# SAIP: A Prototype Tool for Semi-Automatic User Interface Prototyping

Yuqing Chen, Shengqi Cao, Guozhang Ma and Tong Li

Beijing University of Technology, Beijing 100124, P.R.China

#### Abstract

User interface prototyping is an efficient approach for eliciting user requirements in the early phase of software development. It is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. It is typically done in a drag-and-drop fashion by professionals, which is time-consuming and requires specialized knowledge. This paper presents a prototype tool SAIP (Semi-Automatic user Interface Prototyping), which allows users to express their natural language requirements and semi-automatically generates a corresponding graphical user interface. Based on a systematic literature review of the user interface prototype, we design a framework that maps natural language to the user interface prototype. Furthermore, develop a demo version of the SAIP - a user interface prototyping tool served from natural language processing and neural network for the user interface automatic generation. Our tool aims to help users who have little knowledge in user interface design to express their ideas and requirements freely.

#### **Keywords**

User interface prototyping, Natural language processing, Prototype tool, Neural network

## 1. Introduction

User Interface prototyping is an essential part of software engineering, effectively facilitating requirements elicitation. It is typically done in a drag-and-drop fashion, using prototyping software such as Adobe XD<sup>1</sup>. However, it is time-consuming to learn and master such techniques. The prototypes of mentioned software are designed by individual users and then uploaded on the official website to allow other users to edit them, which will duplicate styles.

In recent years, many researchers and developers have experimented with the automatic generation of user interfaces. As early as 2005, Nichols et al. proposed model-based concepts to solve automatic interface generation from a theoretical level. [1] Tony Beltramelli in 2017 proposed a deep learning method to transform screenshots of user GUI into computer code with over 77 % accuracy. [2] Another recent example is DeepCoder [3] - a system that can generate computer programs by leveraging statistical predictions to augment traditional search

D 0000-0002-8881-0037 (T. Li)

In: F.B. Aydemir, C. Gralha, S. Abualhaija, T. Breaux, M. Daneva, N. Ernst, A. Ferrari, X. Franch, S. Ghanavati, E. Groen, R. Guizzardi, J. Guo, A. Herrmann, J. Horkoff, P. Mennig, E. Paja, A. Perini, N. Seyff, A. Susi, A. Vogelsang (eds.): Joint Proceedings of REFSQ-2021 Workshops, OpenRE, Posters and Tools Track, and Doctoral Symposium, Essen, Germany, 12-04-2021

Chenyqsigua@emails.bjut.edu.cn (Y. Chen); caoshengqi@emails.bjut.edu.cn (S. Cao);

maguozhang@emails.bjut.edu.cn (G. Ma); litong@bjut.edu.cn (T. Li)

<sup>© 0 2021</sup> Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

<sup>&</sup>lt;sup>1</sup>https://www.adobe.com/products/xd.html

techniques. Despite their success, we still have many problems to solve. For example, Tony Beltramelli's approach still requires an existing user interface to generate another interface. We wondered if there could be a way to help designers without computer knowledge and people with a need, to solve the problem of quickly creating deliverable interface prototypes.

This paper presents a practical prototype tool that takes natural language as input and directly generates corresponding user interface prototypes using NLP (Natural Language Processing) and image processing techniques. It can generate customizable interface prototypes that are non-repetitive.Compared with commercial products generated by models such as Canva, its prototype recommendations and functions are recommended based on user popularity and downloads. This means that the more exquisite and the higher the prototype's ranking, the more users it is, the more likely it is to be reused by different people. When designing products, designers hope to refer to more innovative prototypes.

For products such as WordPress that provide static templates, we believe that in the early stages of prototyping, the introduction of visual templates will affect users' freedom of expression and limit the width of users' thinking. Using text, a more abstract and faster way, can give users more room for innovation and design more novel prototypes. Specifically, our proposed prototype functions in a pipeline manner, which recommends various interface designs to users.(e.g., layouts and colors of different components) If users' requirements are not satisfied by the recommended solution, our tool also provides a self-development mode, allowing users to customize the designed interfaces. At present, SAIP already has a demo version - a demonstration video can be found here.<sup>2</sup>

## 2. A Semi-Automatic User Prototyping Framework

Figure 1 illustrates the main workflow of our framework. We divide the entire project into two layers, the Context Layer and the Rendering Layer, used to receive and understand the user's requirements for the user interface prototype. The user will input the prototype's layout requirements and the color scheme requirements in two steps, in natural language.

#### 2.1. Layout Requirements

According to interaction patterns in user interfaces, it can be understood that the layout of user interfaces can be broadly classified into a limited number of categories. Moreover, these categories correspond to website features that are distinct and well-differentiated. For example, we naturally think of YouTube for streaming sites, which uses a peer-to-peer grid pattern to show as many video covers as possible to attract viewers.

We then classify the natural language input of users by manually labeling the corresponding layouts of websites with different features and then training the framework through neural network. In this form, we can determine what type of website is roughly described by the user's natural language and adopt the mainstream layout to meet the user's needs. Specifically, our proposal starts from word tokenization, which is an inevitable step for processing Chinese.

<sup>&</sup>lt;sup>2</sup>https://tinyurl.com/9pofvkaf



Figure 1: Complete workflow of SAIP

Then, we conduct TF-IDF (Term Frequency–Inverse Document Frequency) to extract key words, identifying the core descriptions of user inputs.

The core descriptions are then inputted into the neural network model. Our labeled training data-set trains this model in advance, which can classify the user input requirements and map them to a specific layout. If there is no successful mapping (i.e., the classification probability is below a threshold), it defaults to the most widely adopted center stage layout.

#### 2.2. Color Requirements

Impressive color style is essential to catch the user's attention at first glance. For better aesthetics and simplicity, modern single-page applications should ideally have no more than three colors, with Primary Color for mainframes and components and Secondary Color for secondary accents.<sup>3</sup> So the question turned to - how to extract the color preference from the natural language description?

The human brain can imagine different colors directly through spoken words. For example, people can quickly think of colors like black, gray, and navy for business style. For another example, steak, people can easily associate with red or brown. Nevertheless, computers are difficult to associate colors directly from natural language except for much training for computers. In daily life, people often use natural language in search engines to search for images. Images

<sup>&</sup>lt;sup>3</sup>https://material.io/design/color/the-color-system.html#color-theme-creation



Figure 2: De-depuplication result and color recommendation system.

can be transformed into colors using extraction algorithms. Thus, our approach is to use search engines to transform natural language into images and then use a color extraction algorithm to extract the potential colors from the images.

However, it would be very messy to give this result to users directly. First, there are multiple images for each term. In this case, too many colors will be recommended to the user. Secondly, the duplication rate of colors would be very high. The color extraction algorithm is segmented based on pixel points, and many colors have slight differences in RGB, but the colors are almost indistinguishable to the naked eye. So the tool needs to de-duplicate the colors to remove the similar ones based on their Euclidean distances in RGB space [4].

In de-duplication process, if a color is removed because of duplication, it means that it is too similar to another color, meaning that this color appears more often. So our prototype tool records the weights of different colors. When a color is de-duplicated, then tool will boost its weight and prioritize the subsequent recommendations.

The impact of doing this are twofolds. First, if the user input requirements are too distinct, all the colors are more similar, and fewer colors are available after de-duplication. The effect of de-duplication is shown in Figure 2(a). Second, the tool's de-duplication algorithm keeps the higher weight of the two colors, which may remove the color that the user wants. The last step of color processing is that our tool uses an improved Item-based Collaborative Filtering (ItemCF) algorithm [5] to analyze the weight of colors extracted from the user's input. Associate the colors that the user may want and recommend them based on the weight ranking. Figure 2(b) shows our proposed color recommendation based on an improved ItemCF.

#### 2.3. Rendering

After the above two steps, the tool extracts the layout and color scheme from the user's requirements, and all that is left is to render it to the front-end. The approach we currently use is to pre-set up some front-end templates that can be inserted. These templates have distinct layout features, and each template corresponds to the layout strategy. The tool modifies the color scheme of the different components to meet the user's color scheme requirements.

Table 1The technical architecture of SAIP

Layer	Related techniques
Rendering	React.js, Material-UI, Redux
Server Layer	FastAPI, Nginx, Uvicorn
Algorithm	Jieba Tokenizer, fastText, Median Cut Algorithm
Data Source	Manual Labeling, Tencent Al, Google Image

Simultaneously, a drag-and-drop page layout feature is embedded in the tool to help users make adjustments to the laid-out pages.

# 3. A Prototype Tool

Based on the above research, we have developed a prototype tool, implementing semi-automatic user interface prototyping. The architecture of our tool is shown in Table 1. Technically speaking, given the requirements text in Chinese, we first perform tokenization based on Trie tree structure, Viterbi algorithm, and HMM model<sup>4</sup>. Then we use FastText, which is fast and robust for small data-set, to process word vectors and embed them into neural networks.

Moreover, to complete the classification of different words and phrases on the layout, we manually label more than 2000 words and sentences. The tag types are mainly different web page layouts (e.g., peer-to-peer grid mode, heading area mode) and different template names. Meanwhile, to solve the ambiguity caused by user input errors, we use the Tencent NLP library to help us do text error correction and find synonyms.<sup>5</sup>

For color extraction, we use the mainstream Median Cut Method [6] to extract up to 256 colors from a single image. Based on the layout and color scheme extracted, the prototype will be rendered. Finally, we use the industry's most popular React.js framework and the Material-UI component library to render the interface of our tool. To support customization, our tool has implemented drag-and-drop modification using react-grid-layout<sup>6</sup>.

## 4. Conclusion and Future Work

This paper introduces SAIP, a prototype tool for Semi-Automatic user Interface Prototyping. It takes layout and color requirements as input in natural language and generates corresponding user interface prototypes, assisting in requirements elicitation.

Our tool is mainly aimed at prototype creators with a light technical background or designers who need to prototype quickly. First of all, it follows the idea of rapid prototyping in software engineering. A brief natural language input could automatically generate a preview interface prototype for the users. This quickly and easily helps users to define their requirements, eliminating the need for multiple iterations for further development. Secondly, each interface prototype generated by the system can be adjusted and modified, and software engineers can

<sup>&</sup>lt;sup>4</sup>https://github.com/fxsjy/jieba

<sup>&</sup>lt;sup>5</sup>https://cloud.tencent.com/document/product/271

<sup>&</sup>lt;sup>6</sup>https://github.com/STRML/react-grid-layout

iterate and improve. Thirdly, according to human-computer interaction theory[7], layout and color are two of the essential factors for interface prototypes. Our current functions can satisfy most user needs, giving a prototype interface that is simple to use for reference. In the future, we will add more features, such as adding font style selection, component selection, and other features to meet precise needs.

In the future, our primary task is to use machine learning techniques to replace the manual labeling approach to more accurately obtain the layout and colors mapped by user needs. For example, refer to the VIPS algorithm [8] to extensively learn the layout and color scheme of websites, optimize the classification model, and recommend more reasonable and richer layouts for users. And derive the DOM tree of the prototype interface from helping further software engineers to develop it. Besides, we will improve our functions, and by exploring natural language processing techniques, we hope to meet more user needs, such as analyzing user needs for functions, to derive a set of interface groups with internal logical relationships, including jumps and links to different pages. Finally, we will improve the robustness and usability of our system to make it more mature.

For the empirical evaluation, our plan is as follows: First, we will improve our current tool's stability and usability and make it available to any Internet user by using our server to launch our product. After that, we will publish a survey about our tool and collect feedback from users to extend our tool from a functional perspective. After several iterations, our ultimate goal is to make it commercially available and accessible to all prototype designers.

## Acknowledgments

This work is partially supported by the National Natural Science of Foundation of China (No.61902010), Beijing Excellent Talent Funding-Youth Project (No.2018000020124G039), and the Project of Beijing Municipal Education Commission (No.KM202110005025).

## References

- [1] J. Nichols, A. Faulring, Automatic interface generation and future user interface tools, in: ACM CHI 2005 Workshop on the future of user interface design tools, 2005.
- [2] T. Beltramelli, Pix2code: Generating code from a graphical user interface screenshot, in: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2018.
- [3] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, D. Tarlow, Deepcoder: Learning to write programs, arXiv preprint arXiv:1611.01989 (2016).
- [4] D. H. Alman, Cie technical committee 1–29, industrial color-difference evaluation progress report, Color Research & Application 18 (1993) 137–139.
- [5] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Computing 7 (2003) 76–80.
- [6] D. S. Bloomberg, Color quantization using octrees, Leptonica, ss (2008) 1–10.
- [7] M. G. Geraci, Designing web-based instruction: A research review on color, typography, layout, and screen density, University of Oregon, Rapport technique (2002).

[8] D. Cai, S. Yu, J.-R. Wen, W.-Y. Ma, VIPS: a Vision-based Page Segmentation Algorithm, Technical Report MSR-TR-2003-79, 2003. URL: https://www.microsoft.com/en-us/research/ publication/vips-a-vision-based-page-segmentation-algorithm/.