

## An evaluation model of website testing framework based on ISO 25010 performance efficiency

Dias Tri Kurniasari, Siti Rochimah

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

### Article Info

#### Article history:

Received Jul 10, 2024

Revised Sep 21, 2024

Accepted Sep 30, 2024

#### Keywords:

Framework

ISO 25010

Model evaluation

Performance efficiency

Web testing

### ABSTRACT

Testing is an important aspect of software development. Automation testing is now widely used to achieve better and more efficient results. Various automation testing frameworks are available in the market. However, one of the major challenges is determining which automation testing framework is suitable for testing. This study proposes an evaluation model for evaluating web automation testing frameworks based on seven performance efficiency factors to address this issue. The model evaluates five types of transactions commonly used on the web; CRUD, Get Massive Data, search, file upload, and file download. In addition, the tested frameworks are categorized as good, medium, and low. To measure the success of the research, expert weighting was also used. Based on the results obtained for all types of transactions, almost all classifications between the experimental results and weighting were in the same class. Although the model was found to be effective with a 100% accuracy rate, it had an accuracy rate of 80% for upload transactions. The outcomes of this study serve as a valuable reference for choosing suitable software for both tested frameworks and other software applications. In future studies focus on narrowing the selection based on not only performance but also functionality and ease of use.

This is an open access article under the [CC BY-SA](#) license.



### Corresponding Author:

Siti Rochimah

Department of Informatics, Institut Teknologi Sepuluh Nopember

Surabaya, Indonesia

Email: siti@its.ac.id

## 1. INTRODUCTION

Software testing is an essential part of software development or SDLC [1]. SDLC is a software development flow consisting of product design, development, and testing products [2]. Software testing detects defects in applications [3]. In addition, software testing attempts to ensure that the developed software meets requirements and specifications. A high-quality product is one aspect of a software project [4]. Various types of software testing exist, including functional suitability, performance efficiency, security, and usability [5]. Two methods can be used for software testing: manual and automated [6]. Manual testing is suitable for testing complex test cases; however, an automated testing framework simplifies and speeds the testing process [7]. Frameworks are guidelines used to help software execute specific commands [8].

One of the challenges in automation testing is that many tools exist today. Various tools create confusion for ordinary people when selecting suitable tools. In addition, not all tools are ideal for each web transaction and testing goal. Therefore, previous studies have used various software testing tools to obtain the best software performance. Previous studies have used various software testing tools in software testing: Selenium, Katalon, and Telerik Test [9], [10].

Software testing is performed by measuring the performance of the software testing tools. Research related to comparing testing frameworks that focus on software performance efficiency is rare. Similar to the

research conducted by Yuniasri *et al.* [11], performance measurement is a major consideration in maximizing the test automation function. Other research conducted by Thooriqoh *et al.* [12] is also related to performance efficiency by focusing on resource utilization and using the weight coefficient by expert judgment. However, the two studies have their respective weaknesses, namely, the research by Yuniasri *et al.* [11] only compared two frameworks, and there was no validation by experts. Meanwhile, research by Thooriqoh *et al.* [12] succeeded in improving previous research but only focused on resource utilization. In performance efficiency, there are two other important aspects, namely, time behavior and capacity.

Based on various studies, various frameworks for automated testing are effective. The advantages and disadvantages of some previously mentioned studies can be combined to facilitate new research into selecting software testing frameworks. This is because each software or testing goal requires a suitable framework to obtain satisfactory results. In addition, previous studies still used one testing tool and did not compare the performance of the software tools. Therefore, this research proposes an assessment model for web-based software quality testing frameworks. The framework is recommended based on the calculation of several factors related to three aspects of performance efficiency, in accordance with ISO/IEC 25010 standardization. In addition to using the calculation results from testing, this study uses the weight coefficient to evaluate the testing framework. The coefficient is obtained from the assessment of the scale of importance by experts and is used as the basis for calculating the assessment. Later, the assessment model can be used for various website testing frameworks, although it was not used in this study. This research is expected to serve as a reference or consideration for developers when determining a framework that suits their needs.

The following section will explain this further. Section 2 describes the literature review. Section 3 presents the proposed methodology for classifying the quality of web-based software. Section 4 describes the classification results obtained using the proposed method. Section 5 presents the conclusions of the analysis.

## 2. RELATED WORK

Software development, whether application programming interface (API), website, or mobile, is inseparable from one of the critical stages, testing. Software testing has become a popular research topic in many software fields, including websites. Website-based software testing has been extensively conducted in various studies [13], [14]. Mazhar *et al.* [15] analyzed and compared the performance of several web-based applications. This study successfully identified components that can be used to improve website performance. However, there was no comparison with other automation tools used in this study, which limited the ability to assess the effectiveness and efficiency of the methods.

Software testing generally involves automation testing. This is achieved because it takes less time, incurs lower costs, and identifies a more diverse range of errors. Various tools, such as Postman, Katalon, JMeter, and Selenium, can be used for automation testing. Various automation testing tools have become interesting research topics [16], [17]. Bahaweres *et al.* [18] applied a method for automation of graphical user interface (GUI) testing on the CURA and Swag Labs web applications. The behavioral-driven development (BDD) technique was used, and running was performed using the Katalon studio. This study obtained promising results, and almost all test cases were successfully executed. However, no other method was used for comparison in this study, and no expert assessments were conducted.

The existence of various tools that can be used today is one of the research topics, namely comparative studies [19], [20]. However, using multiple tools can confuse ordinary people when selecting suitable tools. One way to overcome these results is to build a recommendation model. Various studies have discussed this issue, such as Thooriqoh *et al.* [12], which creates a recommendation model for API testing tools based on resource use performance. This study succeeded in classifying the tools according to the experts' expectations. However, the scope of transactions in this study is minimal, and it may not perfectly cover the functionality and scenarios of the real world.

One of the important factors that affect the selection of automation tools is their performance. However, there are few studies that focus on testing the performance of tools. A study on performance efficiency was conducted by Yuniasri *et al.* [11]. This study compares the performances of two automated tests: cucumber and mocha. Fifty test cases of the Prestashop web content management system were tested in this study. This research shows positive results or succeeds in identifying more efficient tools, namely cucumber. This type of research should be conducted using more tools to broaden the coverage beyond just comparing the two tools.

Based on various studies, various frameworks for automated testing are effective. However, selecting is rarely a significant topic in research and industry. Every application or testing goal requires a suitable framework to obtain satisfactory results. Therefore, this research proposes a testing framework recommendation model based on performance efficiency for websites. The existence of this research is expected to serve as a reference for determining suitable testing frameworks to be used as needed.

### 3. METHOD

This chapter describes the stages of the research. The recommendation model should undergo three main stages: exploratory study, data collection, and development model evaluation. The stages of developing the web testing framework evaluation model are illustrated in Figure 1.

An exploratory study was conducted to identify the components needed in the research. This can be in the form of transaction types, performance efficiency ISO 25010, and testing tools. In the data collection stage, five transaction types represented by a test case were used to test the performance of the five existing tools. To obtain accurate results, each test case was repeated five times. The proposed framework is based on the findings from the exploratory and data collection stages. The details of these three stages are explained in the next section.

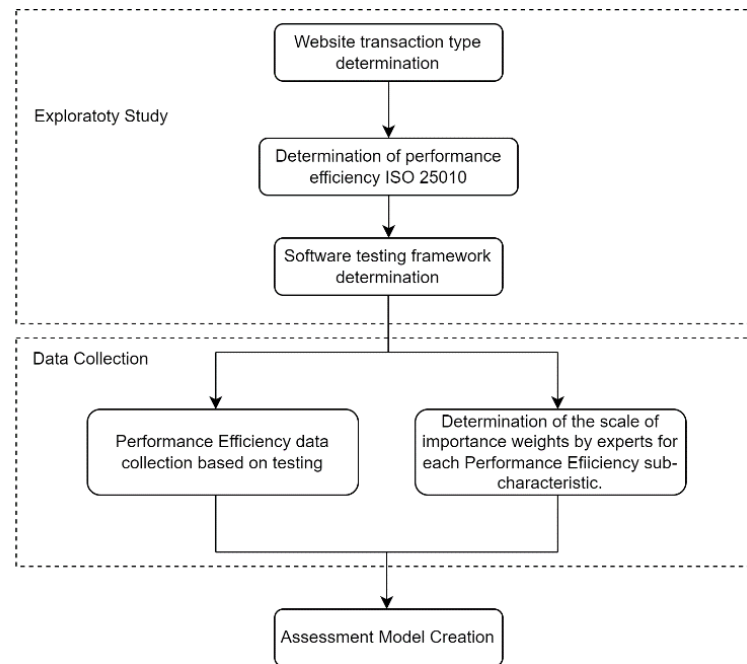


Figure 1. Research design

#### 3.1. Exploratory study

The exploratory phase of this research is called an exploratory study, and it includes a thorough and critical analysis of the relevant literature. This phase aims to determine the proposed framework's scope. Three activities are carried out at this stage. First, identifying the types of transactions that are commonly used on websites today. Second, identification of the characteristics in the performance efficiency based on ISO 25010. Third, the determination and identification of tools used in automated web testing. Detailed information about the three stages is presented in the following subsection.

##### 3.1.1. Website transaction type

This study considers several types of transactions that exist on the website. The types of data that will be used are getting massive data, creating, reading, updating, and deleting (CRUD), searching, uploading, and downloading files. A test case is created for each transaction type, and the execution time is calculated. In this study, the request to retrieve a large amount of data or get massive data is exemplified by retrieving data in the form of a graph with several parameters. CRUD transactions are performed by adding users, changing user data, and deleting users. Search transactions are performed by searching items on the landing page. At the same time, file upload is performed by uploading certificate files during item creation. File download is performed by downloading item data in Excel format.

##### 3.1.2. Performance efficiency ISO 25010

ISO/IEC 25010 is an international standard that defines the software product quality model. ISO 25010 is used to evaluate or ensure testing and research in accordance with software quality standards [21]. By using ISO 25010, researchers can measure and compare the quality of various tools and frameworks in an

objective and structured manner. An attribute of ISO 25010 is its performance efficiency. Performance efficiency has three sub-characteristics: resource utilization, time behavior, and capacity [22]. Resource utilization measures the amount of resources needed. The time behavior measures the system response, processing, and output times. The capacity subcharacter is used to evaluate the optimum limit of the system. Resource utilization analyses the required resources, including CPUs, memory, I/O devices, storage, and bandwidth. The real processor time, operating time, device storage, and data transmission capacity are the subcharacteristics and metrics employed in this study. The average resource value was calculated by contrasting the resources available on the device with the resources required by the framework in each trial [23]. Time behavior uses metrics like mean response time, turnaround time conformance, and throughput conformance to determine the outcomes of computations. The system’s mean response time is useful for determining how quickly it reacts to user input. In addition, the overall time spent in each reaction to user input was divided by the total number of trials conducted. The total completion time required for the system to complete one test case is the total completion time. Meanwhile, one way to gauge throughput appropriateness is to count the number of tasks that are completed successfully in a certain time [24].

**3.1.3. Automation testing tools**

A testing automation framework is a collection of rules, guidelines, and integrated modules and libraries used to create and run test scripts. Five frameworks are evaluated in this research. The Selenium IDE, Katalon, Mabl, Test Telerik Studio, and Cypress frameworks were used. Selenium IDE is a browser automation tool that allows users to create, record, and playback automated tests for web applications [25]. Katalon is an integrated software testing platform that provides a comprehensive solution to test automation [26]. Mabl is a test automation platform that provides a simple, scripless, and intelligent approach to automated testing [27]. Telerik Test Studio is a test automation tool designed to simplify User Interface (UI) and performance testing for web, desktop, and mobile applications. An open-source framework for end-to-end web application testing is called Cypress [20]. Then, the five frameworks implement the previously defined cases, and the test time is stored in each case.

**3.2. Data collection**

In this subsection, the data collection process used in this research is explained. Two types of data will be used: testing execution data from the frameworks and expert importance weight scale data. Test execution data will be collected by running various types of transactions using different web automation frameworks. Meanwhile, importance weight scale data will be gathered through questionnaires distributed to experts in software development and testing.

**3.2.1. Performance efficiency scores**

In this stage, performance efficiency scores are obtained through testing. The tests are conducted by executing predefined test cases on each testing framework. The initial step involves compiling a list of test cases for each transaction type. These test cases will be run on the testing frameworks, and the performance efficiency scores for each framework and transaction type will then be calculated.

**a. Test case design**

A test case is the design of an action performed by a user on the system. In this study, each transaction was represented by a single test case, which is happy flow. The Happy flow test condition is performed by predetermined steps to obtain the results as expected [28]. The website tested in this study was pipesales.com in the test environment. An example of a test case used in this study is given in Table 1.

Table 1. Test case example of changing user data

ID	Detail test Case	Precondition	Steps	Data	Expected result
01	Users can change personal data	Users already logged in	Click the account button.	-	Data changes are successfully made and saved in the database.
			Click profile	-	
			Click the edit button.	-	
			Input new name	Dias New	
			Click the Update button	-	

**b. Creating a data collection program**

In collecting performance efficiency data, not all of them can be done directly or need program assistance, namely, resource utilization data. In this study, a python-based program was used by using the psutil and libraries, which are run in real time. Scapy is a python library that manipulates network packets [29].

This library is used to obtain values for performance efficiency aspects, namely bandwidth utilization. Psutil is a Python library for obtaining information about system resources and processes running on the operating system. The library is used to obtain the value of the performance efficiency aspects of mean processor utilization and storage utilization. While, the data required for time behavior is obtained from the test result report provided by the testing framework. The results are then calculated based on a previously defined equation. Capacity was defined as 1 because in the testing process, each framework was only used by 1 device.

#### **c. Test case implementation**

The test case implementation describes the steps involved in applying the test cases previously created in the test framework. This process begins by using the record feature in the test framework to record actions performed during manual testing. This record feature automatically generates a test script based on these actions, including all test steps, interactions with page elements, and input data. Before recording, it is necessary to enter the base url of the target website.

Once the test script is generated, the next step is to re-run the script to ensure that all test steps are recorded correctly and could be executed without errors. This re-running process is important for identifying and correcting errors or discrepancies in the script. If there are problems, such as script writing errors, failures in interaction with page elements, or invalid input data, the script will be analyzed and corrected. This iterative process continues until all test cases are successfully run.

#### **d. Test execution**

At this stage, the original test is performed by executing the previously implemented test script. The data from this test will be recorded in detail and used in the next stage. Along with the execution of the test script, the created data capture program is also run. This program collects resource utilization data during the testing process. The combination of test execution and data capture ensures that the information required for the next stage is obtained successfully. At this stage, the test framework produces a test report that can be used to retrieve the time behavior data, namely the total test time.

### **3.2.2. Determination of the importance weight scale**

In this study, in addition to using the performance efficiency values obtained from the test results, expert judgment is also used. This expert assessment will be calculated according to the weight of the scale of importance for each aspect of performance efficiency used in the construction of the assessment model. The use of importance scale weights is useful for evaluating website testing frameworks that cannot be performed simply by comparing the value of one aspect to another. This is because each aspect has its own importance in each test case.

#### **a. Question list design**

In this stage, a list of questions related to the performance efficiency metrics based on ISO/IEC 25010 is designed. The importance weight scale values were obtained through questionnaire data collection to determine appropriate attributes for measurement. Each question in the seven performance efficiency metrics was grouped according to each type of transaction or test case. The questions were presented using a 5-point Likert Scale to avoid bias in the scores provided by respondents if the intervals were too close [30]. The Likert scale was measured as follows: a score of 5 for “very important,” a score of 4 for “important,” a score of 3 for “neutral,” a score of 2 for “not important,” and a score of 1 for “very unimportant.” Examples of this question are “How important is the processor in crud transactions”.

#### **b. Questionnaire data collection**

The questionnaire data collection was conducted by distributing the questionnaire to 5 respondents with the help of Google Forms. The targeted respondents are individuals who are experts in the field of software, specifically developers and testers who are accustomed to using testing frameworks. Additionally, respondents were familiar with the five types of transactions on websites. With evaluations from 5 experienced experts who know the correlation between hardware and the performance of testing frameworks, a basis for constructing the weighting model was established.

#### **c. Data validity check**

This stage was performed to ensure that the questionnaire used in the research was appropriate. One method used to check the validity of the questionnaire was to calculate the standard deviation of the responses received. The standard deviation is a measure that shows how much variation or dispersion from the average value (mean). The calculation is shown as (1).

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (1)$$

From (1),  $n$  is the number of respondents. Where  $x_i$  is the score of each respondent. And  $\mu$  is average response's score. The standard deviation is  $\sigma$ .

By calculating the standard deviation, the spread of respondents' answers that vary from the average can be observed. A low standard deviation indicates that most respondents gave similar answers, demonstrating consistency in their understanding and interpretation of the questionnaire questions. Conversely, a high standard deviation indicates a large variation in respondents' answers, which indicates that some questions may be unclear or understood differently by respondents. Standard deviation analysis helps evaluate and improve the validity of the questionnaire.

#### d. Importance scale weight calculation

After the questionnaire data passed the distribution check in the previous stage, the next step was the calculation of the importance weight scale. The first step is to convert the data from a Likert scale into an interval form. Data conversion was performed as (2) and (3).

$$Score = \sum_{i=1}^n T \times P_n \quad (2)$$

$$Index = \frac{Score}{Y} \quad (3)$$

From (2) and (3),  $n$  is the number of likert scales. Where  $T$  is the total number of respondents. And  $P_n$  is Likert score number choices. While  $Y$  is the maximum value.

After this conversion, the next step is to convert the interval values into a percentage index. This step is performed by summing all converted interval values to obtain the total interval value. Then, each interval value is divided by the total value as (3). The percentage index becomes the relative importance weight of each metric. By ensuring that the total weight is 1 (or 100%), each metric is assigned a weight that reflects its importance in the context of the test.

### 3.3. Model evaluation

The assessment model was based on the time data obtained during the data collection stage. The model was created using Python. The method used in this scoring model is classification with thresholding. Before determining the threshold, the score for each framework was normalized to ensure that both datasets (experiment and scale weight) were on the same scale so that they could be compared or combined accurately. After normalizing the data, the normalized performance efficiency value is multiplied by the importance weight scale, which is already on a 0-1 scale. This step produces a new value that reflects the relative contribution of performance efficiency based on the importance weight determined as (4).

$$X_i = \left( \frac{\sum_{j=1}^n (A_j \times B_j)}{n} \right) \quad (4)$$

From (4),  $i$  is the web transaction type. Where  $j$  is the performance efficiency aspect. And  $n$  is seven aspects of performance efficiency. The performance efficiency value of the testing experiment is  $A$  while  $B$  is the weight value of the expert. The total value after weighting of transaction type  $i$  is  $X_i$ .

After obtaining the score, which was the average of each aspect, the calculation was continued by determining the threshold 1/3 of the max and minimum scores obtained previously. The threshold is used to build and calculate the model, which is divided into three, whether it falls into the good (1), medium (2), or low (3) category. The threshold is determined by dividing the value range between the minimum and maximum values into three equal parts. Using this method, each class obtains balanced and measurable values. This makes it possible to set clear boundaries between each class so that assessment can be performed objectively and standardized.

## 4. RESULTS AND DISCUSSION

This paper proposes an evaluation model to assess web automation testing frameworks based on performance efficiency. Five frameworks—Selenium, Katalon Studio, Mabl, Telerik Test Studio, and Cypress—were tested and categorized into good, medium, and low performance classes. The classification was determined using data collected using a Python-based monitoring tool that tracked the execution of test cases in each framework. Expert weighting, which was obtained from questionnaires filled out by industry

professionals, was also used to validate the results and ensure the accuracy of measuring performance efficiency. Seven factors were used to assess performance, and the frameworks were evaluated based on their ability to handle five common web transactions. These include CRUD operations as seen in Table 2, get massive data as seen in Table 3, executing search functions as seen in Table 4, file uploads as seen in Table 5, and file downloads as seen in Table 6. The results are illustrated in Figure 2 for each transaction type. Figure 2(a) illustrates CRUD, Figure 2(b) illustrates get massive data (GMD), Figure 2(c) represents search, Figure 2(d) represents file upload, and Figure 2(e) represents file download.

The evaluation model developed in this study successfully classified each web automation testing framework based on its performance across various transaction types. The results indicate that different types of transactions yield distinct framework recommendations although certain transactions yield nearly identical recommendations. In CRUD operations, Mabl and Selenium demonstrate strong performance because they are designed to handle DOM element interactions quickly and effectively while supporting multiple programming languages and integrating well with other development tools [31]. Selenium and Cypress excel in handling massive data retrieval and search transactions due to their non-blocking architecture, which allows them to process multiple requests simultaneously with low latency, aligning with Gojare *et al.* [32]. Selenium's high performance in managing large data loads due to its efficient WebDriver implementation. Mabl is particularly effective in file upload transactions because of its AI-driven testing and smart wait features, which enable the framework to adaptively adjust wait times and reduce failures caused by varying test environment conditions [27]. For file download transactions, Katalon, Telerik, and Mabl perform well due to their built-in handling features that facilitate stable and accurate file downloads. Katalon and Telerik have management capabilities, including automated error handling, making them particularly effective in download scenarios [33].

Based on both the experimental and weighted results, the results indicate that the relative classification of the frameworks is largely consistent across the two methods. Nearly all transaction types exhibited a 100% match when compared, with only one transaction—file upload—showing an 80% match. This discrepancy is due to the weighted data calculation, where the score for file upload was slightly lower than the threshold for being classified as highly recommended. However, these results suggest that using either experimental data alone or incorporating weighting does not significantly alter the overall outcome, and the effectiveness aligns with the findings of Thooriqoh *et al.* [12], which ranged from 87.5% to 100%. The results of this study provide a valuable reference for determining suitable software for both the frameworks tested and those not considered in this study.

Table 2. Performance efficiency of CRUD

Framework	ISO 25010		Weighted ISO 25010	
	Score	Class	Score	Class
Katalon	0.37693	3	0.05016	3
Telerik	0.57267	2	0.08091	2
Mabl	0.68300	1	0.10060	1
Selenium	0.78204	1	0.11438	1
Cypress	0.63223	2	0.08497	2

Table 3. Performance efficiency of get massive data

Framework	ISO 25010		Weighted ISO 25010	
	Score	Class	Score	Class
Katalon	0.59243	3	0.08501	3
Telerik	0.57651	3	0.08133	3
Mabl	0.66877	2	0.09754	2
Selenium	0.85271	1	0.11773	1
Cypress	0.80052	1	0.11031	1

Table 4. Performance efficiency of search

Framework	ISO 25010		Weighted ISO 25010	
	Score	Class	Score	Class
Katalon	0.59243	3	0.08501	3
Telerik	0.57651	3	0.08133	3
Mabl	0.66877	2	0.09754	2
Selenium	0.85271	1	0.11773	1
Cypress	0.80052	1	0.11031	1

Table 5. Performance efficiency of file uploads

Framework	ISO 25010		Weighted ISO 25010	
	Score	Class	Score	Class
Katalon	0.63268	1	0.09100	2
Telerik	0.60039	2	0.08660	2
Mabl	0.74930	1	0.10874	1
Selenium	0.57143	2	0.08066	2
Cypress	0.37450	3	0.05719	3

Table 6. Performance efficiency of file download

Framework	ISO 25010		Weighted ISO 25010	
	Score	Class	Score	Class
Katalon	0.83742	1	0.11937	1
Telerik	0.67749	1	0.09527	1
Mabl	0.79344	1	0.11184	1
Selenium	0.56991	2	0.08173	2
Cypress	0.28556	3	0.03996	3

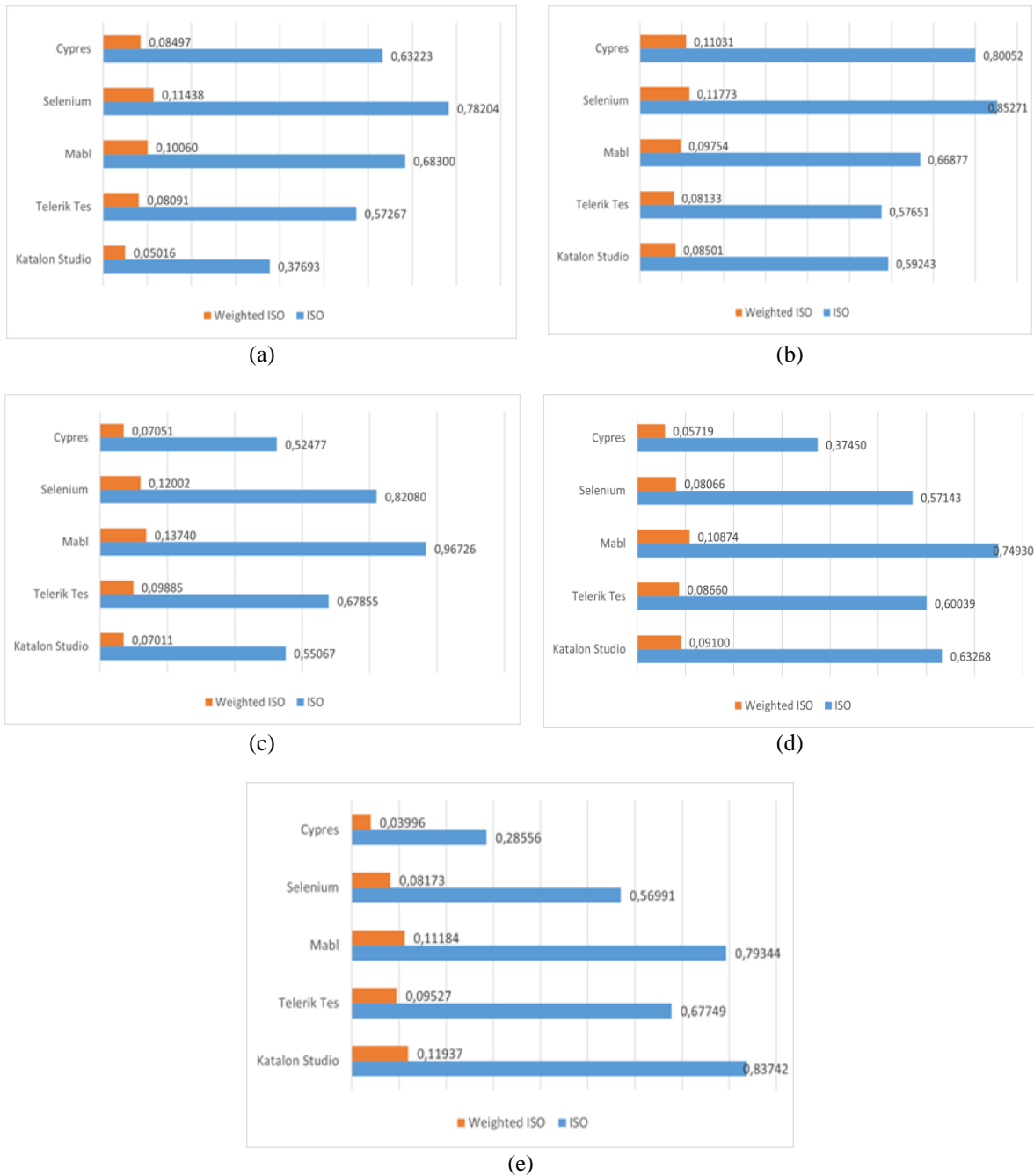


Figure 2. Distortion of each type of transaction result, (a) CRUD, (b) GMD, (c) search, (d) file upload, and (e) file download

### 5. CONCLUSION

Testing is a crucial aspect of software development, and with the rise of automation testing, it has become essential to achieve more efficient and accurate results. Although numerous automation testing frameworks are available, selecting the most suitable one for a specific testing scenario remains a significant challenge. This study addresses this issue by proposing an evaluation model based on seven performance efficiency factors to classify web automation testing frameworks. The proposed model successfully classified the frameworks across various transaction types, revealing that different transactions often require distinct framework recommendations. Mabl is effective for transactions requiring complex interactions and time adjustments. Selenium and Cypress are well-suited for transactions involving data requests and processing. Meanwhile, Katalon and Telerik excel in handling file download transactions. Across all transaction types, Selenium consistently performs excellently due to its WebDriver architecture, which provides efficient



performance with minimal computational overhead. Furthermore, the study demonstrates that experimental and weighted results largely align with 80% - 100% match, with nearly identical recommendations for most transactions. The outcomes of this study serve as a valuable reference for choosing suitable software for both tested frameworks and other software applications in the market. Additionally, these findings can serve as a reference for selecting appropriate software for untested transaction types by aligning their characteristics with those tested in this study. Furthermore, future recommendations should focus on narrowing the selection based on not only performance but also functionality and ease of use, especially for beginners.

## ACKNOWLEDGEMENTS

The authors would like to thank Inosoft Trans Sistem for allowing access and use of the Pipesales website as an object and for providing access to the official site.





## REFERENCES

- [1] Ashima, G. Shaheamlung, and K. Rote, "A comprehensive review for test case prioritization in software engineering," in *International Conference on Intelligent Engineering and Management (ICIEM)*, 2020, pp. 331–336. doi: 10.1109/ICIEM48762.2020.9160217.
- [2] M. A. Akbar *et al.*, "Improving the quality of software development process by introducing a new methodology–AZ-model," *IEEE Access*, vol. 6, pp. 4811–4823, 2018, doi: 10.1109/ACCESS.2017.2787981.
- [3] A. Salahirad, G. Gay, and E. Mohammadi, "Mapping the structure and evolution of software testing research over the past three decades," *Journal of Systems and Software*, vol. 195, p. 111518, 2023, doi: 10.1016/j.jss.2022.111518.
- [4] E. Cibir and T. E. Ayyildiz, "An empirical study on software test effort estimation for defense projects," *IEEE Access*, vol. 10, pp. 48082–48087, 2022, doi: 10.1109/ACCESS.2022.3172326.
- [5] A. J. Abdulwareth and A. A. Al-Shargabi, "Toward a multi-criteria framework for selecting software testing tools," *IEEE Access*, vol. 9, pp. 158872–158891, 2021, doi: 10.1109/ACCESS.2021.3128071.
- [6] K. Thant, H. H. K. Tin, and Ind, "The impact of manual and automatic testing on software testing efficiency and effectiveness," *Indian Journal of Science and Research*, pp. 88–93, May 2023.
- [7] M. A. Umar and Z. Chen, "A study of automated software testing: automation tools and frameworks," *International Journal of Computer Science Engineering (IJCSE)*, vol. 8, pp. 217–225, Dec. 2019, doi: 10.5281/zenodo.3924795.
- [8] A. Almogahed *et al.*, "A refactoring classification framework for efficient software maintenance," *IEEE Access*, vol. 11, pp. 78904–78917, 2023, doi: 10.1109/ACCESS.2023.3298678.
- [9] B. Majeed, S. K. Toor, K. Majeed, and M. N. A. Chaudhary, "Comparative study of open source automation testing tools: selenium, katalon studio & test project," in *2021 International Conference on Innovative Computing (ICIC)*, 2021, pp. 1–6. doi: 10.1109/ICIC53490.2021.9693066.
- [10] H. V. Gamido and M. V. Gamido, "Comparative review of the features of automated software testing tools," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, pp. 4473–4478, Oct. 2019, doi: 10.11591/ijece.v9i5.pp4473-4478.
- [11] D. Yuniarsi, P. Damayanti, and S. Rochimah, "Performance efficiency evaluation frameworks based on ISO 25010," in *10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 2020, pp. 254–258. doi: 10.1109/EECCIS49483.2020.9263432.
- [12] H. A. Thooriqoh, S. Rochimah, C. Faticah, and M. Alfian, "A recommendation model of REST API testing framework based on resource utilization of ISO/IEC 25010," in *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2022, pp. 383–389. doi: 10.1109/ISRITI56927.2022.10052922.
- [13] K. Nagendran, A. Adithyan, R. Chethana, P. Camillus, and K. B. Bala Sri Varshini, "Web application penetration testing," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 1029–1035, Aug. 2019, doi: 10.35940/ijitee.J9173.0881019.
- [14] G. Desolda, R. Lanzilotti, D. Caivano, M. F. Costabile, and P. Buono, "Asynchronous remote usability tests using web-based tools versus laboratory usability tests: an experimental study," *IEEE Trans Hum Mach Syst*, vol. 53, no. 4, pp. 731–742, 2023, doi: 10.1109/THMS.2023.3282225.
- [15] S. Mazhar, S. Rashed, and S. Zubair, "Comparative analysis of software performance on web based application," in *IEEE 9th International Conference on Information, Communication and Networks (ICICN)*, 2021, pp. 346–350. doi: 10.1109/ICICN52636.2021.9673934.
- [16] V. Tiwari, S. Upadhyay, J. K. Goswami, and S. Agrawal, "Analytical evaluation of web performance testing tools: apache jmeter and SoapUI," in *IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, 2023, pp. 519–523. doi: 10.1109/CSNT57126.2023.10134699.
- [17] Palak, P. Gulia, and N. S. Gill, "Hybrid swarm intelligence-based software testing techniques for improving quality of component based software," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, pp. 1716–1722, Jun. 2021, doi: 10.11591/ijeecs.v22.i3.pp1716-1722.
- [18] R. B. Bahaweres *et al.*, "Behavior-driven development (BDD) cucumber katalon for automation GUI testing case CURA and swag labs," in *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 2020, pp. 87–92. doi: 10.1109/ICIMCIS51567.2020.9354325.
- [19] R. K. Lenka, U. Satapathy, and M. Dey, "Comparative analysis on automated testing of web-based application," in *International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, pp. 408–413. doi: 10.1109/ICACCCN.2018.8748374.
- [20] A. Chevuturu, D. Mathur, B. J. Kumar Reddy, and D. R., "A comparative survey on software testing tools," *International Journal of Engineering and Advanced Technology*, vol. 11, pp. 32–40, Aug. 2022, doi: 10.35940/ijeat.F3664.0811622.
- [21] I. U. Haq and T. A. Khan, "Penetration frameworks and development issues in secure mobile application development: a systematic literature review," *IEEE Access*, vol. 9, pp. 87806–87825, 2021, doi: 10.1109/ACCESS.2021.3088229.
- [22] J. Miguel, D. Mauricio, and G. Rodriguez, "A review of software quality models for the evaluation of software products," *International journal of Software Engineering & Applications*, vol. 5, pp. 31–54, Nov. 2014, doi: 10.5121/ijsea.2014.5603.





- [23] A. Ravello, J.-M. Desharnais, A. April, A. Gherbi, and L. Bautista Villalpando, "Associating performance measures with perceived end user performance: ISO 25023 compliant low level derived measures," in *The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization*, Mar. 2015. [Online]. Available: <https://espace2.etsmtl.ca/id/eprint/12245/> (Accessed: Jun. 03, 2024)
- [24] S. Karnouskos, R. Sinha, P. Leitão, L. Ribeiro, and T. I. Strasser, "The applicability of ISO/IEC 25023 measures to the integration of agents and automation systems," in *44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 2927–2934. doi: 10.1109/IECON.2018.8592777.
- [25] V. V. Krishna and G. Gopinath, "Test automation of web application login page by using selenium ide in a web browser," *Webology*, vol. 18, no. Special Issue, pp. 713–732, 2021, doi: 10.14704/WEB/V18SI04/WEB18160.
- [26] A. M. Sinaga and M. Sitanggang, "Performance comparison of katalon studio and appium on investree application in funding module," in *2023 IEEE International Conference on Data and Software Engineering (ICoDSE)*, 2023, pp. 196–201. doi: 10.1109/ICoDSE59534.2023.10291590.
- [27] T. Kazimov, T. Bayramova, and N. Malikova, "Research of intelligent methods of software testing," *System research and information technologies*, pp. 42–52, Dec. 2021, doi: 10.20535/SRIT.2308-8893.2021.4.03.
- [28] T. Atanasijevic, How To Create A Successful User Experience (UX) on The Happy Path and The Unhappy Path, *Conference: Human Computer Interaction, Master 4.0 Program, University of KragujevacAt: Kragujevac*, Serbia, 2020. doi: 10.13140/RG.2.2.10387.71207.
- [29] M. A. S. Al-Nuaimi and A. A. Ibrahim, "Analyzing and detecting the de-authentication attack by creating an automated scanner using scrapy," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 2, pp. 131–137, Feb. 2023, doi: 10.17762/ijritcc.v11i2.6137.
- [30] L. South, D. Saffo, O. Vitek, C. Dunne, and M. Borkin, "Effective use of likert scales in visualization evaluations: a systematic review," in *Eurographics Conference on Visualization (EuroVis)*, Apr. 2022. doi: <https://doi.org/10.1111/cgf.14521>.
- [31] S. Nyamathulla, Dr. P. Ratnababu, N. Sultana Shaik, and B. Lakshmi, "A review on selenium web driver with python," *Ann Rom Soc Cell Biol*, pp. 16760–16768, 2021.
- [32] S. Gojare, R. Joshi, and D. Gaigaware, "Analysis and design of selenium webdriver automation testing framework," *Procedia Computer Science*, vol. 50, pp. 341–346, 2015, doi: 10.1016/j.procs.2015.04.038.
- [33] R. Samli and Z. Orman, "A comprehensive overview of web-based automated testing tools," *İleri Mühendislik Çalışmaları ve Teknolojileri Dergisi*, vol. 4, no. 1, pp. 13–28, 2023, [Online]. Available: <https://dergipark.org.tr/en/pub/imctd/issue/79680/1299155> (Accessed: Aug. 25)

## BIOGRAPHIES OF AUTHORS



**Dias Tri Kurniasari**     has successfully earned a bachelor's degree (S.Kom.) in informatics from the Institut Teknologi Sepuluh Nopember in 2023. She is currently studying for a master's degree at the Department of Information Engineering, Institut Teknologi Sepuluh Nopember. Her research interests include software quality, traceability, and testing. She can be contacted at email: 6025231090@student.its.ac.id.



**Siti Rochimah**     successfully earned a doctoral degree (Ph.D.) in software engineering from Universiti Teknologi Malaysia in 2010. Currently, she is the head of the Software Engineering laboratory at the Department of Informatics, Institut Teknologi Sepuluh Nopember. Her work involves writing more than 100 articles related to software engineering. Her research interests include software quality, traceability, and testing. She can be contacted at email: siti@its.ac.id.