International Journal of Engineering & Technology, 7 (4) (2018) 3777-3781

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet. v7i4.15081
Research paper

Automation link checker testing with continuous integration

Joe Erik Carla Wijaya [1], Abba Suganda Girsang [1*]

[1] Computer Science Department, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480
**Corresponding author E-mail: agirsang@binus.edu*

Abstract

Many links from websites are damaged due to the development of an increasingly large and complex website features. To ensure that no links are broken there is a need for a link checker testing application. But manual testing requires a lot of resources, expensive cost, and takes a long time to check the whole deployment must be done every day, so in this study the manufacture of link checker testing application will run automatically and integrated with Jenkins. With this application, developers can fix broken links in the development stage before being consumed by the public. Automation link checker testing application is using Jenkins as continuous integration tools, Shell Script, PHP Unit framework and PHP programming language will check all the links in the website automatically and report from the test results. Once compared with manual link checking, the test automation application of this link checker is more accurate and faster than checking links manually.

Keywords: *Automation Testing; Testing; Link Checker; Continuous Integration; Jenkins.*

1. Introduction

Software can be interpreted as a collection of instructions that when executed will create a computer program [1]. The development of increasingly large and increasingly complex software resulted in the importance of a software that has good quality.
Website applications are becoming increasingly more complex and important for the company. At the development stage requires an approach by using support tools, where truth and reliability are crucial to the success of a business and an organization [2].
Writing and executing tests for manual software testing is a very tedious and very expensive task. The report gets an annual cost of infrastructure for software testing ranging from 22.2 million dollars to 59.5 million dollars [3].
Software testing is a key technique for ensuring quality and bug findings. It is generally very difficult and time consuming. Websites are getting more complex, testing is less frequent and even bypassed by testers. Test automation can avoid that. Test automation is the use of testing tools and reduces human involvement, repetition or excessive tasks [4].
A bad interface will make the user uncomfortable using it. User interfaces (such as broken links, color combinations, and navigation) that indirectly affect a website's performance. A good user interface can improve usability which means ease of user to use the app [5].
Continuous integration helps the project release more frequently and many projects use continuous integration to develop its products [6]. From a survey of 523 software developers worldwide, 78% of developers feel more productive while using continuous integration, and 85% of developers think that continuous integration plays an important role in test automation [7].
Continuous integration is not enough to support the development of the project. Developers will need a software project ecosystem, such as Github that is integrated with continuous integration in the automation process, gathering data on various factors that can affect productivity and quality. One of the influencing factors is data history. Where can be observed the quality of the code of each process [8].
In a study conducted by Glenn A. Stout, the Senior Functional Specialist, The Revere Group, pointed out sites that have broken links are amazing, and even very important to online businesses. The study also showed that when found fault on the e-commerce website, 28% of people stopped shopping on the website, 23% stopped buying from the website, and 6% people were so angry, so they stopped buying on e-commerce website [9]. We can only guess whether customers feel there cannot provide quality website sites, they may not be able to sell quality products from their stores.
The author also analyzes the links on media websites that have a total of 475M + monthly visits. In February, the total number of broken links on the website was 46.26% of the total links on the website.
Based on the above description, the authors make an automation link checker testing application with Jenkins as a continuous integration tool. The contribution of this paper is to hasten the broken link in the development stage before entering the deployment stage.

2. Related work

2.1. Broken link

Hypertext Transafer Protocol (HTTP) is the standard method for transmitting information over the internet. An HTTP status code is returned by the server to the client in response. Some HTTP status codes are 1xx (information), 2xx (successful), 3xx (redirection), 4xx (client error), 5xx (server error) [10]. Hyperlinks, commonly known as links, provide a means to access between documents. Hyperlinks often appear as underline or highlighted text but can be found also in the form of images. There are several types of HTML hyperlinks, and each is used in different situations [11].

Copyright © 2018 Joe Erik Carla Wijaya, Abba Suganda Girsang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Absolute URL: the link address that must be written in full. An example of an absolute url google .
Relative URL: links to pages with the same web server, so it does not require a complete link address. An example of a relative url home
Linking within a document: link to search documents in one website. Examples of linking within a document Chapter2

2.2. Broken link

In a website, broken links are a problem of navigation problems. Broken links can result in various levels of the structure of a website. Most of these Internet users experience a "file not found" problem in a website that is generally represented with a 404 as response status, this error message indicates the name of the server exists, but the website page can not be found in a particular location. The cause is [11]:

- The website page is moved to a new page. In this case, the user can try to search the new page.
- Syntax error in link.
- Change the name of the website page.
- Removing website pages from website.

One of the most important indicators of a site that has good quality is the absence of broken links. The following disadvantages of broken links are:

- The bad thing for the user is when the user opens a link, but the link leads to a blank page.
- Broken links will damage the reputation of the site and lower the rating on search engines.
- The rating of the search engines will decrease if it finds any broken links on the website page

2.3. Automated testing

Testing can automatically reduce or eliminate the cost of testing. A computer can follow a sequence that is memorized faster than from a human [12].
Automating software testing involves coding code using several programming languages such as Phyton, Javascript, or Tcl (Tool Command Language), so that testing can be executed by the computer. Using a test automation tools it is possible to record the test and be able to repeat it if necessary. The purpose of automation testing is to reduce the number of manually executed case cases [13].
Advantages of automation testing:
Faster than manual testing.
Costs are more effective: Testing is executed using automation tools so that testers are less needed in automation testing.
Repeats: The same case test (recording and rewinding) can be re-executed using the testing tool.
Reusable: Tests can be reused with different versions of software.
Programmable: Testers can create sophisticated testing to find hidden information.
Comprehensive: Testers can build several test cases that include every feature in the software.
More reliable: Automation testing will do the same thing every time it runs without anyone being missed.
Scope of testing: wider coverage testing of application features.

2.4. Continuous integration

Continuous Integration (CI) is a software development practice where team members integrate their work periodically, including compiling, deploying, and getting feedback. CI is useful to deal with risks more quickly and gradually. [14]
CI can solve problems in terms of maintaining quality and accuracy through various combinations to build software and testing to verify that all compilers are supported, a combination of software, the development of each software and the generality of a test. CI is adapted to create a software and test it from a composite of all developer's code tested into one testing place, so that it can tell the error. The principles of continuous integration are maintain one code repository, create an automation software, make the own test, everyone can put the code into the source code repository as often as possible, each code change must pass the test within CI, build software quickly, testing in cloning environment production, simplify the distribution of execution, developing software should communicate and collaborate with each other, automation deployment [15].

2.5. Jenkins

Jenkins, originally named Hudson, is a Continuing Integration tool with open-ended copyrights known as open source written in Java. Jenkins is used by groups with as many members as any, for projects with multiple languages and technologies. Jenkins can be given an instruction in accordance with the wishes of the developer, so as to provide reports if the project passed or failed, and can display the cause of the failure.
Installing and managing servers requires little or no time with the IT department, within 30 minutes of initial setup. Jenkins works with various version control systems and can integrate with each other [16].

3. Proposed method

Automation link checker testing with continuous integration is a website-based application using Jenkins as a continuous integration server. Jenkins is configured to run automatically updates the source code if there is a change from the github repository, and rebuilds from the updated code. After the code is completely rebuilt, Jenkins will also be configured to perform link checker testing. After the test is complete, the test results will appear whether the test is successful or failed. If it fails, it will show up a log link anywhere error with the supporting information. Inside Jenkins, there is also a history of every software built and tested. Thus, the developers can find out whether the code created and already in the push into the github repository will create a link-link in the website is broken or not.

3.1. Flow of automation link checker testing

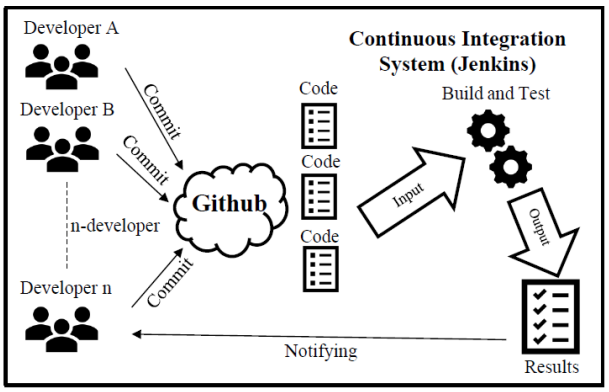


Fig. 1: Flow of Automation Link Checker Testing Application.

The link checker software in this research will run automatically if there is a code change inside the github repository. Therefore, to automate the link checker testing will use Jenkins as a continuous integration tool. For the architectural design of the system to be made in this study will be described in Fig 1. Fig 1 illustrates the developers who are in a group and work on the same project. Developer will enter the code that has been made into the github repository. Repository github serves as a code store so that it can unify code created by the developers. If any code changes inside the repository, then Jenkins will run automatically. Where Jenkins will rebuild from all the code that is in the repository and test the code

by using the link checker to be created. Once completed, Jenkins will release the results of the test.

3.2. Algorithm of automation link checker testing

Fig.2 is a flowchart of a link checker testing application. In the flowchart illustrated how the flow of the application to test the over-all link in a website link without registering all links in the website manually. Fig. 3 is a pseudocode of a link checker test application. In the pseducode described algorithm how the appli-cation to test the overall link in a link website without registering all links in the website manually. On line 3 to line 21 is the code that will check the link, where line 5 to line 7 to check the link, if the link is broken it will be inserted a group of broken link variable. After checking the link is completed, line 9 will take all links from the parent link. Line 11 to line 20, checks whether the child of the parent link includes an external url or a previously checked url, if included in an external url or a pre-checked url, will be ignored. Line 22 to line 30 represents the result of a link test. Link 22 to line 27 will display failed test if there is a broken link and display details of the link. Line 28 to line 30 will display test success if no link is broken.

4. Result and discussion

4.1. Preparing dataset

The author will perform application testing made on four different media websites. For manual testing will be done by 20 different people to test the links on the four websites and will be done only for 1 minute every website. Manual testing is done to determine the comparison of manual testing performed by humans made. The to-tal of all links and total broken links of each website will be shown in Table 1.

Table 1: Total Links

Website	Total of All Links	Total Broken Links
Website A	301 links	77 links
Website B	245 links	0 links
Website C	206 links	5 links
Website D	232 links	20 links

4.2. The applications

On the home page there is some information gained among devel-opers to find out whether testing of code entered into the github repository passes the test or not and some other information (such as when it was last succeeded, when it was last failed, and running time). Developers will also be able to add Jenkins configuration when needed. In automation link checker testing, there are several jobs that are used and have their own usability. Here are the details of the jobs used:

- Publishing: jobs to run automation link checker testing.
- Publishing-child-Get-Rev: jobs to get the revision to be re-trieved from the git repository.
- Publishing-child-SCM-Poll: jobs used to check every time, whether there is a change of code from git repository or not.
- Publishing-Test-Parent: jobs used to receive revision and run job publishing

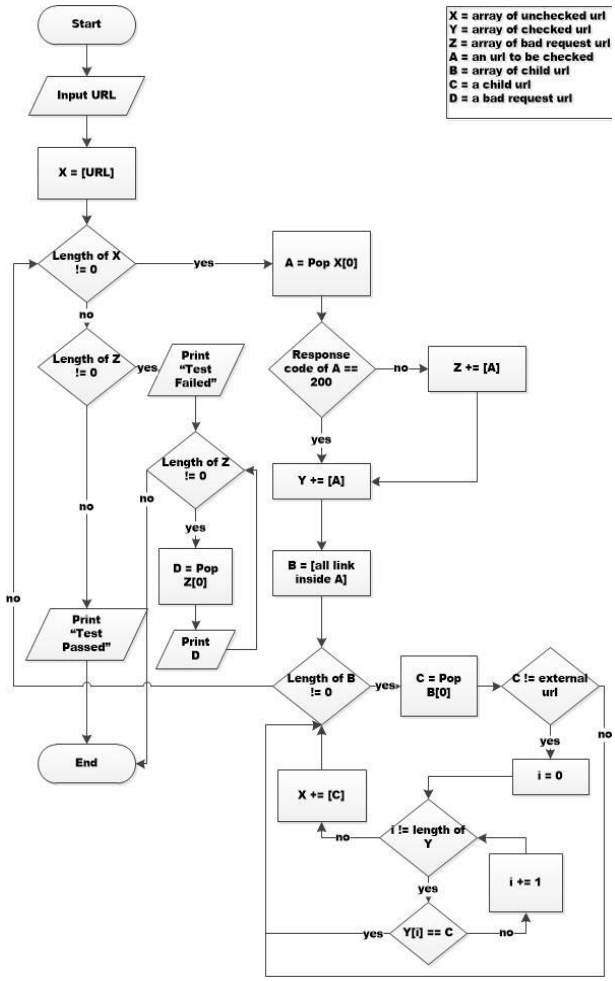


Fig. 2: Flowchart of Link Checker Testing

```
Algoritma Link Checker

 1: READ url
 2: SET X to [url]
 3: WHILE length of X != 0
 4:     SET A to pop X[0]
 5:     IF response code of A != 200 //check status code
 6:         SET Z to Z + [A]
 7:     END IF
 8:     SET Y to Y + [A]
 9:     READ child_links[] of A //get child link
10:     SET B to [child_links]
11:     WHILE length of B != 0
12:        SET C to pop B[0]
13:        IF C != external url //exclude external url
14:           FOR i = 0 to length of Y
15:              IF C != Y[i] //exclude checked url
16:                 SET X to X + [C]
17:              END IF
18:           END FOR
19:        END IF
20:     END WHILE
21: END WHILE
22: IF length of Z != 0 // test failed
23:     PRINT "Test Failed"
24:     FOR i = 0 to length of Z
25:        PRINT Z[0]
26:     END FOR
27: END IF
28: ELSE THEN //test passed
29:     PRINT "Test Passed"
30: END ELSE
```

Fig. 3: Flowchart of Link Checker Testing.

In Fig. 5 is the result of testing all the links in the website. From the results of automation link checker will show the number of links that have been in the check that is as many as 301 links that have been in check and display broken links along with the description of the link that is as many as 77 broken links. If there is a broken as shown in the Fig. 6, then the result of automation link checker test-ing on website fails and given the mark '×'. In Fig. 7 is the test result on a website. After the automation link checker testing tested 114 links on the website, there was no broken links on the website, so the test for the website was successful and marked 'v'.

4.3. Comparison with manual test

One of the advantages of using test automation is in terms of speed of testing an application. To show that automation testing is faster than manual testing then experiments are performed to prove it. Both tests are done with data development, and the same server device. For the first stage, the test is done manually for 1 minute by 20 different people and 4 different websites. The result of manual Testing is the average of 20 people who test, they can only test as many as 9 to 13 links in each websites in 1 minute. Some of them even made mistakes in documenting the test. Fig.8- Fig 12 show the result of automation.' The number of links tested on website 'A' as many as 301 links and testing for 57.85 seconds. The number of links tested on website 'B' as many as 245 links and testing for 32.46 seconds. The number of links that are tested on website 'C' as much as 206 links and testing for 32.80 seconds. The number of links that are tested on website 'D' as many as 232 links and testing



Fig. 4: Jenkins Home Page.



Fig. 5: An Example Automation Link Checker Testing Result Failed.

For 35.66 seconds. Of the four websites with a total of 984 links, the
Test only takes 158.76 seconds or 2.65 minutes. For more details on Table 2. So manual testing can only check 9 to 13 links in each website in 1 minute, while automated testing can check as many as 984 links from 4 websites in just 2.65 minutes. Not only that, manual testing allows errors in writing documentation, whereas automation testing is more accurate than manual testing

Table 2: Detail Speed of Automation Testing

Website	Total Links	Duration
Website A	301 links	57.85 seconds
Website B	245 links	32.46 seconds
Website C	206 links	32.80 seconds
Website D	232 links	35.66 seconds



Fig. 6: Mark for Failed Test.



Fig. 7: Example of Successful Automation Link Checker Testing Results.

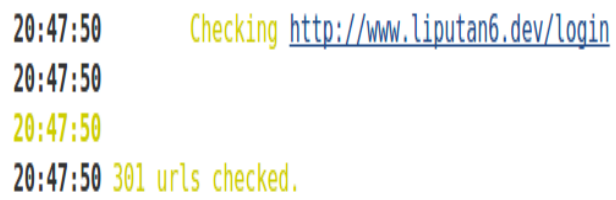


Fig. 8: Number of Automated Test Results Links on Website 'A'.

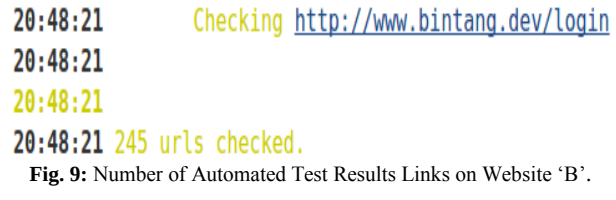


Fig. 9: Number of Automated Test Results Links on Website 'B'.

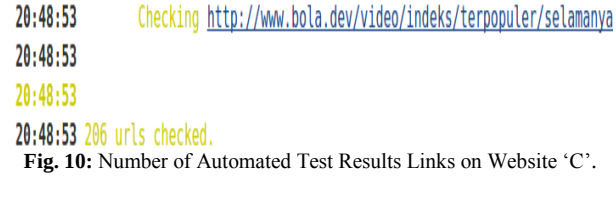


Fig. 10: Number of Automated Test Results Links on Website 'C'.

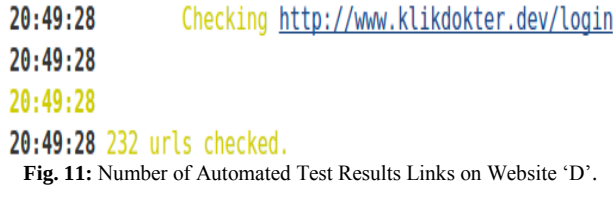


Fig. 11: Number of Automated Test Results Links on Website 'D'.

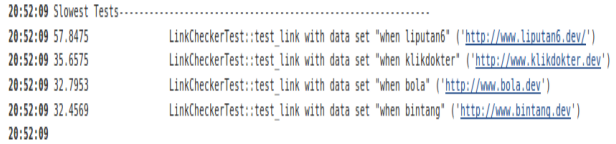


Fig. 12: The Speed Sequence of Automation Testing Results from Four Website.

The next stage is testing stages by the test engineer and after that can continue into the deployment stage. For further research and devel-opment, the feature can be embedded into system which works with some variety of links.

5. Conclusion

The automation link checker is implemented using Jenkins as a continuous integration tool to make it easier for developers. The result shows the system can reduce to check the broken link. The developers can also know the history of testing performed by the application, and can find out who changed the code and the code changes made. If the test is successful, no broken links can then proceed to the next stage, is testing stages by the test engineer and after that can continue into the deployment stage. For further research and development, the feature can be embedded into system, which works with some variety of links.

References

[1] R. S. Pressman, "Software Engineering A Practitioner's Approach (8th Ed.)", New York: McGraw Hil, 2014.

[2] M. S. Hossain and M. S. Hosain, "Web Test Integration and Performance Evaluation of E-Commerce Web Sites," International Journal of Computer Science and Information Security, Vol. 10, No. 9, pp. 65-69, 2012.

[3] G. Tassey, "The economic impacts of inadequate infrastructure for software testing." National Institute of Standards and Technology. Planning Report 02-3, 2002.

[4] S. Singla and H. Kaur, "Selenium Keyword Driven Automation Testing Framework," *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 4, Issue 6, pp. 125-129, 2014.

[5] S. Kaur, "An Automated Tool for Web Site Evaluation", *International Journal of Computer Science and Information Technologies*, Vol. 3(3), pp. 55-69, 2012.

[6] M. Hilton, T. Tunnell, K. Huang, D. Marinov and D. Dig, "Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects," *Automated Software Engineering*, 2016.

[7] M. Hilton, N. Nelson, and D. Dig, T. Tunnell and D. Marinov, "Continuous Integration (CI) Needs and Wishes for Developers of Proprietary Code," Corvallis, OR: Oregon State University, Dept. of Computer Science, 2016.

[8] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 805-816, 2015.

[9] P. Gerrard. "Risk-based e-business testing. Part 1: Risks and test strategy". www.gerrardconsulting.com, 2000.

[10] N. K. Tyagi, A. K. Solanki and M. Wadhwa, "Analysis of Server Log by Web Usage Mining for Website Improvement," International Journal of Computer Science Issues, Vol. 7, Issue 4, pp. 17-21, 2010.

[11] D. J. F. Tawfeq, D. A. M. S. Rahma and E. Q. Ahmed, "Detecting a Broken Link in a Web Site," Al-Mansour Journal Issue, 2012.

[12] V. Kumar, "Comparison of Manual and Automation testing," International Journal of Research in Science and Technology, Vol. No. one, Issue No. V, 2012.

[13] R. M. Sharma, "Quantitative Analysis of Automation and Manual Testing," International Journal of Engineering and Innovative Technology, Volume 4, Issue 1, 2014.

[14] P. M. Duvall, S. Matyas and A. Glover, "Continuous Integration: Improving Software Quality and Reducing Risk", Massachusetts: Pearson Education, Inc., 2007.

[15] R. M. Betz and R. C. Walker, "Implementing Continuous Integration Software in an Established Computational Chemistry Software," Software Engineering for Computational Science and Engineering (SE-CSE), pp. 68-74, 2013.

[16] J. Gray, "A 30 Minute Project Makeover Using Continuous," Verilab, Inc., 2012.