SCIENCE CHINA
Information Sciences

September 2020, Vol. 63 190105:1–190105:3
https://doi.org/10.1007/s11432-019-2860-3

• LETTER •
Special Focus on Big Data-Driven Intelligent Software Development for Onsite Programming

A data-driven approach for recommending UI element layout

Yonghao LONG[1], Xiangping CHEN[2*] & Yuting XIE[1]

[1]*School of Information Science and Technology, National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou* 510006*, China;*
[2]*Guangdong Key Laboratory for Big Data Analysis and Simulation of Public Opinion, School of Communication and Design, Sun Yat-sen University, Guangzhou* 510006*, China*

Received 19 September 2019/Revised 31 December 2019/Accepted 20 March 2020/Published online 7 August 2020

Citation Long Y H, Chen X P, Xie Y T. A data-driven approach for recommending UI element layout. Sci China Inf Sci, 2020, 63(9): 190105, https://doi.org/10.1007/s11432-019-2860-3

Dear editor,

Smartphones and tablets with rich graphical user interfaces (GUI) are becoming increasingly popular. GUI design plays an important role in offering a smooth user experience. The complexity of the apps often depends on the user interface (UI) [1] with minor data processing or data processing delegates to the backend component.

When developing a UI, considerable effort can be saved by seeking inspiration from examples than by starting a design from scratch [2]. Considerable work has been proposed to ease GUI development. Doppio [3] generated a screenflow diagram organized by callback methods and interaction flow to assist developers to understand user interaction implementation. GUI code search engineer [4] was proposed to generate and recommend UI implementation examples. In some previous study, knowledge was gained from UI examples that facilitated in guiding UI development [5, 6]. In these studies, datasets containing a large number of UI examples were constructed, and the examples were provided as a design gallery and were used to mine design patterns.

Fast prototyping development of mobile applications requires the understanding of user habits and design trends when designing the layout of UI elements. To facilitate the developer to read a large number of UI design cases, we propose a data-driven approach that attempts to mine the UI element layout pattern and recommends proper positions and sizes for layout implementation of the UI visual component. Our approach collects the layout information of UI elements from apps using a black-box testing tool [7]. Based on a repository [7] containing more than 3585976 UI elements of 146942 UI pages collected from 21233 Android apps, our approach analyzes the layout distribution of the elements, which contains the keyword and are of the same component type, to determine whether a layout pattern exists. If so, the layout will be extracted and recommended for implementation. The experiment conducted by us shows that the recommended layouts can be used to guide UI development.

Approach overview. A novice developer who is not sure about the layout of an element can input the keyword and the element's type to our tool. As shown in Figure 1, suppose the user wants to search the skip button's layout and inputs the keyword "skip" and the type "button", then our method will generate the sample set from the dataset [7]. Based on the elements' layout information, our method will generate a heatmap where the color of each pixel represents the layout possibility. Subsequently, the approach mines the frequent pattern based on edge detection and heuristic rules. Finally, the skip button's recommendation layout is showed to the user.

Heatmap generation. Our approach selects UI

* Corresponding author (email: chenxp8@mail.sysu.edu.cn)

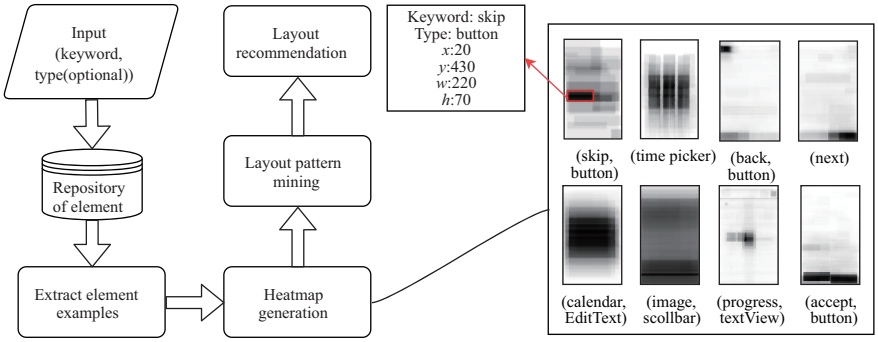

Figure 1 (Color online) Approach overview.

element examples according to the input keywords with component type as an optional supplement. Because the types of UI elements are limited, it is hard to find a layout pattern for a particular type. Keywords are texts in the UI element, which usually describe or imply the functional information of a component. Keywords with the component types are used to define which UI elements can be used as examples. It should be noted that text information of the UI elements is processed by standard normalization techniques of stemming and removing stop words. If the number of UI element examples is large enough for mining layout patterns (a threshold of 100 is defined in this article), the examples will be extracted.

A heatmap is a graphical way of displaying a table of numbers using colors to represent the numerical values. In this study, a heatmap is used to indicate the possibility of the component appearing in each UI pixel. For a pixel $\langle x, y\rangle$, we define its possibility $p(x, y)$ considering the number of UI components covering this pixel $c\_n(x, y)$, and the maximum number of UI components coverage among all the pixels $\langle i, j\rangle$ in the UI.

$$p(x, y) = \frac{c\_n(x, y)}{\max_{(i,j)\in \mathrm{UI}} c\_n(i, j)}. \tag{1}$$

The number of words in the UI elements may differ and affect the importance of the UI element for a specific input keyword. For example, a UI element containing "If you like our app, please rate it!" and a UI element containing "like" may have different importance for the button with the input keyword "like". In addition, we find that some apps use templates to generate UIs. An application may contain a set of interfaces with the same UI element in the same position. The times of keywords' appearance in component c covering the pixel $\langle x, y\rangle$ as $\mathrm{cover}(c, x, y, \mathrm{keyword})$ is computed and the number of UI components for a pixel $c\_n(x, y)$ is computed as

$$c\_n(x, y) = \sum_{c\in E} \frac{\mathrm{cover}(c, x, y, \mathrm{keyword})}{\sqrt{|c.\mathrm{word}| \cdot \mathrm{App}(c)}}, \tag{2}$$

where $|c.\mathrm{word}|$ is the number of words in this component and $\mathrm{App}(c)$ represents the number of component appearances in all UIs of the same app.

Then, the heatmap can be drawn using the following rules: the heatmap starts from a grayscale image of white with the most commonly used size of 480×800. The color of each pixel depends on its distribution probability: the higher the probability is, the darker the pixel's color will be

$$\mathrm{color}(x, y) = 255 \cdot (1 - p(x, y)). \tag{3}$$

Mining layout patterns. As a UI element may be in different sizes and placed in different positions, the most frequent layouts professional developers prefer to use are needed to be mined.

First, we evaluate whether the heatmap H contains a layout pattern. According to the statistics of our dataset, we find that the possibility of a layout pattern existing is positively correlated with the average heat and the hot-area's size. The score to indicate the possibility of a layout pattern $s(H)$ is calculated as

$$\begin{cases} s(H) = \dfrac{\sum_{\langle x,y\rangle\in H} p(x, y)}{\mathrm{count}(H, 0)} \cdot \dfrac{\mathrm{count}(H, t)}{\mathrm{fs}(E)}, \\ \mathrm{count}(H, n) = \displaystyle\sum_{\langle x,y\rangle\in H} p(x, y) > n?1 : 0. \end{cases} \tag{4}$$

However, the size of the layout pattern is not known and more than one pattern for recommendation may exist. The closed areas in the heatmap are extracted as candidate patterns. Opening operations are used to remove the noise and the edge detection method [8] is used to obtain the bounds. The edge detection result is a set of closed areas.

We calculate the score of candidate layout pattern lp based on the average distribution possibility $p(x, y)$ of all the pixels and the distance between the size of lp and the most frequent size

in UI element examples $\mathrm{fs}(E)$ using the following equation:

$$\mathrm{score}(\mathrm{lp}) = \frac{\sum_{\langle x,y\rangle\in \mathrm{lp}} p(x,y)}{\mathrm{size}(\mathrm{lp}) \cdot \max(\frac{\mathrm{fs}(E)}{\mathrm{size}(\mathrm{lp})}, \frac{\mathrm{size}(\mathrm{lp})}{\mathrm{fs}(E)})}. \quad (5)$$

The top-3 layout patterns are then recommended. For each layout pattern, a description of its position and size is provided, as shown in Figure 1.

Experiments. Two research questions are proposed in this study to check the correctness and effectiveness of the proposed method. RQ1: How effective the proposed approach is in detecting layout patterns? RQ2: Are the recommended layout patterns valuable for developers?

Regarding RQ1, 50 pairs of keywords and component types were randomly selected and the proposed approach was used to select the UI element examples to generate heatmaps. These heatmaps are considered with layout patterns by the proposed approach. Then, 25 participants (8 of them were Android developers) were asked to answer whether they can find the layout patterns in the heatmap and manually draw the bounds of the patterns in the heatmaps. The 50 heatmaps were divided into 5 groups, and each heatmap was presented to 5 participants. A heatmap was considered to contain a layout pattern if more than 3 participants agreed on it. The edge detection result was considered to be identical if the offsets of x and y-axis between the two results were less than 5 pixels. The detection results were compared with the results drawn by the participants. The experimental results showed that 41 heatmaps (82%) were thought to provide considerable location recommendations and the proposed method could detect the bounds of components with an accuracy of 60.4%. This accuracy is relatively low because not all the participants can correctly extract layout patterns.

Regarding RQ2, heatmaps for 50 pairs of keywords and component types were generated and the layout patterns were also generated. Then, 25 participants (13 of them were experienced users and 12 were Android developers) were asked to evaluate whether the recommending patterns were valuable. The 50 heatmaps were divided into 5 groups, and each heatmap group was presented to 5 participants. It was found that 84.8% of the recommended patterns were considered as valuable.

A comparison experiment was conducted to compare the proposed method's results with the random method results. For the 50 pairs of keywords and component types used in RQ2, 50 corresponding samples from the proposed dataset were randomly selected. For each pair of keyword and component type, two layout recommendations were generated: one using the proposed method and the other from the sample. Then, 5 questionnaires were sent to 25 participants. Each questionnaire contained 10 questions. Participants were asked to select the recommendations that were consistent with their knowledge of the components' layouts. The recommendation for a pair of keyword and component type was judged by 5 participants. The recommendation selected by the majority of the participants was set as the components' layout recommendation. 43 recommendations by the proposed method were regarded as satisfying, which was much more than the corresponding number of random sampling (7 samples were selected to be satisfying).

Conclusion. We propose an approach to mine layout patterns and recommend the proper placement and size for UI layout implementation. Based on 146942 UI pages collected from 21233 Android apps, our approach could extract the UI element examples to generate a heatmap based on user input. Heuristic rules and edge detection were used to find layout patterns. The experiment results show that our method is effective in recommending valuable layout patterns and the accuracy of our approach needs to be improved.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFB-1004800), National Natural Science Foundation of China (Grant No. 61672545), and Science and Technology Program of Guangzhou (Grant No. 201902010056).

References

1 MacHiry A, Tahiliani R, Naik M. Dynodroid: an input generation system for Android apps. In: Proceedings of the 9th Joint Meeting on Foundations of Software Engineering, 2013. 224–234
2 Miller S R, Chang C C, Krantzler J, et al. Getting inspired!: understanding how and why examples are used in creative design practice. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, 2009. 87–96
3 Chi P Y, Hu S P, Li Y. Doppio: tracking UI flows and code changes for app development. In: Proceedings of CHI Conference on Human Factors in Computing Systems, 2018
4 Reiss S P. Seeking the user interface. Autom Softw Eng, 2014, 25: 103–114
5 Kumar R, Satyanarayan A, Torres C, et al. Webzeitgeist: design mining the web. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, 2013. 3083–3092
6 Deka B, Huang Z F, Chad F, et al. Rico: a mobile app dataset for building data-driven design applications. In: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, 2017. 845–854
7 Chen X P, Zou Q W, Fan B T, et al. Recommending software features for mobile applications based on user interface comparison. Requir Eng, 2019, 24: 545–559
8 Marr D, Hildreth E. Theory of edge detection. Proc R Soc Lond B, 1980, 207: 187–217