UDC: 004.93; 60; 620 DOI: https://doi.org/10.15407/jai2022.01.269

API TEST AUTOMATION OF SEARCH FUNCTIONALITY WITH ARTIFICIAL INTELLIGENCE

R. Mysiuk[1], V. Yuzevych[2], I. Mysiuk[3]
[1,3] Ivan Franko National University of Lviv, Ukraine
1, University st., Lviv, 79000
mysyukr@ukr.net; iruna.musyk8a@gmail.com
[2] Karpenko Physico-Mechanical Institute of the National Academy of Sciences of Ukraine
5, Naukova st., Lviv, 79060
yuzevych@ukr.net

[1] http://orcid.org/0000-0001-5244-1850
[2] http://orcid.org/0000-0002-7843-7646
[3] http://orcid.org/0000-0002-3641-4518

Abstract. One of the steps in software development is to test the software product. With the development of technology, the testing process has improved to automated testing, which reduces the impact of the human factor on error and speeds up testing. The main software products for testing are considered to be web applications, web services, mobile applications and performance testing. According to the testing pyramid, when testing web services, you need to develop more test cases than when testing a web application. Because automation involves writing software code for testing, the use of ready-made tools will speed up the software development process. One of the most important test indicators is the coverage of search functionality. The search functionality of a web application or web service requires a large number of cases, as you need to provide many conditions for its operation through the free entry of any information on the web page.

There is an approach to data-based testing, which involves working with a test data set through files such as CSV, XLS, JSON, XML and others. However, finding input for testing takes a lot of time when creating test cases and automated test scenarios. It is proposed to use artificial data set generators based on real values and popular queries on the website to form a test data set. In addition, it is possible to take into account the probable techniques of developing test cases.

It is proposed to conditionally divide the software for testing into several layers: client, test, work with data, checks and reports. The Java programming language has a number of libraries for working at each of these levels. It is proposed to use Rest Assured as a Restful client, TestNG as a library for writing tests with checks, and Allure report for generating reports.

It is noted that the proposed approach uses artificial intelligence for automated selection of test cases when creating a test to diversify test approaches and simulate human input and behavior to maximize the use of cases.

Keywords: automation testing, artificial intelligence, synthetic data, data generator, artificial datasets, web API testing, data-driven testing.

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ ПОШУКОВОЇ ФУНКЦІОНАЛЬНОСТІ ВЕБСЕРВІСУ ІЗ ШТУЧНИМ ІНТЕЛЕКТОМ

Р. В. Мисюк[1], В. М. Юзевич[2], І. В. Мисюк[3]
[1,3]Львівський національний університет імені Івана Франка, Україна
вул. Університетська, 1, м. Львів, 79000
mysyukr@ukr.net; iruna.musyk8a@gmail.com
[2]Фізико-механічний інститут ім. Г. В. Карпенка НАН України, Україна
вул. Наукова, 5, м. Львів, 79060
yuzevych@ukr.net

[1] http://orcid.org/0000-0001-5244-1850
[2] http://orcid.org/0000-0002-7843-7646
[3] http://orcid.org/0000-0002-3641-4518

Анотація. Одним із етапів при розробці програмного забезпечення є тестування програмного продукту. З розвитком технологій процес тестування удосконалився до автоматизованого тестування, яке зменшує вплив людського фактору на помилку та пришвидшує тестування. Основними програмними продуктами для тестування вважаються вебзастосунки, вебсервіси, мобільні додатки та тестування ефективності. Відповідно до піраміди тестування, при тестуванні вебсервісів потрібно розробляти більше тестових випадків, ніж для тестування вебзастосунку. Оскільки автоматизація передбачає написання програмного коду для тестування, то використання готових інструментів пришвидшить процес розробки програмного забезпечення. Одним з найважливіших тестових показників вважається покриття пошукового функціоналу. Пошуковий функціонал вебзастосунку чи вебсервісу потребує опрацювання великого обсягу випадків, оскільки потрібно передбачати множину умов його функціонування через вільне введення будь-якої інформації на вебсторінку.

Існує підхід до тестування на основі даних, який передбачає роботу з тестовим набором даних через файли типу CSV, XLS, JSON, XML та інші. Проте, пошук вхідних даних для тестування забирає багато часу при створенні тестових випадків та автоматизованих тестових сценаріїв. Запропоновано використати генератори штучного набору даних на основі реальних значень та популярних запитів на вебсайті для формування тестового набору даних. Крім цього, є можливість врахувати ймовірні техніки розробки тестових випадків.

Програмне забезпечення для тестування запропоновано умовно поділити на декілька прошарків: клієнтський, тестовий, робота з даними, перевірки та звіти. У мові програмування Java є ряд бібліотек для роботи на кожному з цих рівнів. Запропоновано використати REST Assured, як RESTful Client, TestNG як бібліотеки для написання тестів з перевірками, а також Allure report для генерування звітів.

Відзначено, що запропонований з урахуванням штучного інтелекту підхід для автоматизованого підбору тестових випадків при створенні тесту урізноманітнить тестові дані, максимально стимулюватиме введення даних людиною та поведінку для покриття більшості випадків.

Ключові слова: автоматизоване тестування, штучний інтелект, синтетичні дані, генератор даних, штучний набір даних, тестування вебсервісу, тестування на основі даних.

Introduction

Artificial intelligence involves modeling the ability of human intelligence by intelligent machines of various kinds. Today, artificial intelligence is increasingly being integrated into software quality assurance. Such an approach is best suited for automated type testing. In addition, it is worth mentioning that the web service requires the most attention when testing because malfunctions in the various Application Programming Interfaces (APIs) can cause errors in the work and and affect the status of the web pages. The web service is used to communicate with clients and the web server using the Internet [6]. It is also worth noting that it is still possible to test a web page manually, and testing a web service with a large number of requests can take a lot of time and resources. Therefore, API testing requires writing test scripts.

One of the main reasons for failing tests is frequent data changes. In test scenarios, you need to specify the expected result to check for convergence procedures. Usually, these are usually specific values that can vary.

Among the most common types of web services are REST, SOAP, XML-RPC.

In recent trends, the implementation of the REST - Restful architecture is the most popular in recent years. When generating checks for web services, you should take into account the status of codes and responses in JavaScript Object Notation (JSON) format. Therefore, API testing involves understanding and working on the Internet [6].

Literature review

In recent years, the use of artificial intelligence in testing has been particularly intensively considered [1, 4]. There are a large number of materials for data generation using genetic algorithms with metaheuristic search methods [11], and simple solutions, namely synthetic ones are described in [3, 8]. Moreover, different layers use machine learning approaches, depending on the implementation. The basic architecture of the framework extension consists of testing based on the data approach presented in [7]. Regarding the process of data generation, the article [9] describes various approaches, such as random, target-oriented, intelligent, path oriented data generators. However, it is worth considering the use of existing ready-made libraries and capabilities in selecting the right data for the chosen data-driven testing approach.

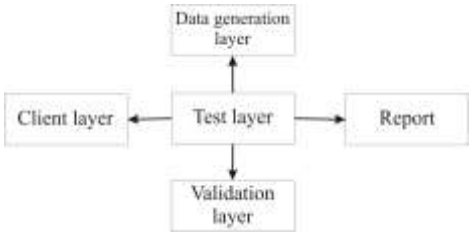

Fig.1. Basic layers of the framework for automated testing

Figure 1 shows the main components that make up the framework for automated testing. The main types of API [2] and graphical user interface (GUI) [5] testing using different techniques for writing code are considered in the works.

Testing can be applied to a pre-designed system for detecting defects [10]. This system has the ability to fully and partially search for data.

Methods and Materials

When developing a framework for automated testing, it is necessary to use several important libraries that simplify code writing.

RestAssured library is designed to communicate with the web service.

TestNG is considered to be the most well-known tool for setting up and writing test scripts due to a large number of possibilities for setting up the test.

Allure report is the most convenient for reporting after testing. This reporting form displays trends, path rate, number of passed and failed tests. The obtained parameters help to determine the test indicators and present the results in a modern presentation.

Among the main approaches to testing are behavior-driven, data-driven, test-driven, keyword-driven testing, and others. Since working with data is most used in data-driven testing, it is worth considering a method using the closest possible generated data set.

A maven is a popular tool for assembling a framework with different phases of construction and automatic loading of external libraries.

Assert J provides a set of advanced validations to flexibly process existing and expected results.

The Apache POI library belongs to the Java API and is used to work with Microsoft files. In the case of automated testing, the library is applied to read the generated values into tests.

A synthetic method uses for filling an artificial data set of different resources. Synthetically generated datasets are increasingly used for machine learning. The Java Faker library in the developed framework is used for the generation of this type of data.

Test case design techniques are used from testing theory [13].

Problem Formulation

Lack of data in testing is a fairly common problem faced by most researchers and software testers. This is especially common with search queries for picking up non-existent values and is time-consuming.

In addition, to obtain information, you need to access the web service using Uniform Resource Locator (URL). To form a specific request, you need to build a path to the resource in the network. The main components of a URL are host, port, and set of parameters. In turn, the parameters form pairs with a key and a value.

The q parameter is often used for search queries. Such queries are the most complex because they require checking the conditions of a full search, partial, and in some cases autocomplete.

The formation of a uniform data set complicates the testing process due to a large number of duplicates. That is why the approach proposed in this paper allows you to replace duplicates with randomly generated characters that are used as a test data set. In this case, testing will be as close as possible to user input.

It is also worth mentioning that duplication of tests is reduced by selecting a test data set.

Objective

The main purpose of this work is to use the automatic generation of realistic data taking into account the synthetic method for the automated testing process.

In most cases, a separate framework is developed to ensure uniformity and realism of type testing as a client. In addition, access to

the web service is external, not directly through the code. In such cases, the web service is considered as a black box with known inputs and expected outputs. The inputs mean that there are related to business requirements and documentation. This allows you to develop test cases based on various techniques, such as specification-based and structured-based. This is especially important when the outputs are the values that in the tests should match the expected results.

Testing results

The process of automated testing consists of the selection of test tools for the framework, the formation of test scenarios, writing the tests themselves, and analyzing the results.

In fig. 2 shows the main tools used to develop the framework and characterize the interaction between them. The central element of the system is TestNG, which is used to configure the test, as by analogy with fig.1 it is a formed test layer. This tool provides communication with other components of the framework.

Fig. 2. Interaction between the main tools in the developed framework for automated testing

Equivalence partition and boundary value analysis can be used in the case of testing the web service of the basic test design techniques of black-box testing.

Relevant test data are formed on the basis of test design techniques according to the following rules:

- No data. Empty values for text fields or null values for numeric values
- With correct and incorrect data
- Illegal symbols
- Boundary conditions [13]

Figure 3 shows the basic rules for forming many types of data sets.

The process of data generation is shown in Fig.4. The input components are real data from the Java Faker library, a file with illegal symbols for generating incorrect queries, and a list of popular queries for compiling statistics of the most used parts of the functionality.

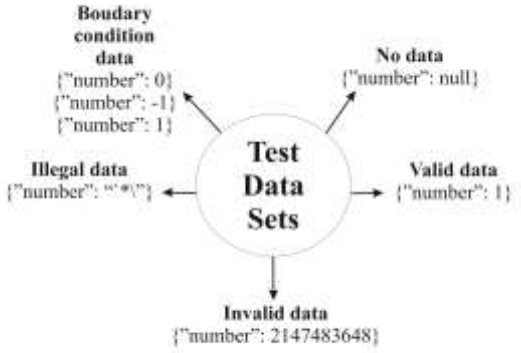

Fig. 3. Description of test data set options

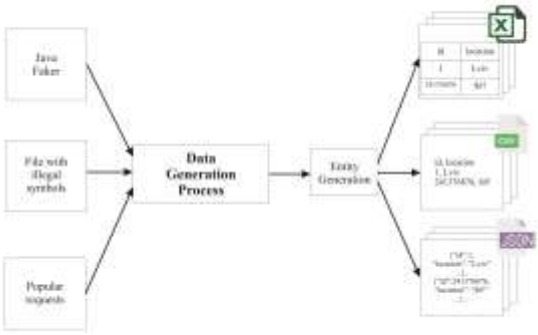

Fig. 4. The process of data generation

Popular queries can be obtained using New Relic [12] or from other monitoring tools.

The generation process is based on the random extraction of each data element and the formation of a complete record in the appropriate format in the file.

Entity Generation is based on serialization/deserialization processes from objects of a certain format and vice versa. The main file types are Excel Binary File Format (XLS), Comma-Separated Values (CSV), and JavaScript Object Notation (JSON) for storing test data.

Obtaining and visualizing the results is an important step in the development of

automated tests because it allows you to analyze the test results.

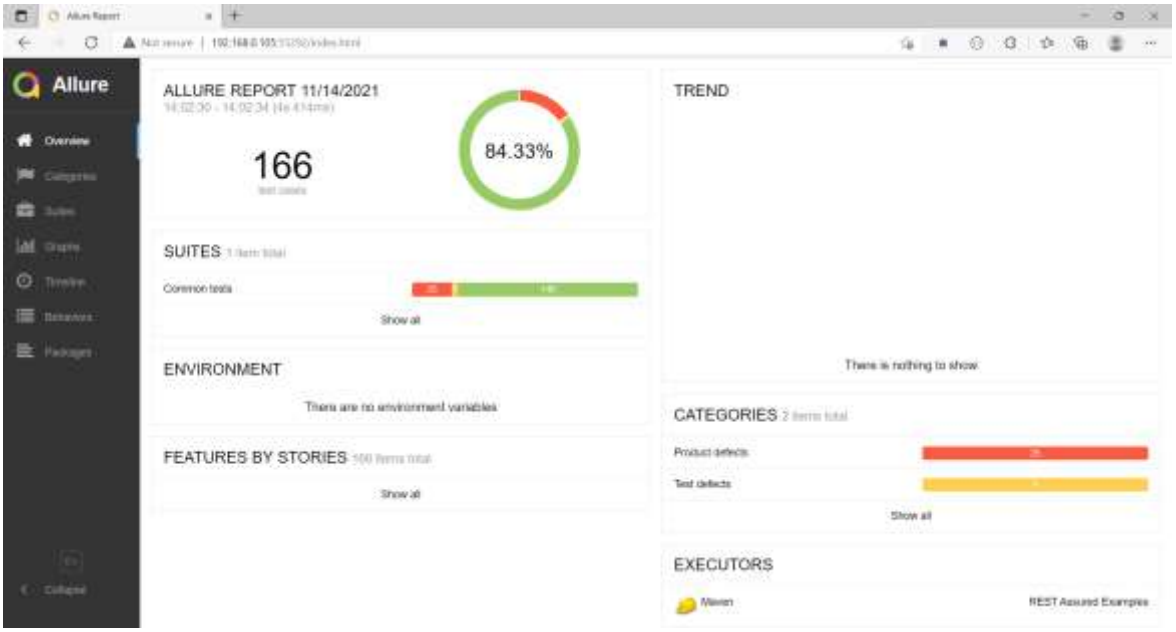

Fig. 5. Graphical user interface with test results

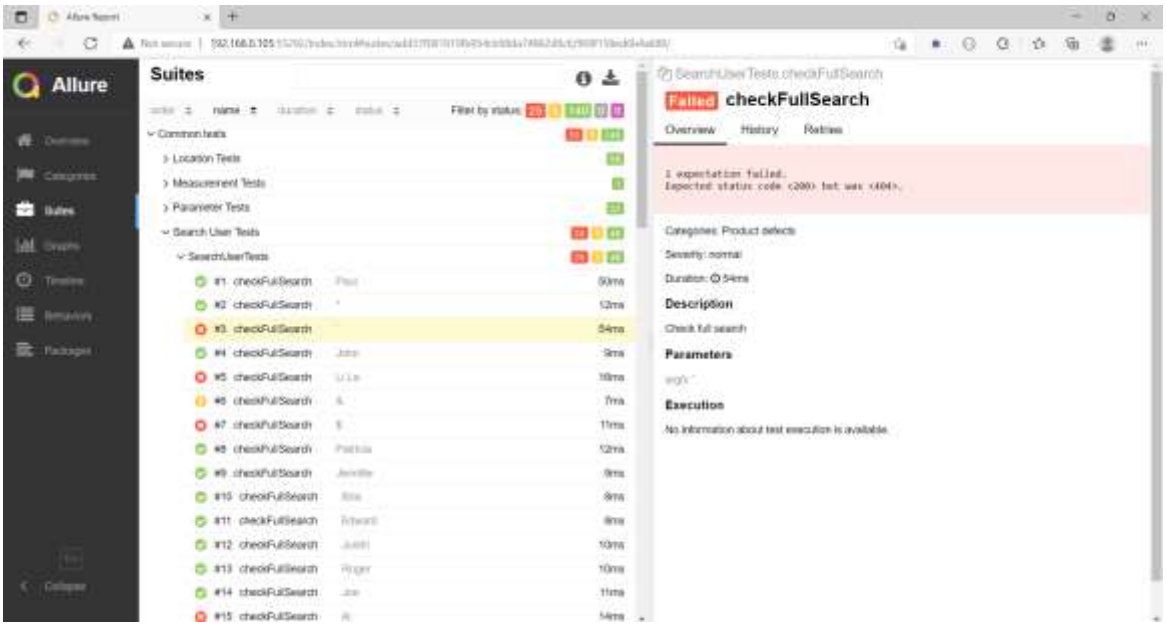

Fig. 6. Detailed test results

In fig. 5 shows the graphical interface of the tool for reporting Allure with test results. 166 different test data sets have been generated in this run. In addition, we can see from the report that the path rate is approximately 84%. The next step is to analyze the list of failed tests to correct errors in the web service.

After restarting the procedure several times, we are constantly improving the software. In this case, unpredictable combinations may occur that the tester may miss when compiling the test scenario.

As you can see in fig. 6, deploying all the test cases, you can determine the cause of failure of a particular scenario. In this case, the search engine did not use illegal

characters such as`$ and others. In addition, you can analyze the search capabilities of the web service in terms of the full or partial search for information.

Conclusion

The paper presents one of the examples of the use of artificial intelligence in the process of writing and implementing a method of automated testing. Namely, synthetic data are used with the formation of ready-made data sets, as well as taking into account the techniques of developing test cases.

Most sites generate page content based on responses from web services. The most difficult to cover and most used are the search functions on the web page. Search is a fairly flexible property on most sites. Therefore, it is noted that the offered technique allows to enter the text as full, partial, and also on the identifier. In partial cases, auto-replenishment may occur. For the correct and most similar to the input of data from a wide range of consumers need to have a real set of values, which are often personalized, and sometimes there is a limited number.

Synthetic data are quite useful for applied research, as it is possible to implement a procedure for generating an artificial data set to replace real values. In addition, such a data set can be used for the process of entering data into the database and then the tests allow to obtain the appropriate realistic expectations.

The testing process helps to reduce the number of errors in the software and constantly improve it.

Thus, the testing method proposed in this paper is best suited for the data-driven approach and simplifies the data generation process.

References

1. Trudova, Anna & Dolezel, Michal & Buchalcevova, Alena. (2020). Artificial Intelligence in Software Test Automation: A Systematic Literature Review. 181-192. doi:https://doi.org/10.5220/0009417801810192.
2. Alberto Martin-Lopez. (2020). AI-driven web API testing. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings (ICSE '20). Association for Computing Machinery, New York, NY, USA, 202–205.doi:https://doi.org/10.1145/3377812.3381388.
3. Jimena Torres Tomás, Newton Spolaôr, Everton Alvares Cherman, Maria Carolina Monard. (2014) A Framework to Generate Synthetic Multi-label Datasets, Electronic Notes in Theoretical Computer Science, 302, 155-176, ISSN 1571-0661, doi: https://doi.org/10.1016/j.entcs.2014.01.025.
4. Faezeh Khorram, Jean-Marie Mottu, Gerson Sunyé. (2020) Challenges & Opportunities in Low-Code Testing. ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS '20 Companion), Virtual, Canada. doi: ff10.1145/3417990.3420204ff. ffhal-02946812f.
5. Nguyen P., Maag S. (2021) A Machine Learning Based Methodology for Web Systems Codeless Testing with Selenium. In: van Sinderen M., Maciaszek L.A., Fill HG. (eds) Software Technologies. ICSOFT 2020. Communications in Computer and Information Science, 1447. Springer, Cham. doi: https://doi.org/10.1007/978-3-030-83007-6_9.
6. Halili, Festim & Ramadani, Erenis. (2018). Web Services: A Comparison of Soap and Rest Services. Modern Applied Science. 12. 175. doi: https://doi.org/10.5539/mas.v12n3p175.
7. Kachewar, Rohan. (2013). K model for designing Data Driven Test Automation Frameworks and its Design Architecture Snow Leopard. doi: 10.5120/3835-5331.
8. Ayala-Rivera, Vanessa & Mcdonagh, Patrick & Cerqueus, Thomas & Murphy, Liam. (2013). Synthetic Data Generation using Benerator Tool.
9. Hitesh, Tahbildar & Bichitra, Kalita. (2011). Automated Software Test Data Generation: Direction of Research. International Journal of Computer Science and Engineering Survey.2. doi: 10.5121/ijcses.2011.2108.
10. Shuvar R.Y., Prodyvus A.M., Yuzevych V.M., Ogirko I.V., Ogirko O.I., Kovtko R.T., Mysiuk R.V. (2021) Information technologies and threats in cyberphysical systems for displaying information in underground metal structures with defects. Stuc. intelekt. 26(1), 85-94. doi: https://doi.org/10.15407/jai2021.01.085.
11. Aleb, N., & Kechid, S. (2013). Automatic Test Data Generation Using a Genetic Algorithm. ICCSA.
12. New Relic [Online]. Available: https://docs.newrelic.com/docs/data-apis/get-started/introduction-new-relic-data-ingest-apis-sdks/.
13. Test Design Techniques [Online]. Available: http://istqbfoundation.wikidot.com/4.

Received 28.02.22
Accepted 10.05.22