobile application. Nevertheless, after using Puzzle they realized the usefulness of the approach and high satisfaction levels were achieved with its usage. Thus, we can argue that research, investment and improvement of these tools can bring benefits for many stakeholders interested in mobile applications and the Internet of Things. Indeed, in the same way social tools invite end users to contribute with content through good satisfaction levels; end user development frameworks can allow users to create applications that are adapted to their requirements and would enable developers to evaluate users' requirements and provide further tools that would empower users to keep exploring new and innovative applications.

9. Conclusions and future work

Puzzle targets users without programming skills desiring to start developing mobile applications on their touch-based phones and then executing such applications still in mobile devices. From our results, Puzzle has addressed the initial goals and enables users to easily interact with web services, phone functions and smart things including the ability to interact through touch and voice. In addition, a high programming granularity level to create and modify applications has allowed users to easily explore the framework and create their own applications.

One important aim of the project is to contribute to the research on mobile end user development in order to stimulate users to go beyond consuming content and applications to start creating their own applications adjusted to their requirements and possibly change such applications when requirements change. The proposed environment allows users to explore possible combinations of technologies and, consequently, contribute to a new and innovative view of the possible applications, including those for internet of things.

Future work will include further exploration of metaphors, interaction techniques and technologies to integrate and use in the framework and allow end users to explore diverse scenarios, e.g. the home automation scenario. Inclusion of support for additional programming granularities levels to create and modify applications is another research topic considered. A further aspect to be considered is the use of different modalities and context-aware features to support development and modification of applications as well as supporting different pictorial metaphors according to the context and/or preferences of the user.

References

[1] M. Eisenberg, G. Fischer, Programmable design environments: integrating end-user programming with domain-oriented assistance, human factors in computing systems, in: CHI'94 Conference Proceedings, Boston, MA, (1994), pp. 431–437.
[2] T. Zhang, B. Brugge, Empowering the user to build smart home applications, toward a human-friendly assistive environment: ICOST'2004, in: Second International Conference on Smart Home and Health Telematics, 170, IOS Press, 2004.
[3] G. Fischer, E. Giaccardi, Meta-design: a framework for the future of end-user development end user development, in: H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, Vol. 9 (2006), pp. 427–457.
[4] H. Lieberman, F. Paternò, M. Klann, V. Wulf, End-user development: an emerging paradigm, in: H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, Vol. 9 (2006), pp. 1–8.
[5] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, B. Steece, Software Cost Estimation with COCOMO II, Prentice, Upper Saddle River, 2000.
[6] S. Holloway, 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). (2010) ACM, New York, NY, USA, pp. 167–172.
[7] J. Danado, F. Paternò, A prototype for EUD in touch-based mobile devices, in: Proceedings of VL-HCC 2012, October 2012, IEEE, Innsbruck, Austria pp. 83–86.
[8] J. Danado, F. Paternò, Puzzle: a visual-based environment for end user development in touch-based mobile phones, in: Proceedings of HCSE 2012, October 2012, Springer, Toulouse, France, pp. 199–216.
[9] G. Ghiani, F. Paternò, L.D.. Spano, Cicero designer: an environment for end-user development of multi-device museum guides, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.), IS-EUD 2009. LNCS, vol. 5435, Springer, Heidelberg, 2009, pp. 265–274.
[10] R. Hull, B. Clayton, T. Melamed, Rapid authoring of mediascapes, in: N. Davies, E.D. Mynatt, I. Siio (Eds.), UbiComp 2004. LNCS, vol. 3205, Springer, Heidelberg, 2004, pp. 125–142.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005

[11] A. Celentano, M. Maurizio, An end-user oriented building pattern for interactive art guides, in: M. Costabile, Y. Dittrich, F. Fischer, A. Piccinno (Eds.), IS-EUD 2011. LNCS, vol. 6654, Springer, Heidelberg, 2011, pp. 187–202.
[12] V. Realinho, A.E. Dias, T. Romão, Testing the usability of a platform for rapid development of mobile context-aware applications, in: P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, M. Winckler (Eds.), INTERACT 2011, Part III. LNCS, vol. 6948, Springer, Heidelberg, 2011, pp. 521–536.
[13] A. Namoun, T. Nestler, A. De Angeli Service composition for non-programmers: prospects, problems, and design recommendations, web services (ECOWS), in: 2010 IEEE Eighth European Conference on, vol., no., , 1–3 Dec. (2010), pp. 123–130.
[14] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, Scratch: programming for all 52 (2009) 60–67. (in:)Commun. ACM 52 (2009) 60–67.
[15] U. Tuomela, I. Kansala, J. Hakkila, J. Mantyjarvi, Context-studio: tool for personalizing context-aware applications in mobile terminals, in: Proceedings of 2003 Australasian Computer Human Interaction Conference, OzCHI 2003, (2003), pp. 64–73.
[16] J. Danado, M. Davies, P. Ricca, A. Fensel, An authoring tool for user generated mobile services, in: A.J. Berre, A. Gómez-Pérez, K. Tutschku, D. Fensel (Eds.), FIS 2010. LNCS, vol. 6369, Springer, Heidelberg, 2010, pp. 118–127.
[17] S. Cuccurullo, R. Francese, M. Risi, G. Tortora, MicroApps development on mobile phones, in: M. Costabile, Y. Dittrich, G. Fischer, A. Piccinno (Eds.), IS-EUD 2011. LNCS, vol. 6654, Springer, Heidelberg, 2011, pp. 289–294.
[18] J. Seifert, B. Pfleging, E. Bahamóndez, M. Hermes, E. Rukzio, A. Schmidt, 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 2011), ACM, New York, NY (2011), pp. 109–112.
[19] S. Holloway, 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 2010), ACM, NY (2010), pp. 167–172.
[20] B. Athreya, F. Bahmani, A. Diede, C. Scaffidi, End-user programmers on the loose: a study of programming on the phone for the phone, in: IEEE Symposium on Visual Languages and Human-Centric Computing, (2012), pp. 75–82.
[21] Atooma, ⟨http://www.atooma.com/⟩, (Accessed May, 2013).
[22] Tasker, ⟨http://tasker.dinglisch.net/⟩, (Accessed May, 2013).
[23] Locale, ⟨http://www.twofortyfouram.com⟩, (Accessed May, 2013).
[24] IEEE 802.15 WPAN™ Task Group 4 (TG4), ⟨http://www.ieee802.org/15/pub/TG4.html⟩, (accessed May, 2013).
[25] I.P. Cvijikj, F. Michahelles, The toolkit approach for end-user participation in the internet of things, in: D. Uckelmann, M. Harrison, F. Michahelles (Eds.), Architecting the Internet of Things, Springer, Berlin Heidel, 2011, pp. 65–93.
[26] A.F. Blackwell, Pictorial representation and metaphor in visual language design, J. Visual Lang. Comput. 12 (3) (2001) 223–252.
[27] G. Fischer, E. Giaccardi, Y. Ye, A.G. Sutcliffe, N. Mehandjiev, Meta-design: a manifesto for end-user development, Commun. ACM 47 (9) (2004) 33–37.
[28] K. Hinckley, J. Pierce, M. Sinclair, E. Horvitz, Sensing techniques for mobile interaction, in: Proceedings of the 13th annual ACM symposium on User interface software and technology (UIST '00). ACM, New York, NY, USA, (2000), pp. 91–100.
[29] B.A. Myers, J.F. Pane, A. Ko, Natural programming languages and environments, Commun. ACM 47 (9) (2004) 47–52.
[30] Pure Data, ⟨http://puredata.info⟩, (accessed May, 2013).
[31] Microsoft Reaction Card Method, ⟨http://uxmatters.com/mt/archives/2010/02/rapid-desirability-testing-a-case-study.php⟩, (accessed May, 2013).
[32] A.F. Blackwell, See what you need: helping end-users to build abstractions, J. Visual Lang. Comput. 12 (5) 2001, pp. 475–499.

UNCORRECTED PROOF

Please cite this article as: J. Danado, F. Paternò, Puzzle: A mobile application development environment using a jigsaw metaphor, Journal of Visual Languages and Computing (2014), http://dx.doi.org/10.1016/j.jvlc.2014.03.005