Design and Implementation: the End User Development Ecosystem for Cross-platform Mobile Applications

Zhongyi Zhai, Bo Cheng, Zhaoning Wang, Xuan Liu, Meng Niu, Junliang Chen
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
[zhaizhongyi@126.com, chengbo@bupt.edu.cn]

ABSTRACT
In this paper, we present an ecosystem for mobile application development by end-users. An advantage of this ecosystem is that the graphical user interface (GUI), as well as the application logic, can both be developed in a rapid and simple way. This ecosystem is mainly implemented through development and integration of two sub-systems, namely *EasyApp* and *LSCE*. *EasyApp* is responsible for developing the mobile app with the compatibility of multiple mobile platforms, while *LSCE* is in charge of creating the service process that can be invoked by mobile app directly. A case study is presented to illustrate the development process using this ecosystem.

Categories and Subject Descriptors
D.1.7[Programming Techniques]:visual programming; D2.2[Software Engineering]: design tools and techniques—*user interfaces*;

Keywords
Mobile application, end-user development, service mashup

1. INTRODUCTION
Mobile terminals have become the primary application and communication platform for people's daily life. With the rise of *App Store*, more and more mobile users (*called end-users in the paper*) can download and install the applications, such as *online meal ordering*, to facilitate their life. However, in contrast to the widely available of mobile terminals, the mobile applications provided by *App Store* are insufficient for the user requirements. To address this problem, the end-user development (EUD) has been introduced to create mobile applications according to user's own requirement with the aided tools.

At present, there have some development tools that focus on the creation of mobile applications. For example, the *MIT App Inventor* [1] is a drag-and-drop development tool for designing and building fully functional mobile apps for Android. In this tool, the mobile development is divided into two parts: the GUI, which can be developed using the *Designer*, and the *application logic*, which can be developed using the *Block Editor*. The *Designer* and *Block Editor* both provide a visual development interface, but the *Block Editor* can only be used by end-users equipped with programming skills. *MicroApp*[2] is another development tool to enable end-users to graphically develop the mobile applications directly on the mobile phone. In *MicroApp*, the mobile development is completed by composing services following a data-flow approach, and the GUI can be created automatically by integrating all of service interfaces in sequence. The applications developed by *MicroApp* can only run on mobile Android devices. Existing tools, such as *MobiMash* and *AppCan*, are similar to above two tools. There might be some defects for these tools: (*i*) the mobile development is just for specific platform; (*ii*) the application logic is executed on the mobile terminal, which is inappropriate for practical applications; (*iii*) some tasks in the mobile development are beyond the capability of end-users.

To address above issues, we design and implement a EUD ecosystem for cross-platform mobile applications, including *EasyApp* and *LSCE*. *EasyApp* is a mobile development environment, which is developed based on OSGI framework and Web techniques, to support multiple mobile platforms. *EasyApp* is responsible for developing the mobile app as the GUI of the whole application. *LSCE* is a service mashup environment in a drag-and-drop mode, in which service processes (i.e application logic) can be developed following a dataflow approach. The service process developed by *LSCE* would be deployed in the execution environment, and its open interfaces would be published for the corresponding mobile app.

2. ECOSYSTEM AND IMPLEMENTATION
The overview of the ecosystem is shown in Fig.1. *EasyApp* allows end-users to design the GUI through a WYSIWYG (what you see is what you get) editor. Once a GUI is completely developed, the end-user can acquire expected installation package of mobile app (such as *.apk*, *.ipa*) by a one-click way. When designing a GUI, the end-user may need to bind a composite functionality to an element in it. In such case, the end-user should also develop and deploy the composite functionality using *LSCE*, and then, publish its interface (such as RESTful) for the corresponding app.

2.1 EasyApp
Fig.1(a) shows the development interface of *EasyApp*. On the left of this interface is a widget library, including *UI elements*, *mobile device services* (*SMS*, *Bluetooth*) and *user interaction services* (*Google Map*). Utilizing the library, mobile app can be developed on the visual mobile phone interface, considering the page layout and page forwarding. On the left of the interface is the *setting*, including *properties of page*, *page event handing* and *remote service binding*. The remote service can be an atomic service, and can also be a composite service developed by *LSCE*.

The *EasyApp* was developed based on OSGI framework and mobile web techniques. Also, the library adopted by *EasyApp* is compatible with most *WebKit browsers*, so that the mobile app can be directly executed via the mobile browser. Furthermore, the *EasyApp* introduces the *PhoneGap plug-in*, so that it can call device functionalities through JS API. In next work, we aim to design a widget recommendation approach for GUI development.

2.2 LSCE
The LSCE is a development and execution environment for service mashup. It consists of service creation environment (SCE) and *service execution environment* (*SEE*). Fig.1(b) shows the *SCE*,

WWW 2016 Companion, April 11-15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04
http://dx.doi.org/10.1145/2872518.2889366

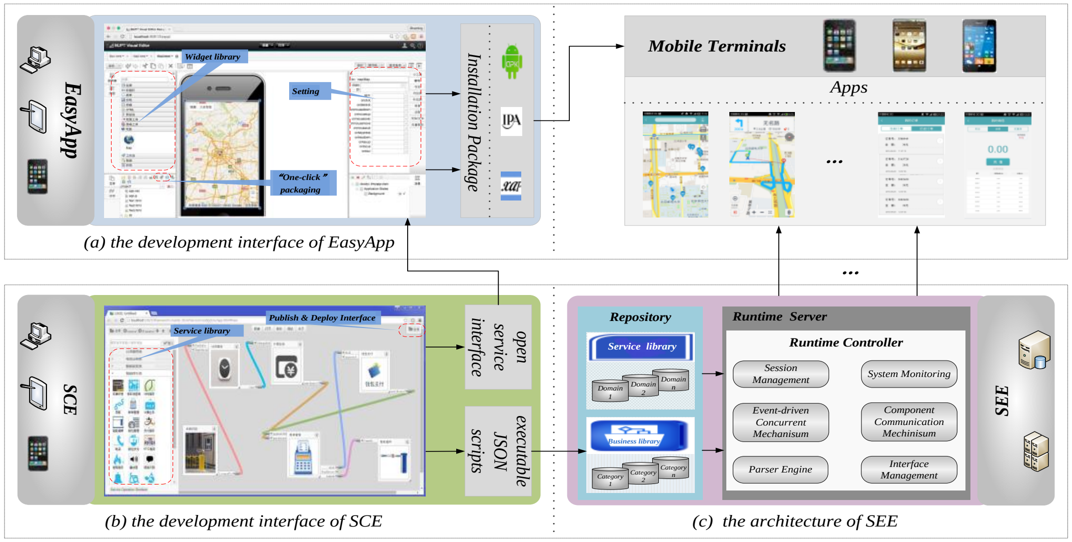

Fig.1. The overview of the development ecosystem for mobile applications.

including *service library*, *workspace* and *publish & deploy interface*. On the left of *SCE* is the *service library*, which is divided into multiple *communities* according to the domain diversity, such as *transportation* and *smart home*. By utilizing *SCE*, the end-user can drag and drop services into a dataflow graph, i.e. application logic. To facilitate the execution of application logic, a *JSON-based notation* is presented corresponding to the dataflow graph. Once a dataflow graph of application logic is completely developed, its interface would be published to the app, and its JSON script would be deployed into *SEE*.

Fig.1(c) shows the architecture of *SEE*. It takes charge of the management of execution engine, and is developed based on Node.js platform. In which, we have implemented the execution engine to be able to parse the JSON script of dataflow graph. Thus, the application logic developed by *SCE* can be parsed and executed directly in *SEE*, without additional transformation.

In the following work, we aim to design an incomplete information-oriented service mashup approach using AI planning techniques, to further facilitate end-user development.

3. CASE STUDY AND EVALUTAION

This section aims to demonstrate the mobile development of *Smart Parking Lots* (*SPL*), following the development approach of Fig.1. For the *SPL* application, we predefined three functionalities: (*i*) users can search nearby parking lots and navigate to the selected parking lot; (*ii*) the application should provide an interface for users to check their order information; (*iii*) the application should provide an e-wallet functionality, so as to pay for the bill automatically.

According to the *SPL* requirements and the development paradigm of this ecosystem, the development of *SPL* was divided into two parts:

(1) Developing user interfaces of *SPL* with *EasyApp*. The interfaces involve *parking lot search and navigation interfaces (SNI)*, *order inquiry interface* (*OII*) and *e-wallet recharge and management (ERMI)*. Furthermore, the page forwarding related to the search and navigation interfaces had also been designed in this part.

(2) Developing application logics for *SPL* with *LSCE*. For the *SNI*, it involves a business of parking lot information in real-time. Similarly, the *OII* involves a business of license plate recognition and charging, and the *ERMI* need to invoke a recharging and management business. All these three businesses were developed using *LSCE*.

In order to understand the development of SPL more clearly, the design and implementation materials are available online (http://114.215.159.178/resources/).

A questionnaire has been conducted on 20 recruits of *Beijing University of Posts and Telecommunications* to compare the effectiveness of the *EUD ecosystem* with the *MIT App Inventor*. The result (Fig.2) shows that the *EUD ecosystem* is more usable than the *MIT App Inventor* on the overall development and design of application logic.

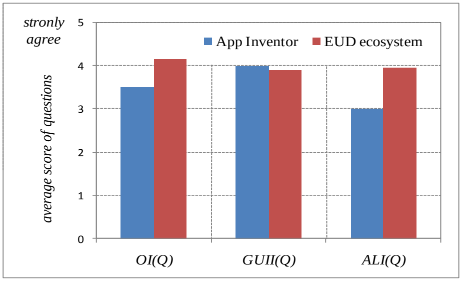

Fig.2 The Reaction of recruits to the App Inventor and the EUD ecosystem.

(Note: The *OI* represents the overall impression, *GUII* represents the impression of *GUI* development, *ALI* represents the impression of application logic development. The questions related to the experiment is shown online(http://114.215.159.178/resources/))

4. ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China (61132001, 61262008), National Key Technology Research and Development Program of China "Research on the mobile community cultural service aggregation supporting technology" (2012BAH94F02).

5. REFERENCES

[1] Shaileen Crawford Pokress, Jose Juan Dominguez Veiga. MIT App Inventor：Enabling personal mobile computing. *PROMOTO'13*, October 26, 2013, Indianapolis, IN, USA.

[2] Rita Francese, Michele Risi, Genoveffa Tortora and Maurizio Tucci. Visual Mobile Computing for Mobile End-Users. *IEEE Transactions on Mobile Computing*, 2015.