2019 5th International Conference for Convergence in Technology (I2CT)
Pune, India. Mar 29-31, 2019

Front End Development Automation Tool: Missing Features?

Hasitha Hiran Walpola
Informatics Institute of Technology
No 57, Ramakrishna Road, Colombo 6, Sri Lanka
+94 (0) 77 7886124
walpolahasitha@gmail.com

Guhanathan Poravi
Informatics Institute of Technology
No 57, Ramakrishna Road, Colombo 6, Sri Lanka
+94 (0) 77 342330 guhanathan.p@iit.ac.lk

***Abstract*—The automation of the front-end development process is a relatively new research area which has gained a more focus in the recent years. But the researchers have followed various approaches, differing in one another, and have found various solutions that have not been adequately documented enough to make them clear to future researches. Often the approaches used in these solution are very different, since they are not affected. If the available works is analyzed under one common domain, it will greatly assist in future research. This paper presents an analysis of the solutions presented so far under the automate the front end development process. It also describes the obstacles and setbacks that have to be confronted there. This paper is a result of initial research that is ongoing these days and in the future, using this knowledge, expect to develop an accurate tool which can fill the gaps in available solutions and add the missing features identified.**

Keywords—Automation, Front End Development, Accuracy, Domain Specific Language, code generation

I. Introduction

This section gives a brief introduction to the domain "Automate the Front-End Development" with some background and information, specifying the problem in the domain and the urgency of addressing the problem.

A. Background

By this time, making creative user-friendly web pages, user interfaces are a very common process. This process can be easily done with various accessories. Each day various types of user interfaces, web pages developed by UI developers. Often, front-end development is primarily used as an HTML markup language. Some also use the resulting templates. It makes it easy to use the UI with the template from start to finish.

If you use an already created template, you have to make an extra effort to customize it. Also, some templates do not complete our requirements properly. Therefore, in such case, from the outset, it has to be designed and developed. This process sometimes involves a large team [1]. But there is a limited number of employees in the size of a small company to meet their target goal.

This design process is perhaps a very long lasting process. As developers spend more time developing the UI, the developers will not spend a lot of time devoting themselves to implementing the function and logic [2], as well as likely to make mistakes. Therefore, a team needs to gather requirements of the client, a separate group for the design process, and the group of UI developers needed to develop the customized user interface or web page.

B. Motivation

Utilizing a finished template with a front-end development process, UI developers make day-to-day life easier. But it's not that easy to customize it for our needs, also the time it takes is considerably higher. Think that if we could make the process, which is draft of a user interface like a hand drawing(wire-frame) or image convert it to appropriate HTML page automatically we can create creative and user-friendly interfaces. Also, the developer's life makes easy. The amount of resources required also is low. Also think of a who does not know about the technology of creating websites, UI s and wants to create a small website for his/her small business. But if he/she hasn't had a budget to separate it for that, this will be very useful and valuable for a small businessman. Those people can produce very attractive user interfaces as well. This helps developers to reduce their workload and they can increase their creativity level as well.

In section 2, we present the methodology used to gather knowledge, data and statistics for the analysis required for this paper, which is an initial phase of our ongoing research. Section 3 presents in-depth details of front-end development and the problem involved. Section 4 covers the what is done previously, existing solutions proposed under the various approaches. Section 5 presents the key components in such a typical system and how the components are designed in these different approaches. In section 6 we analyze the solutions under a few common criteria. Section 7 covers the identified missing elements and in the last section, we conclude the paper by specifying the possible future works.

II. Study Setup

A literature survey was conducted to gain the necessary domain knowledge. We identified three domains that are linked with generating HTML code for given wireframe or a user interface screenshot.

1. Image Processing
2. Language Modeling
3. Generate HTML samples.

We studied around 30 research studies in this domain, and we studied the various strategies and techniques used at that time. More details about this will be discussed in section 4. Here we have identified two main solutions developed by researchers, and more details are discussed in section later chapters.

978-1-5386-8075-9/19/$31.00 ©2019 IEEE

III. FRONT END DEVELOPMENT

The front end development enables us to translate data into a graphical interface using the technology like HTML, CSS and JavaScript to view and use a data user in our possession. This creates web pages, websites. The basic foundation of all these things is HTML. It very easy task if we can get this basic HTML page automatically, which help us greatly in new alternations.

A. Auto Generate HTML Code

Why we need to automate the front-end development process is to increase the quality and productivity. To increase those things, we need lots of time, multiple numbers of employees. In below figure you can see the process of UI development. To reduce these limitations, it came up with a solution to automate the UI development process.

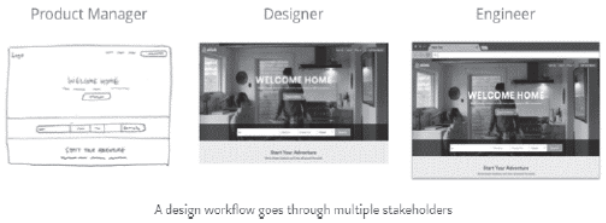


Fig. 1. Design workflow in front-end development. [1]

B. The Problem

The concept of the problem discussed in this paper is focusing on two major factors: 1.) The consuming time of implementing code [2] of GUI then 2.) Need multiple stakeholders for design process [1]. There is some work done previously to solve these problems but in these solutions also might have some limitations, accuracy problems, performance issues in the already developed tools by researchers.

IV. EXISTING WORK

The existing work on automating the front-end development process is discussing in this section.

Using machine learning, the automatic generation of code is one of the most innovative research areas. Recently it addressed the readable and program synthesis format. The research generating computer programs recently is *DeepCoder* [3] in 2016. They proposed a solution by generating code from statistical predictions and search techniques. The approach of them is training a neural network to predict the properties from the input and give the suitable output. Most of these researchers happy to use the DSL model to convert input data into output. In this research also they use a DSL. After this research, several types of researches followed their work and improved the concept of them.

```
a ← [int]                An input-output example:
b ← FILTER (<0) a        Input:
c ← MAP (*4) b           [-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]
d ← SORT c               Output:
e ← REVERSE d            [-12, -20, -32, -36, -68]
```

Fig. 2. DeepCoder input-output sample DSL**. [4]**

Terpret! [4] (2016) is one of the researches that source code generation by understating and learning the relationship between input-output examples via differentiable interpreters. They came up with their own language to convert the input into output source code.

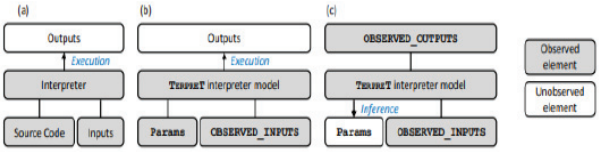


Fig. 3. Terpret high-level view of flow. [4]

Latent Predictor Networks for Code Generation! [5] (2016) by Ling able to generate the source code from a mixed natural language. They use structured program specification as input.

All of the above-mentioned researches are based on the Domain Specific Language(DSL). Computer language or a markup language specific to one domain is called as a DSL. In the application of a DSL, we will have some kind of restrictions. But using DSL can limit the programming language for modelling and reduce the capacity of search space as well.

Based on these factors, this domain is an active researcher in the field recently. Also, generate the code from the visual input is still a far less study area. But there are some researches around this as well. We discuss further more close researches about this domain from now onwards in this section.

REMAUI [6] (2015) is by Nguyen developed a method to reverse engineer Android user interface from screenshots. This research entirely uses engineering heuristics. They mainly focused on mobile application user interfaces. Big gap founded by them between the digital drawing of graphic designer and working user interface implementation. To fix that gap they introduce a technique called Reverse ***Engineer Mobile Application User Interfaces(REMAUI)*** to automatically generate the code. They able to generate user interfaces that very similar to the original design. And also it only wastes just 9 seconds.

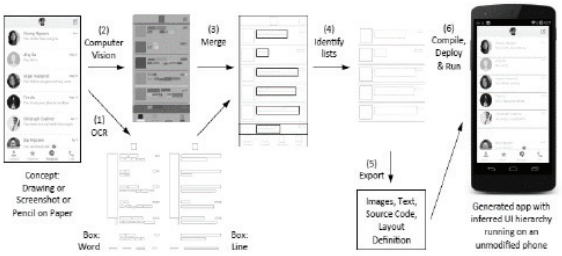


Fig. 4. REMAUI processing steps. [6]

```
<RelativeLayout <!-- List Entry ... --> >
  <ImageView <!-- Horizontal Bar ... --> />
  <ImageView android:id="@+id/ImageView_1"
    android:layout_width="59dip"
    android:layout_height="61dip"
    android:layout_marginLeft="5dip"
    android:layout_marginTop="0dip"
    android:src="@drawable/img_9"
    android:scaleType="fitXY"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"/>
  <RelativeLayout <!-- Nested: Text Block (center) ... --> >
    <TextView <!-- Sender name ... --> />
    <TextView <!-- Message ... --> />
  </RelativeLayout>
  <TextView <!-- Message date (right) ... --> />
</RelativeLayout>
```

Fig. 5. REMAUI generated layout. [6]

```
public class MainActivity extends Activity {
  //..
  private void addListView0() {
    ListView v = (ListView) findViewById(R.id.ListView_0);
    final ArrayList<ListI> values = new ArrayList<ListI>();
    values.add(new ListI(R.drawable.img_4, R.drawable.img_9, R.
        string.string_0, R.string.string_1, R.string.string_2));
    //..
  }
}
//..
```

Fig. 6. REMAUI generated android code. [6]

This has led to the identification of the nature of the input image and found several more important types of researches [7,8,9,10] about this with uses of various ways. For that, they use the concept of image captioning. In the end, all of these concepts were studied together. And we found very close two types of researches about this domain. They are more recent.

pix2code [2] (2017) is by far the most effective system in the domain along with skecthCode which is covered below. pix2code was one of the first approaches to use deep learning methods and trained a model end-to-end automatically to generate a source code from a single design input image. It is with an around 77% accuracy. Also supported for a few different platforms. They found three main problems in this domain.

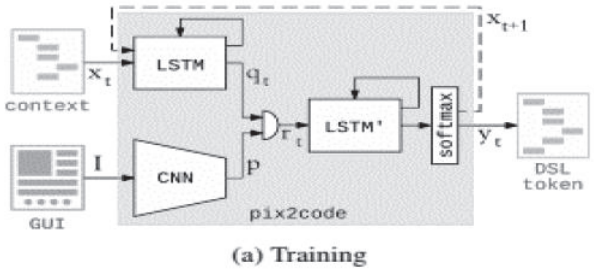
1. Understand the given image with the objects, positions (buttons, labels) with computer vision
2. Understanding the text (computer code) with language modeling
3. Generate the corresponding computer code

After identifying these problems, they use Convolutional Neural Network(CNN) method to solve the issue with the vision and they designed a simple DSL as the solution for a language model

```
stack {
  row {
    label, switch
  }
  row {
    label, btn-add
  }
  row {
    label, slider, label
  }
  row {
    img, label
  }
}
footer {
  btn-more, btn-contact, btn-search, btn-download
}
```

Fig. 7. pix2ocde sample DSL of iOS GUI. [2]

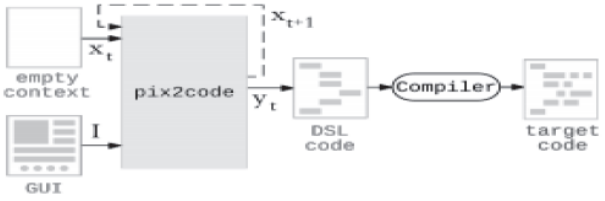
By training the model form data set and they designed a DSL which was suitable for this as below.



(a) Training

Fig. 8. pix2ocde overview of Training. [2]

After the training, they put that DSL token into pix2code compiler to generate the target code. The process of that can be shown as below.



(b) Sampling

Fig. 9. pix2ocde overview of Sampling. [2]

Following those techniques, they were able to come up with a one of the most effective solution for this. Use of these research knowledge and gather data from that sketchCode come up with a solution to generate the source code from wireframes (Hand drawing sketches). This was the latest research done in this domain.

SketchCode [1] (2018) introduced the concept of convert hand-drawn wireframes into HTML web pages using machine learning methods and image captioning techniques. It also trained the model using a dataset that used by pix2code.

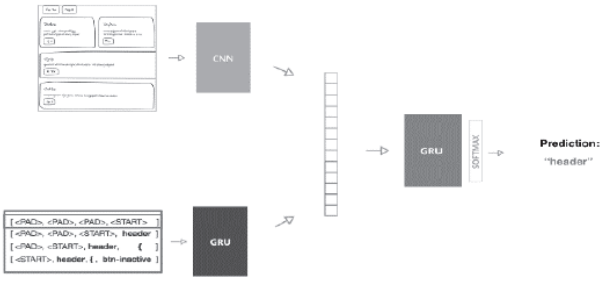


Fig. 10. SkecthCode Training the model. [1]

This research able to solve our main problem domain by accelerating the user interface design process. Both *sketchCode* and *pix2code* likely use the similar approaches. You can see the following example of *sketchCode* wireframe that converted to HTML Page.

Fig. 11. SkecthCode generated HTML code. [1]

Differences between them and which one is more accurate kind of things we discuss in section 7.

V. KEY COMPONENTS REQUIRED

This section presents the key components that need to involved in the developing system with approaches identified in the literature survey.

A.

B. Domain Specific Language

The Domain-specific language model is the main component in the system. Though various approaches used, most of use the DSL. We consider skecthcode and pix2code are the most effective and wide-applicable approach and their DSL model playas a vital role in their approaches. Also, it gives some advantages because it's relies on only that particular domain.

Advantages of DSL: -

1. Platform Independent
2. Only domain specific errors, analysis, optimizations
3. Quality and the Productivity

In these DSLs also have some disadvantages as well, but it can improve and solve those issues and can make a most effective DSL from that.

B. Image Captioning Techniques

Image processing methods are the other main part of the system. Both those research [1,2] uses most efficient techniques along with REMAUI [6] to capture the most effective components and fields of the image. After our analysis *sketchCode* proved to be a better vision model to and combing with *pix2code* will able to get very accurate results.

VI. Analysis of Existing Work

In this section, we analyze the main two solutions under the approaches used, features offered likewise things and what are the other differences with all related work.

The two main solutions mentioned here have been most similarly used. We hope to briefly discuss the other solutions mentioned above without even having a close relationship with the domain, but the approaches used in them. Otherwise, they often do not generate the code using the image as input, and they search for an asterisk that they generate a source code automatically.

In this case, a deep learning algorithm has been used for both *sketchCode* and *pix2code*. They use the image captioning concept to capture input image or hand drew wire-frame. To convert received input image into source code They used DSL. Thereafter, they train a model using a large dataset and obtain a full-blown target code.

Other solutions to generate codes automatically that they also use machine learning, but they are mostly based on the neural network's predictions.

The table 1 below contains the analysis of the features of the two major solutions. In other works, not have many features for analysis and that researches were at beginner level. So that we decide to have a comparison between *sketchCode* and *pix2code*. Below table discussed their various features.

TABLE I. COMPARISON OF FEATURES

	SketchCode	**Pix2Code**
Supported Platforms	Web	Android, iOS
Input Methods	Can input hand-drawn wireframe or image screenshot	Image screenshot
Limitations	Limit only for 16 basic HTML elements	Limited for few parameters
CSS, Bootstrap	Not supported heavily	Not supported heavily
Time is taken to convert	30 seconds	1 minute

VII. Missing Features

In this section, we will look into the findings of the survey focusing on what has been missing and the common issues that were noted in the available approaches.

Generate code for more HTML elements: The tool should be capable of generating HTML code for more HTML elements. It currently works for around 16 elements. This can be a huge issue when it comes to developing websites with forms, dropdown kind of thing. So this features missed very badly.

Support CSS and Bootstrap: The tool need to support CSS, Bootstrap things otherwise the web page generated is like a man without a cloth. This another important feature that was missed.

Navigation between web pages: When we creating a small website for small business at least 4 or 5 pages it should be there. If those pages not linked with each other pages, it will become a problem for the user of this tool. So that it great to have a feature to add navigations between given web pages when it needs.

Support in multi-platforms: The automation tool needs to work on multiple platforms. For now, some of those supports mobile and others support in the web platform. This need to be a generic tool that can generate any user interface code what user needs.

With this knowledge about the missing features, drawbacks of the available systems, and how they can be improved, we hope to come up with an architectural style in near future, towards an efficient automation tool for front-end development.

VIII. Conclusion and Future Work

Analyzing the results from the existing solutions, the following conclusions were derived.

Over the past years' front end is developed by some development cycle. In the company's product manager gather the list of specifications, then designers take those requirements creating mockups, after that engineers/ developers need to implement those designs into code. This process wastes some time; it might be several weeks. So as discussed above our goal is to automate the front-end development process and speed the process with more efficiency with more features.

In the future we will hopefully to add some functions to a generated HTML page, more elements like forms, dropdowns, images etc. more styles, more bootstraps.

References

[1] https://blog.insightdatascience.com/automated-front-end-development-using-deep-learning-3169dd086e82. [Accessed 02 August 2018]

[2] Tony Beltramelli. Generating Code from a Graphical User Interface Screenshot. arXiv preprint arXiv:1705.07962, 2017.

[3] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow. Deepcoder: Learning to write programs. arXiv preprint arXiv:1611.01989, 2016.

[4] A. L. Gaunt, M. Brockschmidt, R. Singh, N. Kushman, P. Kohli, J. Taylor, and D. Tarlow. Terpret: A probabilistic programming language for program induction. arXiv preprint arXiv:1608.04428, 2016.

[5] W. Ling, E. Grefenstette, K. M. Hermann, T. Kocisk y, A. Senior, F. Wang, and P. Blunsom. ` Latent predictor networks for code generation. arXiv preprint arXiv:1603.06744, 2016.

[6] T. A. Nguyen and C. Csallner. Reverse engineering mobile application user interfaces with remaui (t). In Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on, pages 248–259. IEEE, 2015.

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2625–2634, 2015.

[8] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3128–3137, 2015.

[9] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3156–3164, 2015.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In ICML, volume 14, pages 77–81, 2015.

[11] B. Dai, D. Lin, R. Urtasun, and S. Fidler. Towards diverse and natural image descriptions via a conditional gan. arXiv preprint arXiv:1703.06029, 2017.

[12] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. Neural computation, 12(10):2451–2471, 2000.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.

[15] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In Proceedings of The 33rd International Conference on Machine Learning, volume 3, 2016.