Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

Comparison of Graphical User Interface Testing Tools

Arnaldo Marulitua Sinaga[1)], Yohanssen Pratama[2)*], Felix Oswaldo Siburian[3)], Kevin J F Pardamaian S[4)]
[1)2)3)4)] Institut Teknologi Del, Indonesia
[1)]aldo@del.ac.id, [2)]yohanssen.pratama@del.ac.id, [3)]felixsiburian10@gmail.com, [4)]pardamaian97@gmail.com

ABSTRACT

Graphical User Interface or better known as the user interface is the liaison of users with electronic devices such as computers. The Graphical User Interface uses icons, menus, and some other visual indicators to represent the information contained in the interface of the application being used. The Graphical User Interface I must pass the Graphical User Interface Testing stage to ensure that every element in the Graphical User Interface is not an error and by the specified one. Also, we know that Graphical User Interface Testing is a set of activities that aim to test the Graphical User Interface I of the test object to ensure that the Graphical User Interface complies with the specifications specified in the software design document. In this research, we try to compare four Graphical User Interface testing tools which ae: Robotium, Espresso, UI Automator, and Pix2Code. By exploring these 4 testing tools we find out that Pix2code can only identify objects, especially label objects. Pix2code can only meet 3 out of 7 predefined criteria. This indicates that there are still many objects of the android application that Pix2code has not been able to identify. In other words in the Graphical User Interface testing section, pix2code can play a role in identifying each object contained in the application and can be done at the design stage. The result that we get from this research is that the GUI testing tools could identify many parts and almost every object in the application except the Pix2code. For future development, Pix2code as a testing tool requires development in the form of a desktop display such as the UI Automatorviewer so that it can display every detail of the object including the attributes of the object.

Keywords: Espresso; Graphical User Interface; Pix2Code; Robotium; Testing; Tools; UI Automator

INTRODUCTION

The role of mobile devices in our lives is now very important in the discussion of all our daily activities. Through mobile devices, we can now control business, to improve personal abilities, as well as to increase media. The amount of software that can be run on this device is very much as well as from mobile devices starting from software that can be very facilitated

Mobile applications are applications that are developed specifically for the use of small screens such as tablets and smartphones, rather than the use of computers. Android itself is one of the operating systems that are widely used on mobile today. In 2012, total Android smartphone users in some countries reached 31%, this percentage is far higher than other operating systems such as IOs or Symbian. In 2015, the total number of smartphones using the Android operating system reached 1.8 billion and rose to 2.7 billion at the end of 2017 (Satista, 2018). GUI testing is generally testing about ensuring that the functionality of the User Interface is correct, where an application follows the written specifications and defects can be identified. In addition, a tester must be able to design elements of an application properly. This includes checking the screen with controls such as, menu bar, toolbar, color, font, size, content, buttons, etc. including how those controls respond to user input
In Tony Beltramelli's paper (Generating Code form a Graphical User Interface Screenshot) has been explained about generating code from the Graphical User Interface screenshot and vice versa which can facilitate developers in building a user interface (Beltramelli, 2018). However, in this paper we does not just generate code but compares the code from the mockup created with the results of the application that has been built. Research conducted by Tony Beltramelli resulted in a similarity accuracy of 77%. The results of the research conducted by Tony Beltramelli will be used as applications that will be tested during the study

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

LITERATURE REVIEW

One challenge to learning about software testing is that there are many terms in the industry, and these terms often have overlapping meanings or are used inconsistently. It is important to know that UI is a broader concept which can include GUIs and Command Line Interfaces (CLIs) (Idera, 2019).

Command Line Interface (CLI) and Graphical User Interface (GUI). CLI is when the user types a text command into the terminal and the computer responds to that command, while the GUI is when the user interacts with the computer using a mouse or keyboard. This is an interaction where the user manipulates visual elements instead of using text (Karam, 2017).

The GUI was first developed at Xerox PARC by Alan Kay, Douglas Engelbart, and a group of other researchers in 1981. Then, Apple introduced the Lisa computer with a GUI on January 19, 1983. The GUI provides easy interaction with the software. GUI makes software interface easy to understand but GUI also makes it difficult for developers to develop software. Everything (functions, interfaces, etc.) must be tested, because code that is not or has not been tested has the potential for bugs (errors). In this case, GUI testing is important to make all systems that have been built safer and more robust (resistant to attacks) (Idera, 2019).

GUI testing is all about ensuring that the UI is functioning properly, that the application follows its written specifications, and that defects are identified (Inflectra, 2017). Apart from that, check that the design elements are good. This involves examining the screen with controls such as menu buttons, toolbars, colors, fonts, sizes, icons, content, buttons, etc (Karam, 2017).

To carry out a comparative study of the three Automation testing tools above, an Android application is needed as an object to be tested using these testing tools (Kowalczyk, E., Cohen, M.B., & Memon, A.M.). Therefore, the author chooses a Machine Learning-based program that will generate code from the image so that it can be used as a testing object, namely Pix2code, where the Mockup Pix2code testing process can play a role in testing the Mockup (Beltramelli, 2018).

Converting a user interface screenshot created by a Designer into program code is a problem that developers face in building a number of different software.

In the process, the client-side implementation based on the Graphical User Interface mockup created by the Designer is the responsibility of the developer. Implementing the program code from the GUI takes a lot of time and makes the developer to solve the (important) things of the software built by the developer such as the functions and logic of the software that the developer builds on. Therefore, Image to Code can be a solution for developers to minimize their time in building software.

Image to Code is an approach based on Convolutional and Recurrent Neural Networks that allows the change from a GUI screenshot to a computer code, where the screenshot from the GUI will be input. Image to Code is also an application of the Reverse Engineering user interface which comes from an image as input as shown in the image. In this case, Image to Code can provide effectiveness to developers in building software.

The task of producing computer code which is given the programming language of the image (screenshot) GUI can be compared with the problem of producing a text description of a photographic scene where the equation of these two problems is that both want to produce a string with a certain length from the pixels obtained from the image. These problems can be divided into 2 sub-problems, namely: Computer vision problems where the computer understands the given scene (in this case the GUI) and the language modeling problems from text (in this case computer code)

We use the solution to the answers to the two previous problems by using latent variables generated from the scene to generate textual descriptions (in this case computer code) of the objects represented by these variables.

The closest method in implementing Pix2code is reverse engineering where reverse engineering is a process that rearranges an artifact with the aim of seeing the details of the artifact such as the design and architecture as well as analyzing the value of the pixels that have been changed from the GUI. Basically, all user interfaces regardless of platform, and what programming language is used are composed of pixels.

Pix2code's main purpose is to generate a string of tokens of variable length of pixel value. CNN is the method chosen to solve Computer Vision problems. We can use CNN for learning where the input given is an image which will produce a vector with a certain length; so that it can act as a DSL encoder or Domain Specific Language can be used to describe the GUI as shown in Table 1.

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

In this case, the GUI is the main focus, where each GUI has different graphic components along with their relationships with other components (Memon, A., Banerjee, I., & Nagarajan, A., 2003).

The model used is trained by making Image I and a contextual sequence X from the token T xt, t ∈ {0. . .T - 1}. In Table 1. Regarding Image to Code, it has been explained that using a Convolutional Neural Network can change from a given image to a computer code (this process is called encode) (Beltramelli, 2018).

Table 1 Results of Testing Tools Table

Graphical User Interface (GUI)	DSL output generated by GCP (*Google Cloud Platform*)

	stack{
	row{
	switch
	}
	{
	,,
	,,,,,}
	footerfooterfooterbtn-dashboard,,,,

Through a literature study that has been carried out by the author, Pix2code is tasked with translating the components in the screenshot which become input to be translated into a specific programming language. Based on this, the authors observe that Pix2code plays a role in accelerating the GUI testing process as Shift Left testing. This is done by moving the test earlier in the software development life cycle. Pix2code is used in testing before the application or product has been completed, in other words, testing is carried out after the mockup is created.

METHOD

In this study, researchers will explore the potential of Pix2code as a new GUI Testing Tool by utilizing experimental results on three GUI Testing Tools where the experiments of the three GUI Testing Tools will be used to determine the properties of a GUI Testing Tools. In this study, three tools will be used to test the GUI design of an application.

1. UI Automator
2. Robotium
3. Espresso

Pix2code will identify the study object that has been determined for this study. Pix2code will generate a description of the study object design that has been determined for this study. After obtaining the identification results with Pix2code, it will be identified the shortcomings of Pix2code and compared with the properties obtained from the experimental results of the three GUI Testing Tools. At this stage, it will analyze what features can be added to make Pix2code a new GUI Testing Tools.

Furthermore, mutation testing will be carried out in which mutation testing is aimed at making a fault (faults) in the initial design of the object under study. The mutation testing applied is based on seven evaluation criteria.

At the mutation testing stage, the same thing will be done repeatedly where when mutant A is applied, object

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

identification will be carried out using testing tools and Pix2code, then the results will be compared between Pix2code and testing tools from the same mutant whether there is a missing object and whether it can identify a fault that has been created on a mutant (Guru99, 2019).

After mutation testing is carried out on the seven predetermined criteria, an experiment will be carried out on each of the predetermined criteria. The way it works at this stage is, after the object is given the first criterion treatment, it will be tested using UI Automator by utilizing UI AutomatorViewer. Furthermore, after the results of the first criterion are obtained, the modified object will be returned to the original form of the object. After the object is in a normal state, the object will be treated, namely the second criterion and will be tested using UI Automator by utilizing UI AutomatorViewer. Iteration of this method of work will be carried out up to the seventh criterion (Churaev, 2014).

After getting the results from the UI Automator, then the testing is carried out using Robotium. The way of working at this stage is to provide treatment according to predetermined criteria. The object will be given a change in the first criteria, then a class will be created to look for objects that have been given changes according to the first criteria to find out whether Robotium can recognize objects that have been changed according to predetermined criteria. After the results of the first criterion are obtained, the object will be returned to its original form. After the object is in a normal state, it will be given changes according to the next criteria. Iteration of this method of work will be carried out up to the seventh criterion.

After obtaining the results from Robotium, the next testing is carried out using Espresso. The way of working at this stage is to provide treatment according to predetermined criteria. The object will be given a change in the first criterion, then a class will be created to look for objects that have been changed according to the first criteria to find out whether Espresso can recognize objects that have been changed according to the predetermined criteria. After the results of the first criterion are obtained, the object will be returned to its original form. After the object is in a normal state, it will be given changes according to the next criteria. Iteration of this method of work will be carried out up to the seventh criterion.

Testing Tools

- Testing using UI Automator

The stages are

1. Create a new project or open a project to be tested
2. Import ui-Automator in the application module build.gradle file.
3. Making a test case where making test-cases is done by creating a new java class

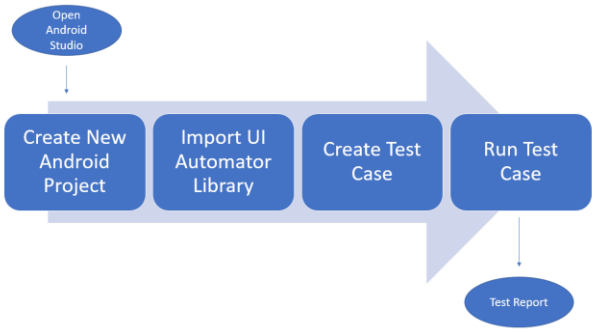


Fig. 1 Test Flow using UI Automator

-Testing using Robotium

The stages are:

1. Open the android application to be tested and add dependencies to build.gradle
2. Add a class to test the application to be tested.
3. Run Class to do the testing

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

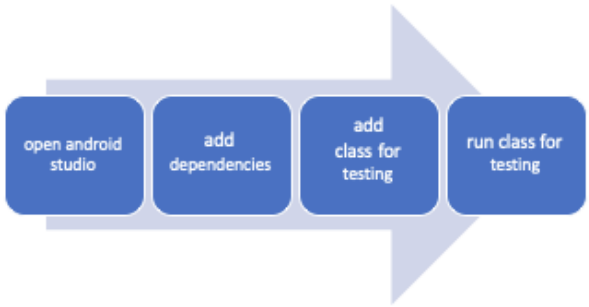


Fig. 1 Test flow using Robotium

Testing using Espresso
The stages are:

1. Add Dependencies to build.gradle
2. Click Run> Record Espresso Test.
3. In the Select Deployment Target window, select the device for which you want to record testing. If necessary, create a new Android Virtual Device. Click OK
4. Espresso Test Recorder triggers the build of the project and the application must be installed and launched before the Espresso Test Recorder allows the user to interact with it. The Record Your Test window appears after launching the application, and because the User hasn't interacted with the device, the main panel says "No events recorded yet." Interact with your device to start recording event logs such as "tap" and "type" actions.
5. After you have finished interacting with the application and added the statement, click Complete Recording. The Pick a test class name window for your test will appear. Then click Save.
6. To run the Espresso testing that has been generated, Right-click on the test and click Run 'testName.'

Observation Results
Observation results obtained, namely:

1. UI Automator: In UI Automator there is UI Automator Viewer where UI Automator Viewer has the same behavior as pix2code and UI Automator Viewer can identify all objects contained in screenshots
2. Robotium: Robotium does not find the same behavior that can be compared with pix2code, so pix2code has the opportunity to improve or support the capabilities of Robotium. The ability of pix2code to support the capabilities of Robotium is pix2code can recognize the controls that exist in the screenshots of an application where Robotium does not have that capability
3. Espresso: Espresso can do testing by running the application first and automatically generate test code so that it makes it easier to do the introduction of elements such as UI Automator Viewer x. So, based on the results of experiments conducted, then Pix2code can improve the capabilities of Robotium where Robotium cannot recognize the controls contained in an application's screenshot. This can give a new feature to Robotium, where Robotium can only search for objects in an application.

Pix2Code

The main purpose of pix2code is to generate a string of tokens along the variable value of the pixel. CNN is the method chosen to solve Computer Vision problems. We can use CNN for learning where the input given is an image that will produce a vector with a certain length; so this can act as a DSL encoder or Domain Specific Language can be used to describe the GUI.

The model used is trained by making Image I and a contextual sequence X from the token T xt, $t \in \{0. . .T - 1\}$. Regarding Image to Code, it has been explained that using Convolutional Neural Network can change from a given image to become a computer code (this process is called encode) (Beltramelli, 2018).

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

Through the study of literature that has been done by we, pix2code is tasked to translate the components of the screenshots into input to be translated into a particular programming language. Based on this, we observes that pix2code plays a role in speeding up the GUI testing process as testing Shift Left.testing is done by way of doing this is by moving testing earlier in the software development life cycle. Pix2code is used in testing before the application or product has been made, in other words testing is done after making a mockup.

Research Process

The research process is divided into several stages. The first stage begins with the identification and selection of testing criteria. At this stage, testing criteria are obtained from the results of literature studies that have been carried out such as extracting information from the website. The criteria obtained are evaluated until there are criteria that can be tested with tools.

After that, the second step is to choose the testing tools to be used. Testing tools that will be used are testing tools that are open source and can support predetermined test criteria.

The third stage is to do testing using the three predetermined testing tools. All three testing tools will be examined and analyzed regarding the behavior of each testing tools used in the fourth step, the design identification will be carried out with pix2code, this step is needed to be able to know what objects can be identified and generated by pix2code.

Next will enter the sixth stage, namely the identification of objects using selected tools. At this stage, it is identified which parts of the tool can be used to identify objects in a GUI in the application. At this stage, the role of pix2code in the GUI testing tools that has been selected and analyzed the behavior or nature of the testing tools is examined.

Test reports obtained will be analyzed, where the analysis carried out is a comparative analysis between the original application test report and the application generated using Pix2code and a comparative analysis between the results of the tests performed by each framework. Finally, conclusions will be drawn. However, before conducting the ninth stage of the research process, the writer has conducted a preprocessing stage where the writer provides information and tools used in the research process. As explained in Chapter II, we examines the usefulness of Robotium, UI Automator, and Espresso.

Hypothesis

In this study a temporary hypothesis is obtained which will refer to the conclusions that will be obtained after conducting the research to completion. The type of hypothesis used in this study is the Associative hypothesis where this hypothesis can be defined as a conjecture of the relationship between two or more variables. The hypothesis to answer the problem statement is:

1. h0: There is no relationship between pix2code and GUI testing that can lead to the role of pix2code in GUI testing
2. h1: There is a relationship between pix2code and GUI testing which can lead to the role of pix2code in GUI testing

RESULT

Results of Testing Tools Experiments

Based on experiments that have been carried out on the three automation testing tools for Android, results are obtained that illustrate the ability of the tools in conducting automation testing. The test results are obtained based on predetermined testing criteria

Table 2 Results of Testing Tools Table

Testing Tools			UI Automator	Robotium	Espresso
Code	Criteria	Testing Object			
K-1	Identification of Size Changes	Notes	Yes	Yes	Yes
K-2	Identification of	Notes	Yes	Yes	Yes

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

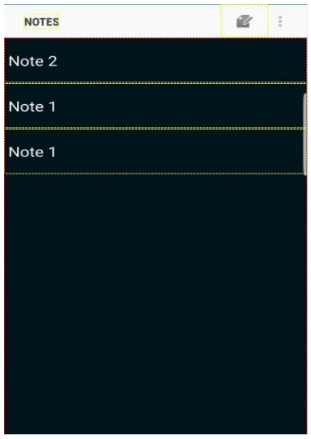
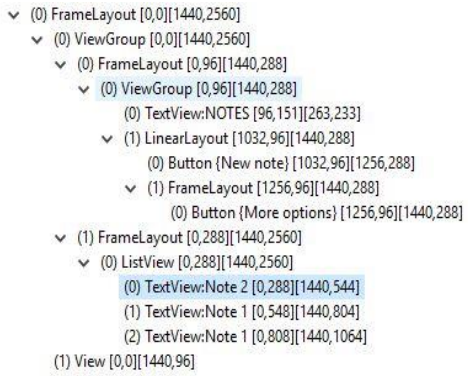
Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

	Changes in Position				
K-3	Input Field Identification	Notes	Yes	Yes	Yes
K-4	Button Identification	Notes	Yes	Yes	Yes
K-5	Identify Font Color Changes	Notes	Yes	Yes	Yes
K-6	Identify Font Type Changes	Notes	Yes	Yes	Yes
K-7	Textview Identification	Notes	Yes	Yes	Yes

From the experimental results in Table 2, it is obtained that Robotium, UI Automator Viewer and Espresso can identify each element or object contained in the application even though the object is given changes according to existing criteria (K1-K7).

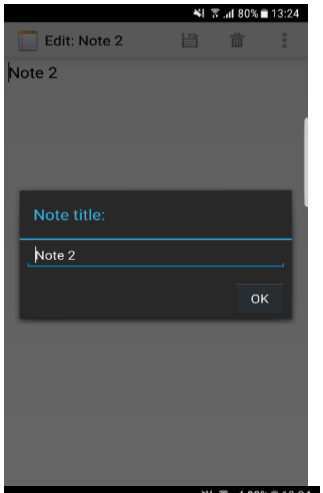
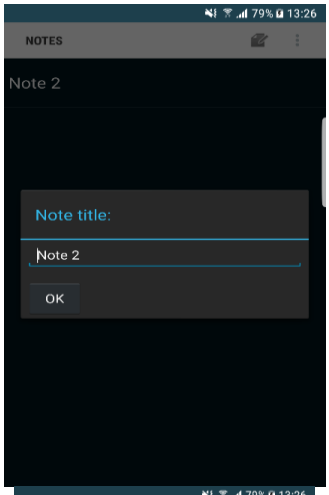
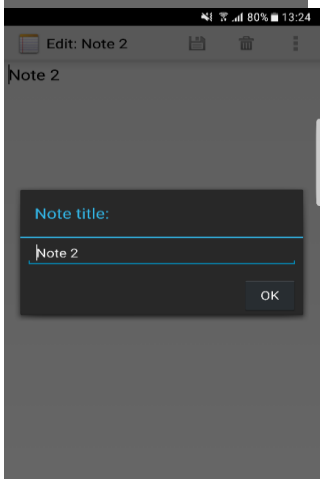
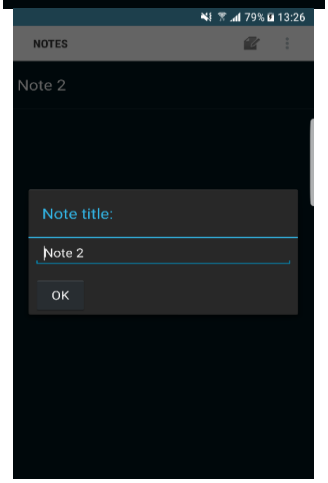
Example testing button identification by using UI Automator(K-4) can be seen in Table 3.:

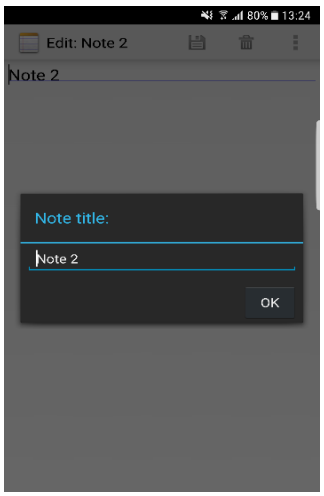
Table 3 Results of Testing Tools Table

Testing Tools	Application	Result	Status
UI Automator			**YES**

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

Robotium			**YES**
Esspreso			**YES**

Pix2Code		stack{
		row{
		switch,,,
		row{
		}
		{{
		{
		,,,,}
		{footer
		,,,,
			NO

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

DISCUSSIONS

Discussion of UI Automator

Through experiments that have been carried out, it was found that UI Automator is a GUI testing tool that focuses on Object Based Testing by utilizing UI AutomatorViewer and then a discussion will be carried out on the results that have been obtained from the exploration results of using UI Automator.

In Table 2, the results of the UI Automator exploration of the study object have been presented, where UI Automator takes advantage of one of its advantages, namely the UI AutomatorViewer which has a function to see what objects are contained in an application screenshot display. By using the UI AutomatorViewer, researchers can see what objects are in the screenshot display of an application and the position coordinates of each object in the screenshot display and with this also the predetermined criteria can be met.

Discussion of Robotium

Through experiments that have been carried out, it was found that Robotium is a GUI testing tool that focuses on Analog Testing by utilizing methods owned by Robotium and then a discussion will be carried out on the results that have been obtained from the exploration results of the use of Robotium.

In Table 2, the results of the exploration of the use of Robotium as a GUI Testing Tools have been presented for the object of study. Robotium is a GUI Testing Tool that uses a test case to test the GUI by utilizing the methods in Robotium. The result of the exploration in using Robotium as a tool for testing the GUI is that the predetermined criteria can be met by Robotium. However, there are several shortcomings of Robotium, namely that Robotium has not been able to recognize every object in an application view as can be done by UI Automator by utilizing UI AutomatorViewer.

Discussion of Espresso

Through experiments that have been carried out, it was found that Espresso is a GUI testing tool that focuses on Analog Recording and Object Based Testing by utilizing the test record and test code owned by Espresso and then discussing the results obtained from the exploration results of using Espresso. In Table 2, the results of the exploration of the use of Espresso as a GUI Testing Tool have been presented for the object of study. Espresso is a GUI Testing Tool that uses a test case to test the GUI by generating test code by running the application with an espresso record. So, when the application is run for, Espresso will record the application and Espresso will generate a test code. The result of the exploration in using Espresso as a tool for testing the GUI is that Espresso can fulfill predetermined criteria.

Discussion of Pix2code

Through the experiments that have been carried out there will be discussion related to the results that have been obtained in the use of Pix2code.

The result of Pix2code in its use is changing the screenshot of an application into a DSL (Domain Specific Language) with an accuracy of ± 77% where a value of 77% is obtained from measurements at the DSL level by comparing the expected results with the results obtained. Any length difference between each result obtained and the expected result is calculated as an error. Of the seven criteria set, there are 4 criteria that cannot be met by Pix2code and the other three criteria get a "partial" answer, which means "partial" is Pix2code can read a label on a row but on the next row, pix2code is not can read the label.

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

Table 4 Results of Pix2Code

Screenshot GUI	Hasil Pix2code

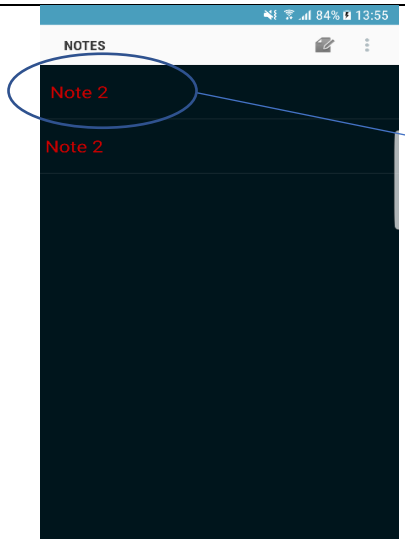
	stack{
	row{
	label,}
	rowrow
	}
	footer{
	,,,,}}
	,,

Table 4 shows that the first label is read by Pix2code and the second label is not read by Pix2code. This is due to the limited memory on the device used and the lack of minimum hardware specifications that must be used to produce the expected DSL.

That way from the results of exploration it can be said that pix2code can only identify objects. The objects that can be identified among others are the Textview that is identified as a label and the object line that is identified as a row. This indicates that there are still many objects than android applications that cannot be identified by Pix2code.

Evaluation of Results

Based on the results obtained, it was found that Robotium, UI Automator and Espresso can become GUI Testing Tools because the three Testing Tools can meet the seven predetermined criteria. Exploration and testing using these three tools to determine the properties of GUI Testing that will be used in this study.

Based on the results obtained from the exploration of Pix2code, it was found that Pix2code can only meet 3 of the 7 predetermined criteria. This proves that Pix2code can identify objects in the screenshot, while the objects that can be identified include Textview that is identified as a label and the object line that is identified as a row. This indicates that there are still many objects than android applications that cannot be identified by Pix2code.

One of the additions given to Pix2code to maximize the potential of Pix2code to become a GUI Testing Tools is the addition of features such as UI AutomatorViewer that can find out the coordinates of the position of each object in the screenshot. The addition of features to find out the coordinates of an element on the screen is to add the coordinates of the elements when doing a train to produce a model. Coordinates ranging from (0,0) to coordinates (max x, max y) will be added to each dataset to get a model that knows the coordinates of each element on the screen. Adding the coordinates is added to train.py so that the model recognizes the position of the elements on the screen. This is necessary in order to be able to test the suitability of the elements on the screen.

In the next stage, pix2code is expected to be able to identify parts of attributes such as font, color, length, width, and so on. This is useful to improve the quality of the application and test quality of the GUI testing.

Additional hardware that will improve the results of Pix2code is to use a minimum of NVIDIA Tesla 1080 11GB GPU because the minimum specification for using the GPU to train and make models is the NVIDIA Tesla 1080 GPU which takes 5 hours. By using a GPU with these specifications, the results obtained will be better and take relatively shorter time. In this study, a GPU with NVIDIA GeForce MX130 8GB specifications was used which took 8 hours to run 1 program code for training data.

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

One of the additions that need to be given to Pix2code to maximize the potential of Pix2code to become a GUI Testing Tools is to add features such as UI AutomatorViewer. From this feature, you can see the coordinates of each object in the screenshot. The addition of a feature to view the coordinates of an element on the screen is to increase the coordinates of the element when doing a train to produce a model. Coordinates ranging from (0,0) to coordinates (max x, max y) will be added to each dataset to get a model that looks at the coordinates of each element on the screen. The additional coordinates are added to train.py so that the model is in place of the elements on the screen. This is necessary in order to test the suitability of the position of the elements on the screen.

CONCLUSION

From the results of the three experimental testing tools, we can conclude that UI Automator has a feature called UI AutomatorViewer that is useful in identifying each object in the application's view and the coordinates of each object that has been captured or screenshot. We found that it has an Espresso test recorder feature that can record every step taken when running an application and generate a test code that will later be useful as an automation testing. This makes it easier to recognize and identify objects that exist in the application to be tested. But on the other side, Robotium can make a stronger test case by utilizing existing methods. Although the disadvantages of Robotium are not being able to identify any objects contained in a screenshot and can only search for an object to be searched.

By exploring these four testing tools to find out what needs to be considered in the GUI testing, note that Pix2code can only identify objects, especially label objects. In other words in the GUI testing section, pix2code can play a role in identifying each object contained in the application and can be done at the design stage. In terms of the testing process, pix2code can potentially do testing like the UI AutomatorViewer which can be used at the mockup testing stage or before development because it can identify objects from the screenshot without the required command line interface of the application. For the future development, Pix2code as a testing tool requires development in the form of a desktop display such as the UI Automatorviewer so that it can display every detail of the object including the attributes of the object. We recommend this because Pix2code has been able to identify objects from the screenshot of the application so that it is easier to develop with the UI Automatorviewer development foundation.

ACKNOWLEDGMENT

We are grateful for the Institut Teknologi Del support and all the facilities that have been provided so this research could be conduct successfully.

REFERENCES

BELTRAMELLI, T. (2018). *PIX2CODE: GENERATING CODE FROM A GRAPHICAL USER INTERFACE.*

Satista. (2018). *Installed base of smartphones by operating system from 2015 to 2017.* https:// www.statista.com/statistics/385001/smartphone-worldwide-installed-base-operating-systems/.

Churaev, E. (2014). *Automatic Android* Testing wiht UiAutomator.* Intel Developer Zone: https://software.intel.com/en-us/android/articles/automatic-android-testing-with-uiautomator.

Firly, N. (2018). *Create Your Own Android Application.* Jakarta: PT Elex Media Komputindo.

Guru99. (2019). *Mutation Testing in Software Testing: Mutant Score & Analysis Example.* https://www.guru99.com/mutation-testing.html.

H., N. S. (2014). *Android Pemrograman Aplikasi Mobile Smartphone dan Tablet PC berbasis Android.* Bandung: Informatika Bandung.

Hendradjaya, B. (ITB). *Konsep Dasar Perangkat Lunak.* Bandung: 2017.

IEEE. (n.d.). *ANSI/IEEE Standard 1059 IEEE Standard for Software Test Documentation.*

Imaduddin, A., & Permana, S. (2017). *Menjadi Android Developer Expert.* Bandung: Dicoding.

Inflectra. (2017). *What is Graphic User Interface (GUI) Testing?* https://www.inflectra.com/rapise/highlights/gui-testing.aspx.

Karam, L. (2017). *A Guide to GUI Testing.* https://dzone.com/articles/a-guide-to-gui-testing.

Kowalczyk, E., Cohen, M.B., & Memon, A.M. (n.d.). *Configurations in Android testing: they matter.* ACM.

Journal of Computer Networks, Architecture and High Performance Computing
Volume 3, Number 2, July 2021
https://doi.org/10.47709/cnahpc.v3i2.951

Submitted : 21 April 2021
Accepted : 28 April 2021
Published : 4 July 2021

Memon, A., Banerjee, I., & Nagarajan, A. (2003). *GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing.* Proceedings of the 10th Working Conference on Reverse Engineering.

Prat. (2015). *10 Best Frameworks for Mobile App Testing.* https://codecondo.com/10-best-frameworks-for-mobile-app-testing/.

Idera, R. a. (2019). *GUI Testing.* https://www.ranorex.com/resources/testing-wiki/gui-testing/.

S., S. (2018). *Mutation Testing: Testing Technique with a Simple Example.* https://www.softwaretestinghelp.com/what-is mutation-testing.

Labs, S. (2016). *MOBILE TESTING TOOLS - 5 FRAMEWORKS COMPARED.* https://saucelabs.com/resources/articles/mobile-testing-tools.

Shao, L. (2015). *Top 5 Android Testing Frameworks with Code Examples.* https://bitbar.com/top-5-android-testing-frameworks-with-examples/.

Sharma, S. (2001). *Graphical User Interface.* New Delhi: Pentagon Press.

Class, S. T. (2018). *GUI Testing in Software Testing.* http://www.softwaretestingclass.com/gui-testing-in-software-testing/.

Fundamentals, S. T. (2018). *Black Box Testing.* http://softwaretestingfundamentals.com/black-box-testing/.

Fundamentals, S. T. (2018). *White Box Testing.* http://softwaretestingfundamentals.com/white-box-testing/.

Swafford, M. (2007). *Definition and Purpose of Testing.* http://media.govtech.net/GOVTECH_WEBSITE/EVENTS/PRESENTATION_DOCS/2007/GTC_EAST/P17 System Testing .