Journal of Engineering and Applied Sciences 13 (6): 1561-1566, 2018
ISSN: 1816-949X

Identifying UI Widgets of Mobile Applications from Sketch Images

[1]Seoyeon Kim, [1]Jisu Park, [1]Jinman Jung, [1]Seongbae Eun, [1]Young-Sun Yun, [2]Sunsup So,
[3]Bongjae Kim, [4]Hong Min and [5]Junyoung Heo
[1]Department of Information and Communication Engineering, Hannam University,
Daejeon, South Korea
[2]Computer Science and Engineering, Kongju National University, Kongju, South Korea
[3]Division of Computer Science and Engineering, Sunmoon University, Asan, South Korea
[4]School of Computer Information Engineering, Hoseo University, Cheonan, South Korea
[5]Department of Computer Engineering, Hansung University, Seoul, South Korea

Abstract: In this study, we present a statistical method for identifying UI widgets of mobile applications from sketch images using geometric and text analysis features. We classify the widgets into three components according to their characteristics: basic component, composition component and container component. We investigate the geometric feature: the relationship among graphic elements of widgets and the text analysis feature: the parts of speech of the widget text. We incorporate the geometric features of widgets in the rules through inclusion, combination and repetition relations in a statistical manner. The proposed method comprises the extraction of graphic elements such as text or shapes from the input sketch image using the Optical Character Recognition (OCR) technique and edge detection. The widgets are then identified by comparing them using the predefined rules. The method also provides meaningful features from the text of widgets. This can be useful in feedback about its recognitions to change the inference results.

Key words: Automatic programming, mobile application, UI component detection, UI code generation, statistical, composition

INTRODUCTION

For mobile application development, the majority of User experience (UX) designers design and implement using study and pencil or design tools such as photoshop (Newman and Landay, 1999). These User Interface (UI) sketch or drawing images have to be converted into UI code that actually works. Typically, this conversion is performed manually by the developer which results in high cost and large errors. We focus on the automatic generation of UI codes in mobile application programming. The UI design tools are also provided in Integration Development Environments (IDE) for mobile applications such as XCode (Allan, 2010) or Android studio (Zapata, 2013) but it requires a large amount of time and is expensive for designers to adapt to these complex development tools.

The detection of UI components from a sketch image is a difficult task owing to the variability of hand drawings. The various widgets make it hard to detect and recognize. Landay and Myers (2001) proposed an interactive UI design tool for recognizing widgets and other interface elements as the designer draws them. Parag *et al*. (2012) proposed a generic meta-grammar for producing hierarchical object description rules. In a rule-based environment they introduced a formal grammar for UI component detection. For mobile applications, Nguyen and Csallner (2015) proposed a technique for generating UI code via computer vision and Optical Character Recognition (OCR) techniques, called REMAUI. It automatically generates the UI portion of the source code of a mobile application using screenshots or conceptual drawings. Based on the research of Nguyen and Csallner (2015) computer vision techniques such as OCR technology or edge detection (Canny, 1986) were used to detect the UI components a core framework for mining the visual log of software was then proposed. They also performed a study on the characteristics of GUI elements in a mobile application.

In this study, we propose a method for identifying UI components from conceptual sketch images drawn by designers. Our approach differs from the existing methods in that it uses two geometric and text analysis features in order to improve accuracy. We first analyze the geometric

Corresponding Author: Jinman Jung, Department of Information and Communication Engineering, Hannam University, 70 Hamman-ro, Daedeok-gu, 306-791 Daejeon, South Korea

feature, i.e., the relationship among the graphic elements of the widgets. Note that, the majority of the hand-drawn sketch images of widgets can be expressed using relationships of simple text and shapes. We investigated the nine types of UI widgets in Android applications by classifying them into three components: basic component (TextView and Button), composition component (EditText, CheckButton, RadioButton and Spinner) and container component (ListView and TabBar). We used the results of this analysis to generate description rules of the various widgets. These rules are based on the relationships among graphic elements such as text and shapes (line, triangle, rectangle, circle, etc.). We also analyzed the text analysis features: the Parts of Speech (POS) of starting words in the text of widgets. Gordo *et al*. (2015) show that, it is possible to extract words directly from artificially generated word images. This is because the text of widgets usually contains semantic information.

The proposed method involves the extraction of graphic elements using computer vision technology (OCR technology and edge detection). By considering both geometric and text analysis features, it identifies the widgets via. the predefined rules and filters out an ambiguity via. text analysis information. Finally, each identified widget is combined with information regarding geometric properties and converted into the XML format file of a mobile application.

MATERIALS AND METHODS

Distribution of widgets: In order to understand the nature of the widgets, we downloaded the most popular apps from 10 different categories in the Google Play Store (USA) as shown in Fig. 1. We selected five screenshots of activities that have a core functionality in each App. A total of 50 screenshots from 10 categories were converted into sketch images using picture stencil maker. We analyzed nine types of widgets with high frequency of occurrence and divided them into three basic UI components: basic, composition and container components. We also, manually investigated the relationship among the graphic elements of the widgets and the POS of the text of widgets. The remaining types of widgets that appeared <1% are not considered.

Overall, 577 widgets were found in our dataset. The distribution of types was 63.08% basic component, 11.79% composition component and 25.13% container component as shown in Table 1. The widgets are distributed as 33.97% TextView, 26.69% Button, 2.43% ImageView, 4.51% CheckButton, 3.12% EditText, 2.25% Spinner, 1.91% RadioButton, 23.05% ListView and 2.08% TabBar in our dataset. The number of TextView and Button widgets was more than half of the total widgets.

Table 1: Distribution of widgets (%)

Variables	Values
Basic component	
TextView	33.97
Button	26.69
ImageView	2.43
Total	63.08
Composition component	
CheckButton	4.51
EditText	3.12
Spinner	2.25
RadioButton	1.91
Total	11.79
Container component	
ListView	23.05
TabBar	2.08
Total	25.13

Relationship among graphical elements for various widgets: Figure 2 shows the relationship among the graphical elements for various widgets. In the basic component, 93.9% of TextView widgets were found to have only text elements. The remaining 6.1% were indistinguishable from EditText because the border was wrapped or lined. The average height of TextView was approximately 2.32% of the total screen height. The majority of buttons are shaped as a rectangle or ellipse. The average height of the buttons was 4.32% of the total height of the screen and the average ratio of the width and height of the buttons was 2.2:1 which indicated a rectangular or elliptical shape. Furthermore, 92.9% of ImageView widgets comprised only images and 7.1% of them included text in the image. The average height of the ImageView widget was approximately 11% of the total screen height; 71.4% of the ImageView widgets were centered.

In the composition components, CheckButton comprised a combination of a rectangle and text of which in 84.6% of the cases, the rectangle was on the right. The average height of the rectangle combined with the CheckButton widget was approximately 3.69% of the total screen height which was smaller than the button widget and the average ratio of the width and height of the rectangle was approximately 1.1:1 which indicates that is was almost a square. EditText comprised a combination of text and lines, accounting for 55.6% of the total. The remainders are bordered by rectangles or consist only of text which make it, especially, difficult to distinguish them from button or TextView widgets, respectively. The average height of the EditText widget was 2.49% of the total screen height and 94.4% was on the left in the layout. Spinner widgets had 69.2% arrows or triangles

Fig. 1: Screenshots of 10 Apps. for empirical study (from various categories in Google Play Store): a) Uber Eats (Food and drink); b) Starbucks (Lifestyles); c) Bible (Book); d) Sure (House and Home); e) Twitter (News); f) PayPal (Finance); g) Snapchat (Social); h) Wish (Shopping); i) Google translate (Tool) and j) Uber (Map and navigation)

combined with text and the rest had only text or only an image. The average size of the arrow or triangle was relatively small (0.99% of the entire screen) and the position of the shape was 40% on the left and 60% on the right. All of the RadioButton widgets were circular and text combined and the circle was positioned on the left. The average distance between the circle and the text was 4.15% and the average height was 4.12%.

In the container component, ListView shows a list of items that contain various type of widgets. These items showed a repeated characteristic and the average number of repetitions of the items was 3.9 times. The average height of each item was 8.86% of the total screen height; 32.3% of the items comprised only TextView while 30.8% comprised both TextView and button widgets where in the TextView widgets were placed first in the majority of cases. In the case of TabBar, the average number of

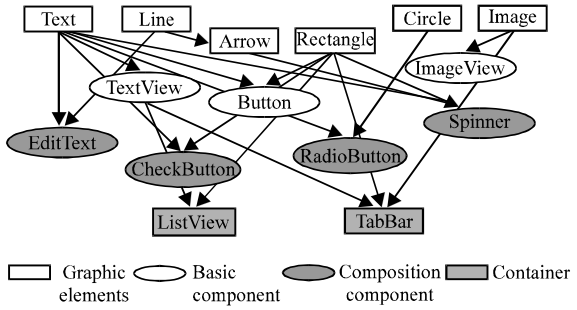

Fig. 2: Relationships among graphic elements of widgets

repetitions of the text specifying the tab was 3.58 times and the average height of the tab's text was 6.63% of the total screen height, 32.3% of the tabs comprised only text while 30.8% comprised text and an image.

RESULTS AND DISCUSSION

Automatic UI code generation using two geometric and text analysis features: One of the objectives is to analyze a statistical model using feature extraction from widgets. The feature extraction of widgets is based on inter-graphic element relationship: relationship between graphic elements and geometric properties of graphic elements: relative size, position and orientation. The relationships between graphic elements are defined as inclusion, composition and repetition relations according to the characteristics of these graphical elements. From these primitives, widgets of sketch images are classified as basic, composition and container components. The relative size of the graphic elements also acts as a parameter of the widget classification. The purpose of the feature extraction is to identify the mapping between the widgets and description rules. The notations used in this study are shown in Table 2.

Using the analysis of statistical relations, we describe the rules of the nine types of widgets as shown in Table 3. Basic components are composed of elements themselves or bordered by inclusive shapes. For example, the majority of TextView widgets have only text (>90%) and while the button widget has a rectangle or circle border. Composition components are represented by a combination of text and various shapes. EditText is combined with a line, CheckButton with a rectangle, RadioButton with a circle and spinner with a triangle, respectively. Relative sizes can also be used as characteristic properties. Container components such as ListView and TabBar can be characterized through the repetition relations of the widgets.

Table 2: Notations used in this study

Notation	Description
○, ⊕, +	Relationship: inclusion, composition, repetition, respectively
·, ·, s	Graphic elements: text, image, shape, respectively
v, h, t, r, c	Shapes: vertical line, horizontal line, triangle, rectangle, circle, respectively
·	Widgets

Table 3: Description rules of widgets

Components/Widgets	Description rules	Relative size or repetitions
Basic		
TextView	T→·	2.29%
Button	B→(· ○c)|(· ○r)	4.32%, 1:2.2
ImageView	I→·	11%
Composition		
CheckButton	CB→(· ⊕r)	3.69%, 1:1.1
EditText	ET→(· ⊕h)	2.49%
Spinner	SP→(· ⊕t)	0.99%
RadioButton	RB→(· ⊕c)	4.12%
Container		
ListView	LV→(·)+	Vertically 3.9
TabBar	TB→(·) +	Horizontally 3.58

The widget identification approach that uses rules of these relationships can cause ambiguity. Figure 3a shows a TextView from the UberEats screenshot that is bordered by a rectangle. This can be incorrectly detected as a button based on the inclusion relation. Figure 3b shows a button from the snapchat screen shot which can be detected as a RadioButton belonging to the composition component owing to the composition relation of the circle and text. For a button, a false negative can occur if there is no border. Figure 3c shows an example of a multi-line TextView shown in the SURE screenshot. This is the case wherein the TextView can be incorrectly detected as a list view belonging to the container component based on the repetition relation. We studied the text contained in the widgets in order to eliminate this ambiguity.

The goal of this stage is to disambiguate the detection of widgets using text analysis features. Text analysis features involving POS tags and lexical characteristics are useful features for detecting ambiguous widgets such as basic components (TextView or button) that contain text. The POS starting word in the text contained within the widget implies the semantic features of the widget. A button that provides the user a simple method for triggering actions contains a short text beginning with a verb (e.g., add, get and buy). Table 4 shows the proportion of POS and length for starting words in texts of widgets. Since, the TextView is a widget that outputs simple information, it begins with 61% nouns. A button that triggers an action from the user, 60% starts with the verb. The verb "add" was most commonly used at 20%, followed by "get." A long button that contains a long text (more than five words) was not found and most of this text were found in TextView and

Table 4: Proportion of POS and length of starting words in widget text

POS and length	T (%)	B (%)	ET (%)	CB (%)	RB (%)	SP (%)	LV (%)	TB (%)
Noun	61	35	53	73	55	55	78	62
Verb	19	60	47	19	18	45	16	14
Adjective	6	5		4	-	-	5	24
Preposition	4	-		-	9	-	2	-
Adverb	2	-		4	18	-	-	-
Numbers	8	-		-	-	-	-	-
Ratio of long sentences (more than five words)	15	-	18	12	18	9	2	-

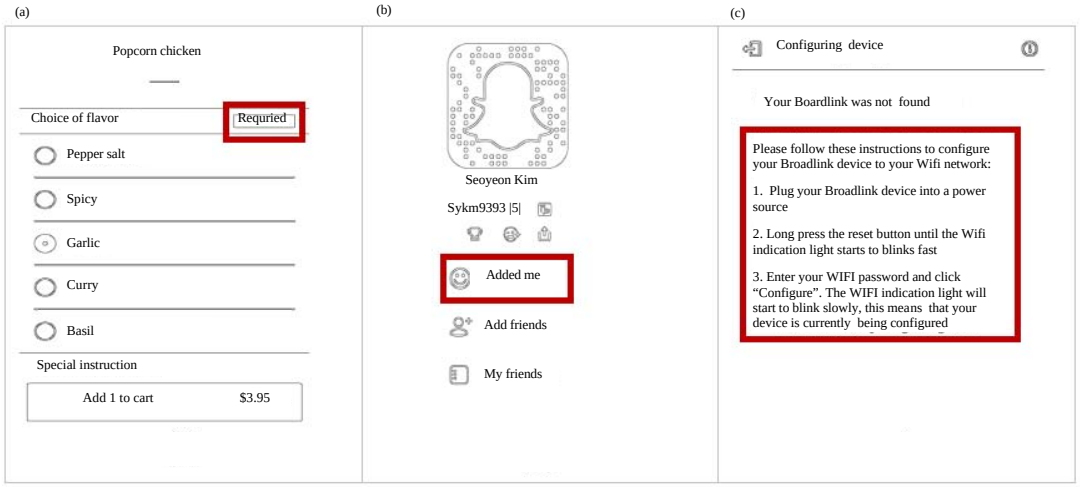

Fig. 3: a-c) Ambiguity in description rules

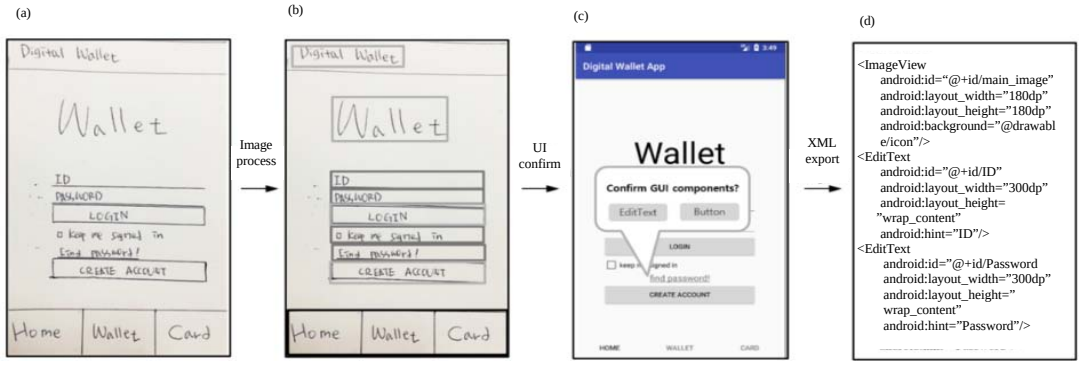

Fig. 4: Automatic UI code generation mechanism based on sketch image: a) Sketch image; b) Detection of UI components; c) Confirm GUI components and d) Automatic UI code generation

EditText. We use the simple text analysis features to support robustly detection of UI widgets by removing any ambiguity.

We design a technique for identifying UI components of widgets from conceptual UI sketch images drawn by designers and to recognize the relative size and position and export the appropriate UI source code. Figure 4 shows our procedure for detection of UI widgets from sketch images. The proposed method involves the preprocessing of a sketch image and the extraction of text from the sketch image using OCR technology. It can be used to extract contours from the sketch image through edge detection (e.g., Canny algorithm) and identify the widgets using predefined rules. The enhancement of the detection accuracy would require the use of sophisticated heuristics of text to handle the ambiguity and feedback regarding its inference results. For example, "Find password" in a sketch image can be identified as an EditText owing to the combination of the line and text but it is delivered as a candidate such that the user can make

a final decision after the text analysis. The method can then be used to merge and convert the identified UI information into an XML format file for a mobile application.

CONCLUSION

The main contribution of this study is a technique for identifying UI widgets of mobile applications as well as providing a statistical model of UI widgets given sketch images containing UI widgets. We propose a method for identifying UI widgets of mobile applications that is focused on geometric and text analyses. The geometric features are identified by using inclusion, composition and repetition rules that take into consideration the text and shape relations of widgets. The text analysis robustly supports the detection of UI widgets with text features needed to remove any ambiguity of detection. Our approach provides feedback to designers in order for them to select the final results.

ACKNOWLEDGEMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (NRF-2017M3C4A7069067,NRF-2017M3C4A7069073)and supported by 2017 Hannam University Research Fund.

REFERENCES

Allan, A., 2010. Learning iPhone Programming: From Xcode to App Store. 1st Edn., O'Reilly Media, Sebastopol, California, ISBN:978-0-596-80643-9, Pages: 359.

Canny, J., 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell., 8: 679-698.

Gordo, A., J. Almazan, N. Murray and F. Perronin, 2015. Lewis: Latent embeddings for word images and their semantics. Proceedings of the IEEE International Conference on Computer Vision (ICCV'15), December 7-13, 2015, IEEE, Santiago, Chile, ISBN:978-1-4673-8390-5, pp: 1242-1250.

Landay, J.A. and B.A. Myers, 2001. Sketching interfaces: Toward more human interface design. Comput., 34: 56-64.

Newman, M.W. and J.A. Landay, 1999. Sitemaps, storyboards and specifications: A sketch of web site design practice as manifested through artifacts. MSc Thesis, Electrical Engineering and Computer Sciences Department, University of California, Berkeley, California.

Nguyen, T.A. and C. Csallner, 2015. Reverse engineering mobile application user interfaces with remaui (T). Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15), November 9-13, 2015, IEEE, Lincoln, USA, ISBN:978-1-5090-0025-8, pp: 248-259.

Parag, T., C. Bahlmann, V. Shet and M. Singh, 2012. A grammar for hierarchical object descriptions in logic programs. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'12), June 16-21, 2012, IEEE, Providence, USA., ISBN: 978-1-4673-1611-8, pp: 33-38.

Zapata, B.C., 2013. Android Studio Application Development. Packt Publishing, Birmingham, England, ISBN:978-1-78328-527-3.