2014 The 2nd International Conference on Information Technology and Science (ICITS 2014)
IPCSIT vol. 59 (2014) © (2014) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2014.V59.29

Socialisable Mobile App Builder for Non-Programmers

Karl Kwan [1*+], Hin-Wah Yip [1] and Ying-Ming Ng [1]

[1]School of Digital Media and Infocomm Technology, Singapore Polytechnic

*This paper is supported by Ministry of Education Singapore

Abstract. The existence of Pareto effect on mobile applications download [1] may imply that 80% or more of the applications available in the online stores are not meeting end-users' needs. This situation may get worse as more developers are needed for mobile enterprise solutions [2]. The solution proposed in this paper, the Socialisable Mobile App Builder, aims to provide an alternative approach in creating mobile applications for end-users.

This product not only allows non-programmers to create personalized apps that satisfy individual needs but also incorporates a sharing framework, hence "socialisable", which allows the creators to easily notify his contacts and share the apps with them. The core of this product is an application server that stores the personalization and distribution functions. The impending success of this product may then shift the current mobile app usage paradigm from a game-centric one to a social- and productive-centric one [3].

Keywords: Mobile Computing, Personalized Mobile Apps, Visual Programming, Social Networking

1. Introduction

The rise of the digital economy coupled with the prevalent use of smart phones has seen the sales of the latter soaring to new heights. It is estimated that one billion smart phones have shipped since the beginning of 2013 [3] while mobile apps users will jump from 1.2 billion in 2012 to 4.4 billion in 2017 [4].

While corporate users engage professional software houses for their mobile application needs, end-users without programming knowhow are faced with a plethora of choices in the Google Play Store and iPhone App Store for theirs. There is an estimated 1 million apps in the Google Play Store in July 2013 [5]. This huge number of available apps ironically presents end-users with the difficulty of choosing the ideal one for their unique individual requirements. In fact, specialized solutions have been built or explored to solve this problem [6].

With funding from the Ministry of Education (MOE), Singapore, the authors have started this project in April 2013 to design and implement a sustainable solution for non programmers to create personalized mobile apps based on their own unique requirements and be able to share them easily.

2. Solution Model and Structures

The proposed solution consists of three major parts: (see Fig. 1)

- The App Builder GUI
- The Application Server for Application Repository and Distribution
- The App Engine (residing at the smartphone)

[+] Karl Kwan. Tel.: + (65-68704733); fax: +(65-67797912).
E-mail address: (karlkwan@sp.edu.sg).

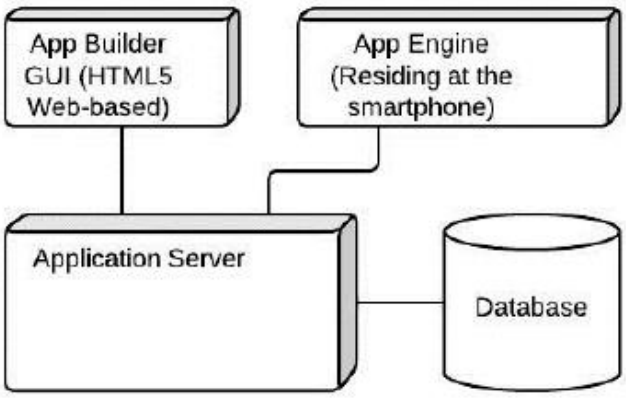

Fig. 1: System architecture

2.1. The Graphical User Interface (GUI) Based App Builder

The mobile app builder provides a set of GUI based building blocks to support the development of simple mobile apps. The GUI enables a user to create content and develop a mobile application using mainly graphical icons/widgets as building blocks without the need to get involved in complex coding. On top of the visual components, the builder also provides a widget attributes editor to enable users to further customise the flow of the application and the event driven actions.

The output of the app builder will be captured and stored along with the application logic and content based on an object model similar to W3C DOM [7].

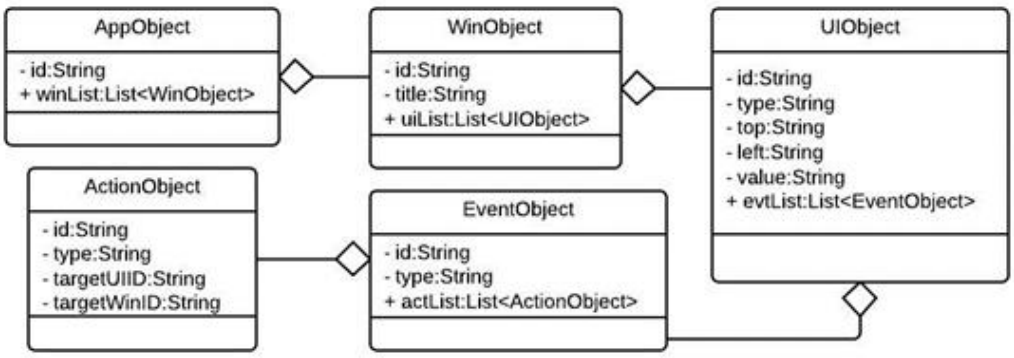

Fig. 2: A simplified view of the Application Object Model

As depicted in Figure 2, the application content and logical flow are encapsulated in a set of objects configured in a tree style. The root of the tree begins at a single AppObject node, which in turn contains winObject nodes, and UIObject nodes etc. To support the logical flow of an app, event driven actions are catered for via the EventObject nodes and the ActionObject nodes.

The actual implementation of the GUI is based on a HTML5 web interface. With the web-based interface, users can access the builder from any HTML5 enabled browsers.

2.2. The Application Server

All The generated application objects will be stored and managed by the Application Server. The Application Server is responsible for user management and application distribution.

As this product also enables users to distribute the personalized applications they have created, the server contains the users' contact information which can be used for sending out sharing invitations or notifications to the targeted user group.

The simplified workflow of the application sharing mechanism is depicted in Fig. 3.

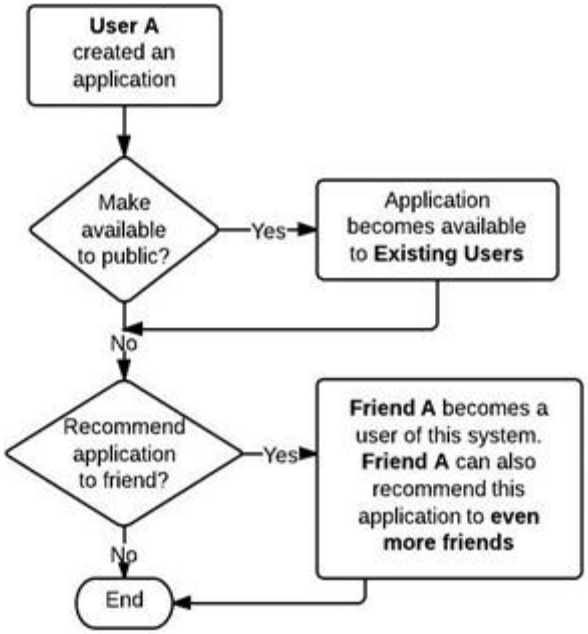

Fig. 3: Mobile Apps Sharing Workflow

2.3. The App Engine

The app engine is a customised mobile application that to be installed in the smart phone. The main function of the app engine is to retrieve the Application Objects from the Application Server and translate them into execution logic to carry out the user defined operations. Titanium Software Development Kit (SDK) is selected to be the implementation platform. It is a Javascript based SDK with the capability to create dynamic objects and functions.

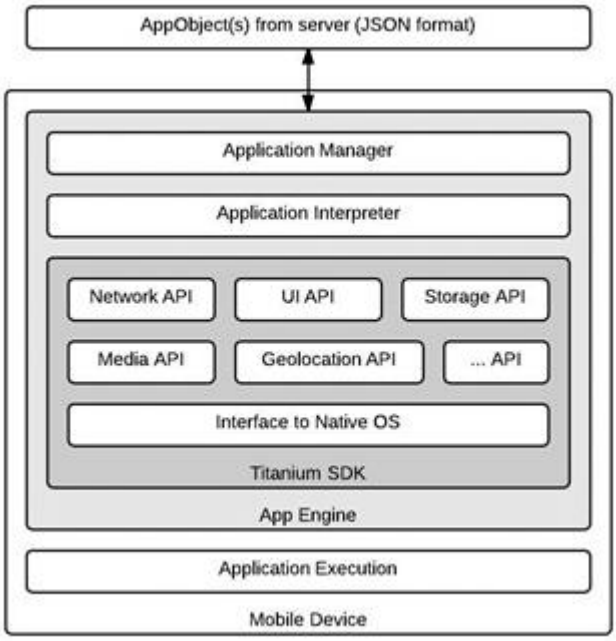

Fig. 4: App Engine Architecture

The architecture of the app engine is illustrated in Figure 4. The application interpreter module can access the smart phone resources via the various API routines supported by the Titanium SDK.

With the one code base approach, it makes it easier to maintain multiple versions of the app builder to support cross-platform solutions (i.e. IOS and Android).

3. Prototype and Initial Findings

A working prototype of the Application Server and some App Engine components has been built. Various types of AppObjects have been hardcoded to work in the prototype while the GUI app builder is still being developed. With the basic engine components, the prototype is able to support the implementations of WhatsApp-like messaging, cloud based image gallery browsing, as well as applications relating to personal and/or group reminders or group based check lists.

3.1. App Engine Construction with Titanium SDK

The Javascript based App Engine supports dynamic object generation and it works well with the Application object model (See Fig. 2). It is fairly straight forward to enable dynamic windows and widget objects creation on the fly as well as event listeners creation and event handlers assignment (See Fig. 5) .

```
function addEventListenerToUI(targetUI, eventObjDef) {
    targetUI.addEventListener(eventObjDef.type, function (evt) {
        //generate actions for event
        for (var w = 0; w < eventObjDef.actionlist.length; w++) {
            var action = eventObjDef.actionlist[w];
            if (action.type == "1") {/*do action 1*/}
            else if (action.type == "2") {/*do action 2*/}
            ...
        }
    });
}
```

Fig. 5: Sample of Event Listener Generation Code Snippet

Titanium SDK provides a comprehensive library to access various resources of the smart phone. A series of AppObjects has been implemented to test some common app widgets such as Google Map, message list, task list, navigation button, text field and others. Testing apps are built based on the combinations of these basic widgets. All the trial tests have been conducted successfully.

3.2. Performance, Use Cases and Limitations

The app engine is basically serving as an interpreter to execute the application logic defined by the AppObject model. Although the application response time and performance is slower than native apps, it is comparable to normal Titanium SDK based applications. However, one of the major limitations is that it is not feasible to implement computation and/or graphics intensive apps.

The working prototype shows the following potential:

For personal/group use:

- Private chat groups to exchange photo / message / location based information.
- Private or group shared task list.
- Personalized event based content sharing applications, such as wedding, outing or any other types of social parties.

For commercial/organization use:

- Content Distribution of marketing campaign.
- Mobile Apps based Customer Relationship Management frontend.
- Event based applications for tour groups, conferences, seminars, election drive, etc.

4. Future Deliverables, Impact and Opportunities

4.1. Deliverables of Next Phases

The current version of the prototype only works with the Android platform and the GUI App builder has not been integrated fully with the App Builder Server. The deliverables in the next phase (by Sep 2014), will include an enhanced version of App Engine that works in both IOS and Android platforms, an integrated GUI App builder and a fully functional application distribution solution. In the final phase of this project, the focus will be on end-users trials to collect user feedback to further optimise the usability and user acceptance level. This project is expected to complete by March 2015.

4.2. Potential Impacts and Opportunities

Solutions such as HyperCard [8] and Raptor [9], which use visual programming to build applications, are not new. Such solutions are good for developers to shorten the development time, or for students to pick up basic programming skills quickly. The focus in this project is to seek users' feedback to co-create a viable application builder for non-programmers.

The success of this project will enable a new pool of smart phone users who can optimize the potential usages of their devices to meet their personal needs via self-created apps. The impact of this solution may be comparable to how the electronic spreadsheet applications impact the personal computers. Both solutions are aimed at enabling end-users to harness more computing power from the computing devices.

5. Conclusion

Based on the initial working prototype and tests, the solution shows potential in meeting the project goal as planned, enabling non-programmers to:

- Define and create their own Mobile Apps.
- Distribute/share these apps to designated users / user communities.

The final deliverable will be working on both Android and IOS platforms.

This deliverable represents a new end-user centric model to the existing mobile Apps paradigms, suggested by Ngu P H [10].

6. References

[1] Thanasis. P, Antonis P, Michalis P, Evangelos P. M, Thomas K. Rise of the planet of the apps: a systematic study of the mobile app ecosystem . 2013 Conference on Internet measurement conference p277-290.

[2] Ayo Omojola. The Shortage Of Developer Talent Is Crushing Mobile – 15 July 2013. Retreived 27 Nov 2013, from *http://www.forbes.com*.

[3] AF-Studio.pl & Super Monitoring. State of Mobile 2013 Infographic. Retrieved 15 Oct 2013, from *http://www.digitalbuzzblog.com/infographic-2013-mobile-growth-statistics/.*

[4] Portio Research. Mobile Factbook – Feb 2013, pp. 49.

[5] Wikipedia. Google Play. Retrieved 15 Oct 2013, from *http://en.wikipedia.org/wiki/Google_Play*.

[6] Alex K,Linas B, Karen C,Mat B. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. 21st ACM intl. conf. on Information and knowledge management, p2527-2530. 2012.

[7] Document Object Model (DOM) Level 1 Specification, version 1. W3C Recommendation. 1998.

[8] Fred Stauder. HyperCard's User Friendly Programming. Retrieved 15 Oct 2013, from *http://www.mactech.com/articles/mactech/Vol.03/03.10/HyperCardProgramming/index.html*

[9] Martin C. C., Terry A. Wilson, Jeffrey W. Humphries, Steven M. Hadfield. RAPTOR: Introducing Programming to Non-Majors with Flowcharts. Department of Computer Science, United States Air Force Academy.

[10] Ngu P H, Do vanThanh. Evaluation of mobile app paradigms. 10th Intl. Conference on Advances in Mobile Computing & Multimedia, p25-30. 2012.