Visual GUI Testing in Continuous Integration Environment

Fachrul Pralienka Bani Muhamad[1], Riyanarto Sarno[2], Adhatus Solichah Ahmadiyah[3], Siti Rochimah[4]
Department of Informatics Engineering
Institute Technology of Sepuluh Nopember
Surabaya, Indonesia
[1]*fachrul.pbm@gmail.com*, [2]*riyanarto@if.its.ac.id*, [3]*adhatus@if.its.ac.id*, [4]*siti@its-sby.edu*

***Abstract*—Graphical User Interface (GUI) testing which is done manually requires great effort, because it needs high precision and bunch of time to do the all scenarios repeatedly. In addition, it can be prone to errors and most of testing scenarios are not all done. To solve that problems, it is proposed automated GUI testing. The latest techniques of automated GUI testing (the 3rd generation) is through a visual approach or called by Visual GUI testing (VGT). To automate the VGT, it is necessary to use testing tools. With VGT tools, GUI testing can be performed automatically and can mimic the human behavior. However, in the software development process, VGT feedback is still not automated, so that the effort is still required to run the VGT manually and repeatedly.**

Continuous integration (CI) is a practice that can automate the build when any program code or any version of the program code is changed. Each build consists of compile, inspection program code, test, and deploy. To automate the VGT feedback, it proposed combination of CI practice and VGT practice. In this paper, the focus of research is combining and assessing the VGT tools and CI tools, because there is no research about it yet. The result of this research show that combination of Jenkins and JAutomate are the highest assessment.

Keywords—assessment; continuous integration; tools combination; visual gui testing.

I. INTRODUCTION

Graphical User Interface (GUI) is an interface program that utilizes computer graphics capabilities, so that the application easier to use [1]. Currently, about 60% of software systems have been built with a GUI-based [2]. The ease of software system operation and the natural interaction between the system with its user is one of the reason that many use of GUI [3]. Although GUI can provide an easy way for users to use the software, a GUI can make software development process becomes complicated [4], [5].

According to Sarno [6]-[9], in every process of software development life cycle, it should be assured for having the optimal granularity of software components. To improve the software quality, it needs to be tested, both at the stage after development or undergoing development. In this study, the focus of testing is GUI. GUI testing can include the functional categories or non-functional categories in four testing levels i.e. unit testing, integration testing, system testing, and acceptance testing [10]. GUI testing is categorized of testing with black-box method.

According to Isabela [11], GUI testing is the process of determining the functional requirements on the software interface whether it has met and done with the right way [12]. GUI testing includes some tasks on the application to compare the output with the expected results. To perform GUI testing, it requires test cases varies at the application interface. GUI testing can be done manually or automatically by using tools.

In software development process, GUI testing is generally done manually, thus requiring a high degree of accuracy and a bunch of time. In addition, it can be very prone to errors, and most of testing scenarios are not all done [13]. To overcome that problems, the automation of GUI testing is proposed as one solution to accelerate testing, perform routine testing, and facilitate the creation of test cases [14]-[17].

There are three approaches (techniques) in the automated GUI testing [18]. The first technique is automated GUI testing through the coordinates. In this technique, the test is highly dependent on the location and position of the GUI. This technique has been abandoned by developer and tester, because it takes a great effort when there is a change in the position or layout of the GUI.

The second technique is automated GUI testing through the architecture. This technique is still used by developer and tester by utilizing the library, program code, and application interfaces. Disadvantages of the second generation technique are highly dependent on GUI technologies, programming languages, and specific platform.

The third technique is automated GUI testing through the visualization of the application. This technique is done by utilizing the image recognition on GUI display. By doing so, GUI testing activities can mimic human behavior. In each test automation techniques, it needs to take an advantages of certain testing tools. However, the feedback of testing tools is still not automated, so that the effort is still required to run the tools manually and repeatedly. In this paper, the focus of research is GUI testing with a third technique, also called visual GUI testing (VGT).

According to Fewster et al. [19], the team that implements automated testing practices need to perform regression testing when the latest version of software was created or the system software has been changed. In addition, the team needs to run the test more often and need to reuse each of test cases, so that the success and quality of the software can be increased.

Continuous integration (CI) is a practice that can automate the build when the source code is changed [20]. Each build consists of compile, testing, program code inspections, and deploy. Automated build practice is run by using the continuous integration tools. The benefit of continuous integration practices i.e. reducing the chances of software

978-1-5090-1381-4/16/$31.00 ©2016 IEEE

failure risks, reducing repetitive processes, produces software products that ready to deploy, provide better project visibility, and increase confidence in the team [21]. According to Muhonen [22], continuous integration techniques can be used to automatically integrate functional testing in order to obtain rapid feedback on the software development process.

Based on the exposures above, there is a gap between VGT practices and CI practices. Besides, there is no research which combining and assessing the VGT tool and CI tools. In previous research, there are papers that discuss about combination of software testing tools and CI tools. However, the focus of previous research is combination between performance testing tools and CI tools [23], also security testing tools and CI tools [22]. To resolve that gaps, this research was proposed to combine the VGT and CI. In this paper, tools of VGT and CI are combined and assessed for determining which combination are the most well.

The sections arrangement of this paper are described as follows: section 2 is described about the definition of CI and its model, build process of CI, benefits of CI, and VGT model. Section 3 is described about related work that done for combining the software testing tools and CI tools. Section 4 is presented the proposed work which consists of the overview of tools that want to combined. Section 5 is explained about the result of experiment and further discussion. The final section, it concluded the experiment result.

II. LITERATURE STUDY

A. Continuous integration (CI)

CI is a process of software development which is automating the build when there is any change of program code or any new version of software [20]. The build is triggered in separated machine when the developer changes the program. The build also can be scheduled by project teams in every night. Each of build consist of several step i.e. compile, test, code inspection, and deploy. After the build finish, the build reports will be sent to specific project member.

According to Martin Fowler [24], CI is a practice of software development where the team members integrating their work as often as possible, usually the team members integrating their works, daily. Each of integration are verified by automated build which consist of testing which help identifying the failures of integration in early. Continuous integration practice is considered by many teams that it can reduce the problem of software integration significantly. Integration is a one of the last phase in software development projects where all of the different unit are merge together and put under integration test for verifying the unit whether they interacted each other as planned [24]. The model of continuous integration can be seen in Fig. 1.

B. Build process of CI

This section is described about the build process in CI practices [24].

1) *Repository of program code.* Program code repository are maintained by using the version control system tools that can help the developers to commit the change of program code, revert to the previous version of program code, and merging the line of codes which is conflict.
2) *Automated build.* All of the build process should be done in automatic way, with a simple process, which is not required the user interaction.
3) *Automated testing.* The build process should consist of testing that utilize a unit testing set.
4) *Code commit.* The program code should be committed by each of developer to the repository of main every day after they made change.
5) *The time of build.* For the effectiveness of the build process, it should all done automatically and does not require a long time. CI build should not be longer than 10 minutes [25].
6) *Feedback.* In the end, it is important for all team members for getting feedback from integration process.

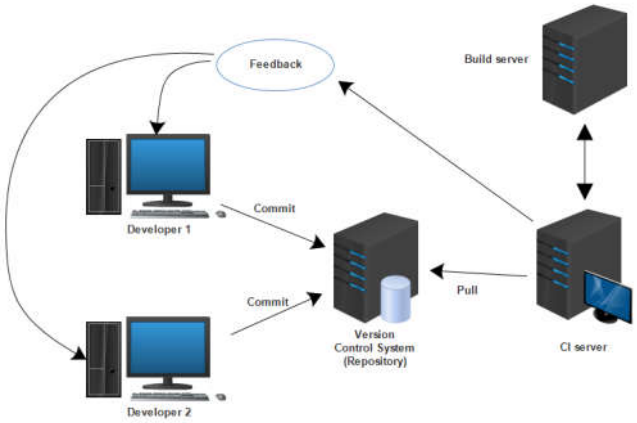

Fig. 1. The model of continuous integration

C. Benefits of implementing CI

Software integration is not a problem when only done by one developer. But in several developers or team, which work in one project together, integration of software can cause a problem, because of the piece of code changed by several developers which worked together. The following are some benefits of implementing CI which is described by Duval [21].

1) *Reduce risks.* The chances of integration failures can be reduced. The problem can be notified early and more often when it was just appearing.
2) *Reduce repetitive processes.* Build process on CI practice may consist of compiling, testing, code inspection, and deploy, which all done automatically. So it can save the time, cost, and effort.
3) *Generate deployable software.* The one of purposes in software development of agile is software deployed as soon as possible and more often. The practice of CI can support for automating the deploy.

4) *Enable better project visibility.* CI that run often can provide the ability for getting the latest actual information about the health of software.
5) *Greater product confidence.* With using CI, team projects will know that any changes may affect the code base. So that it can increase the confidence of the team.

D. Visual GUI testing (VGT)

VGT is automated test technique which utilizing the tools for recognizing the image, interacting the GUI and asserting the expected output. There are available VGT tools, both of commercial e.g. JAutomate and open source e.g. SikuliX. Each of tools has advantages and disadvantages. However, most of the tools used image recognition for controlling the script that allows them to be used in most GUI applications. As a consequence, VGT associated with high flexibility. There are limitations of this technique that cannot be used on non-GUI applications such as an application server.

VGT script can be written or recorded based on the scenario that consists of a method which is usually done as a human interaction in the system under test, for example the mouse and keyboard events. That image is used by tools to recognize and simulate the behavior through the GUI image. In general, the model of VGT can be seen in Fig. 2.

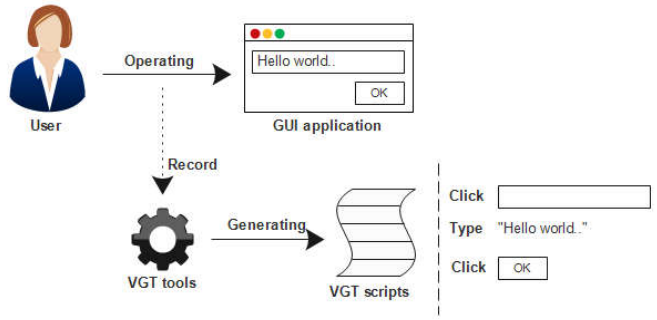

Fig. 2. The model of visual GUI testing

III. Related Work

Currently, research on continuous integration are growing up. Now the stage of testing on continuous integration is not only unit testing anymore, but also include any testing on other levels. Based on research conducted Chris et al. [23], continuous integration combined with performance testing.

Research methodology of Chris et al. is assessing some performance testing tools with specific criteria. That assessment consists of nine criteria, i.e. documentation, community, integration into the CI platform, learnability, likeability, maintenance, software cost, support, and usability [23]. After determining the ninth assessment criteria of some performance testing tools, Chris et al. conducted experiments on a combination of performance testing tools to several CI tools and give assessment to performance testing tools. At the end of the study, it proposed the best combination between performance testing tools and CI tools based on nine assessment criteria.

Besides, there is a research that conducted by Muhonen that combining CI with security testing [22]. The research methodology of Muhonen is improving the software process with executing the commit build first, then executing the functional build, after that executing the security testing, and the last is publishing vulnerability reports. In each process, it is described the model of experiment and its build scripts. To automating the security testing, it utilizing acunetix vulnerability scanner tools. After that, it combined with CI tools, atlassian bamboo.

IV. Proposed Work

This section is described about the overview of VGT tools and CI tools. The way of using VGT tools and the expandability of CI tools are explained to represent the hypothesis that any tools of VGT and CI can be combined. The summary of this overview can be shown in Table 1.

TABLE I. Overview of VGT Tools and CI Tools

Tools	Developer	OS	License	Website
SikuliX	Community	Cross-platform	Open source	Sikuli.org
JAutomate	Swifting AB	Cross-platform	Proprietary	JAutomate.com
Bamboo	Atlassian	Cross-platform	Proprietary	Atlassian.com
Jenkins	Jenkins community	Cross-platform	Open source	Jenkins-ci.org

A. VGT tools overview

In this sub section, it also described the case study and the scenario of using VGT tools.

1) *Case study*
The case study that used in creation of test scenarios is medrecapp. Medrecapp is a simple medical record application in undergoing development, which is made by author as a combination scenario in VGT and CI practices. This application consists of several menus and submenus for giving the example of automating the GUI testing in undergoing development. The project of application can be downloaded from GitHub repository.
2) *Scenario*
Each step in this scenario is created based on visual medrecapp. The scenario used for evaluating the VGT tools. The steps of scenario are as follows: Click the master menu, click submenu specialist, fill the textfield, click insert button, match the success confirmation dialog.
3) *SikuliX*
SikuliX is an open source VGT tools. It can be downloaded from SikuliX website, and it website provided the documentation and forum for asking questions. Based on the scenario, SikuliX did not record and replay any activity in the application. But this tool provides features to take pictures on the screen and select the operation based on that picture. VGT source code can be downloaded from GitHub. Based on the scenario, the overall steps of SikuliX is easy enough

and SikuliX have a good support with complete documentation.

4) *JAutomate*
JAutomate is a semi-paid VGT tools. It can be used with free of charge, but just in certain features. Additional features of JAutomate, require payment in advance. JAutomate display is quite nice, and it support record and replay. Based on the test scenarios, developers who use SikuliX need to take steps more practical than JAutomate, because JAutomate provide record and replay feature. Overall, JAutomate have fun features for testing and it have clear documentation.

B. *CI tools overview*

This overview is done on a windows environment. it used Windows 10 operating system. Plugin of CI tools are overviewed whether VGT tools supported. The CI tools which is overviewed are Jenkins and Bamboo.

1) *Jenkins*
Jenkins is an open source CI tools. It can be installed in a local network or hosted to the Internet. There is no available Jenkins plugin of SikuliX and JAutomate. Jenkins by default does not support GUI testing, but this tool provides a Xvfb plugin which allow GUI testing in Jenkins. However, there is no support windows environment to use Xvfb, so it is not possible to run SikuliX and JAutomate. There are several guides on the internet about GUI testing with Jenkins in windows environment. To overcome this problem, it needs deep investigation for implementing GUI testing in Jenkins with windows environment.

2) *Bamboo*
Bamboo is one of Atlassian products that support CI. It can be installed on a local machine or over the Internet. These tools are semi-paid, because it provides a trial in a few days. For GUI testing, this tool does not provide a plugin for using tools of SikuliX and JAutomate. But, there are several guides on the internet which makes Bamboo possible to perform GUI testing. It needs a deep investigation for implementing GUI testing in Bamboo with windows environment.

As a general, overview of the steps in proposed work can be seen in Fig 3. After the VGT and CI tools are investigated, then both of tools are combined. Any combination is assessed based on the assessment criteria of research which is done by Chris et al [23].

V. RESULT AND DISCUSSION

This section is described about the results of experiments in combination of VGT and CI tools. The result of a combination of VGT and CI are discussed further.

A. *Jenkins and SikuliX*

In this combination, the authors conducted an experiment in two ways. The first way is to write test code through the unit testing. Code is executed locally with build tools first, until SikuliX done the VGT. If the local build success, then the code is pushed to repository. After that, Jekins will pull the code on repository and build in server machine which consist of automated VGT.

The second way is through the command line which triggered by Jenkins. In this step, code of unit testing is not required. This combination need to deep configure the Jenkins for allowing the GUI testing.

B. *Jenkins and JAutomate*

JAutomate provide code version. JAutomate can be integrated with build tools through the JUnit with including the JAutomate library on the project. It required developers to create VGT scenario through the unit testing. For integrating with Jenkins, it required deep investigation about GUI testing in Jenkins with windows environment.

C. *Bamboo and SikuliX*

Since bamboo does not provide a plugin for SikuliX, it would require further investigation regarding the combination of these tools. In some internet sources said about the possibility of testing GUI. To integrate both tools, author needs to create repository trigger in Bamboo and push the code into repository so that Bamboo can pull the code and build the code.

D. *Bamboo and JAutomate*

It similar with the combination of Bamboo and SikuliX, it needs deep configuration in Bamboo for implementing GUI testing. This build process is possible through the unit testing and direct from JAutomate source script.

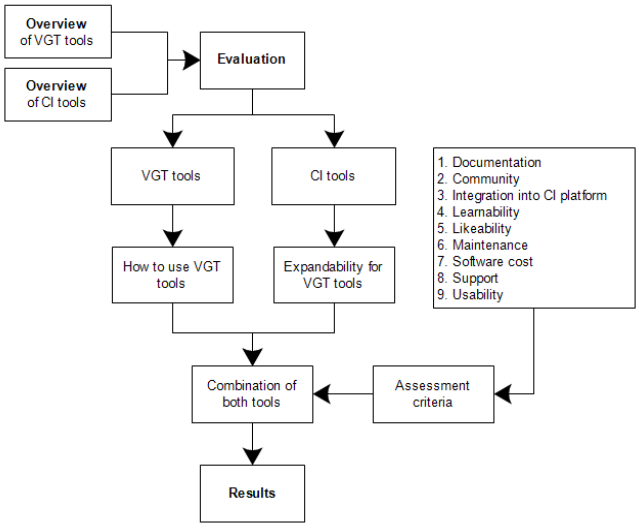

Fig. 3. Proposed work model in this research

The summary of experiments can be seen in Table III. It consists of assessment which conducted from Chris et al research's [23]. There are nine assessment criteria and each of it, may get a positive (+), neutral (.), or negative (-) value. Positive value is worth two points, neutral value is worth one point, and negative value is worth zero point. The detail of assessment can be seen in Table II [23].

TABLE II. ASSESSMENT CRITERIA AND VALUE OPTIONS

Assessment Criteria	Description	Value Options		
		Positive (+)	*Neutral (.)*	*Negative (-)*
Documentation	Whether the documentations tools allow to be used extensively	The products provide extensive documentation	The products provide a brief documentation	The product does not provide clear documentation that does not even exist
Community	Whether there are community for supporting the tools	There are blogs that review products or there are any questions on the stack overflow	Owner products on a site that has features help to frequently asked questions, so as to create a community	No communities established for sharing product information and product owner does not care
Integration with CI platform	Whether VGT tools easily integrate with CI tools	Integration with clear documentation and without the hassle	Successful integration after investigation	The integration looks like impossible to do
Learnability	How easy VGT tools to be learn	Efficient and error-free interaction with the product	It's hard to learn but could eventually be used	Product is difficult to learn
Likeability	Whether VGT tools favored	Fun to work with these products	Good product when used but nothing special	Product annoying and unfit for use
Maintenance	Whether tools are maintenance by the tools owner	There is the latest stable version and subsequent versions are still being developed	There is a version that can still be used and there is support bug fix	A version that can be used without the support bug fix
Software cost	Whether tools can be used in a free or paid license	An open source license and is compatible with CI tools	Prices of products announced	Prices of products are only available upon request
Support	Whether possible to receive at least basic support	The product owner provides free basic support such as forums, mailing lists, or wiki	Only available support paid	There is no support offered
Usability	Whether developer can complete the task by using the tools without any answered question	The principles are most common in interaction design can be understood	Only some of the principles of interaction design that can be understood	There are no design interaction principles that can be understood

TABLE III. SUMMARY OF VGT - CI COMBINATIONS

No	Criteria	Jenkins		Bamboo	
		SikuliX	*JAutomate*	*SikuliX*	*JAutomate*
1	*Documentation*	(+) There are several documentation to combine SikuliX and Jenkins	(+) There is available documentation of combining JAutomate with Jenkins	(.) There are several information, but nothing for guiding the user	(-) There's no documentation of this combination
2	*Community*	(+) Several blog discuss the combination of both tools	(+)Several blog discuss the combination of both tools	(-) There is no community which discuss this combination	(-) There is no community which discuss this combination
3	*Integration with CI platform*	(-) There is no plugin of SikuliX in Jenkins	(-) There is no plugin of JAutomate in Jenkins	(-) There is no plugin of SikuliX in Bamboo	(-) There is no plugin of JAutomate in Bamboo
4	*Learnability*	(.) Easy to use, but need to further investigation	(+) Easy to use and configure	(.) Easy to use, but need to further investigation	(.) Easy to use, but need to further investigation
5	*Likeability*	(.) There is no record & replay features, but it still understand to use	(+) Record & replay is features that make user easy to operate without any question	(.) There is no record & replay features, but it still understand to use	(+) Record & replay is features that make user easy to operate without any question
6	*Maintenance*	(+) Both of tools are maintenanced by their owner	(+) Both of tools are maintenanced by their owner	(+) Both of tools are maintenanced by their owner	(+) Both of tools are maintenanced by their owner
7	*Software cost*	(+) Both of tools are opensource	(.) JAutomate need to licensed if user want any other features	(.) Although SikuliX is opensource, Bamboo need to be licensed if we want to use more than 30 days	(-) Both of tools should to be licensed for futher experiment
8	*Support*	(+) This combination is supported by tools owner	(+) It possible to get support from both of tools	(-) The support is possible for the licensed only	(-) The support is possible for the licensed only
9	*Usability*	(.) In several design makes author wondering before further investigation	(+) The design is clearly understand to use	(.) It's not easy to configure in first experiment	(.) In several design makes author wondering before further investigation
Total score		13 points	15 points	7 points	6 points

VI. CONCLUSSION

The feedback of VGT is still not automated before it combined with CI. Developer should run the VGT tools repeatedly and manually which requires a lot of efforts. There is no research about the combination of VGT and CI, and the assessment of VGT and CI yet. However, the previous research is combination of performance testing and CI, also security testing and CI.

In this paper, both of tools are overviewed and investigated first before it combined and assessed. The way of using VGT tools and the expandability of CI tools are investigated. After that, the next step is conducting the research method of Chris et al. about assessment for combining tools. Each of two VGT tools and two CI tools are combined and assessed with nine criteria.

The proposed method is done for determining the best combination of VGT tools and CI tools. Based on the exposure of tools and the combination, the result show Jenkins and JAutomate are the highest assessment. It because JAutomate design is understandable and more practical. In addition, Jenkins supported by many communities that provide many solutions. With the same combination tools in Linux operating system, it is easier to be combined, because it supported by available Jenkins plugin.

REFERENCES

[1] X. Yuan, M. B. Cohen, and A. M. Memon, "GUI interaction testing: Incorporating event context," *IEEE Trans. Softw. Eng.*, vol. 37, no. 4, pp. 559–574, 2011.

[2] P. Brooks, B. Robinson, and A. M. Memon, "An initial characterization of industrial graphical user interface systems," *Proc. - 2nd Int. Conf. Softw. Testing, Verif. Validation, ICST 2009*, pp. 11–20, 2009.

[3] A. Issa, J. Sillito, and V. Garousi, "Visual testing of Graphical User Interfaces: An exploratory study towards systematic definitions and approaches," *Proc. IEEE Int. Symp. Web Syst. Evol. WSE*, pp. 11–15, 2012.

[4] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Hierarchical GUI test case generation using automated planning," *IEEE Trans. Softw. Eng.*, vol. 27, no. 2, pp. 144–155, 2001.

[5] E. Alegroth, Z. Gao, R. Oliveira, and A. Memon, "Conceptualization and Evaluation of Component-Based Testing Unified with Visual GUI Testing: An Empirical Study," *2015 IEEE 8th Int. Conf. Softw. Testing, Verif. Valid.*, pp. 1–10, 2015.

[6] E. S. Pane and R. Sarno, "Capability Maturity Model Integration (CMMI) for Optimizing Object-Oriented Analysis and Design (OOAD)," *Procedia Comput. Sci.*, vol. 72, no. Cmmi, pp. 40–48, 2015.

[7] R. Sarno, H. Ginardi, E. W. Pamungkas, and D. Sunaryono, "Clustering of ERP business process fragments," *Proceeding - 2013 Int. Conf. Comput. Control. Informatics Its Appl.* "Recent Challenges Comput. Control Informatics", IC3INA 2013, pp. 319–324, 2013.

[8] R. Sarno, P. L. I. Sari, H. Ginardi, D. Sunaryono, and I. Mukhlash, "Decision mining for multi choice workflow patterns," *Proceeding - 2013 Int. Conf. Comput. Control. Informatics Its Appl.* "Recent Challenges Comput. Control Informatics", IC3INA 2013, no. 2007, pp. 337–342, 2013.

[9] N. Y. Setiawan, R. Sarno, "Multi-Criteria Decision Making for Selecting Semantic Web Service Considering Variability and Complexity Trade-Off," *Journal of Theoretical & Applied Information Technology* 86 (2), pp. 316-326, 2016.

[10] A. Seppala, "Improving software quality with Continuous Integration," M.S. thesis, Dept. Comp. Sci. Eng., Aalto Univ., Espoo, Finland, 2010.

[11] Isabella and E. Retna, "Study paper on test case generation for GUI based testing" *Int. J. Softw. Eng. Appl.*, vol. 3, no. 1, pp. 139–147, 2012.

[12] D. Siahaan, "Analisa Kebutuhan dalam Rekayasa Perangkat Lunak (Requirements Analysis in Software Engineering)," Yogyakarta: Andi Offset, 2012.

[13] I. Jovanovic, "Software Testing Methods and Techniques," *IPSI BgD Trans. Internet Res.*, vol. 5, no. 1, pp. 30–41, 2009.

[14] R. Feldt and S. Engineering, "Automated System Testing using Visual GUI Testing Tools: A Comparative Study in Industry," *ICST '12 Proc. 2012 IEEE Fifth Int. Conf. Softw. Testing, Verif. Valid.*, pp. 350–359, 2012.

[15] M. Finsterwalder, "Automating acceptance tests for GUI applications in an extreme programming environment," *... 2nd Int. Conf. ...*, 2001.

[16] P. J. D.Elfriede, R.Jeff, "Automated software testing : introduction, management, and performance," Addison-Wesley Professional, 1999.

[17] M. Grechanik, Q. Xie, and C. Fu, "Experimental assessment of manual versus tool-based maintenance of GUI-directed test scripts," *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 9–18, 2009.

[18] E. Alegroth, "Visual GUI Testing: Automating High-Level Software Testing in Industrial Practice," M.S. thesis, Dept. Comp. Sci. Eng., Chalmers Univ, Goteborg Univ, Goteborg, Sweden, 2015.

[19] G. D. Fewster, Mark, "Software test automation: effective use of test execution tools," Addison-Wesley Publishing, 1999.

[20] S. F. Ahmad, "Test Driven Development with Continuous Integration: A," *Int. driven Dev. with Contin. Integr. a Lit. Rev.*, vol. 2, no. 3, pp. 281–285, 2013.

[21] G. A. Duvall, M. Paul, M. Steve, "Continuous integration: improving software quality and reducing risk," Pearson Education, 2007.

[22] M. Muhonen, "Software Process Improvement: Continuous Integration and Testing for Web Application Development," M.S. thesis, Dept. Comp. Sci., Tampere Univ, Tampere, Finland, 2009.

[23] C. Geiger, D. Przytarski, and S. Thullner, "Performance Testing in Continuous Integration Environments," M.S. thesis, Dept. Soft. Tech., Stuttgart Univ, Stuttgart, Germany, 2014.

[24] M. Fowler and M. Foemmel, "Continuous integration," *(Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, 2006. [Online]. Available: https://www.thoughtworks.com/continuous-integration. [Accessed: 01-Jun-2016].

[25] K. Beck, "Extreme Programming Explained, Second Edition," Addison-Wesley Professional, 2004.