Jurnal ELTIKOM :
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 6, No. 2, December 2022, page 118-125
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10.31961/eltikom.v6i2.550
Available online at http://eltikom.poliban.ac.id

A BOT APPROACH-BASED CAPACITY TESTING AUTOMATION FOR ONLINE VIDEO GAMES

Aditya M. Sidiq, Jati H. Husen*, Sri Widowati
School of Computing, Telkom University, Bandung, Indonesia
e-mail: adityamaulanasidiq@student.telkomuniversity.ac.id, jatihusen@telkomuniversity.ac.id, sriwidowati@telkomuniversity.ac.id

Received: May 19, 2021 – Revised: July 26, 2021 – Accepted: February 9, 2022

ABSTRACT

Online games are a type of computer game that can be accessed using the internet network and played with other players to play the same game. With the advance in online game development, game developers are required to develop games that can be played by many players in one game, especially in the capacity of an online game server. Those needs can be achieved by utilizing bots. However, previous works only conducted bot-based testing for testing network capabilities. In this research, those works will be extended further ftotesting the capacity of a game server. The result of this research suggests that bot approach testing can simulate real players adequately. Other than that, the bot approach also can be scalable. However, the result also suggests that the bot approach still has some limitations as bots cannot simulate the dynamics shown by real players. Special attention is also needed towards clients utilized for executing the bots for them to be scalable.

***Keywords**: Automation testing, Online game, Stress testing.*

ABSTRAK

Game online merupakan jenis permainan komputer yang dapat diakses dengan menggunakan jaringan internet dan juga dapat dimainkan bersama player lainnya untuk memainkan game yang sama. Oleh karena itu diperlukan usaha untuk mengetahui kemampuan daya tampung server pada sebuah game online. Bot dapat membantu untuk melakukan hal tersebut. Namun pada penelitian terdahulu hanya dilakukan penelitian menggunakan pendekatan bot pada game minecraft untuk menguji kemampuan jaringan. Pada penelitian ini, penelitian tersebut akan dikembangkan lebih lanjut untuk menerapkan pendekatan yang sama untuk menguji kapasitas server game. Hasil penelitian menunjukan bahwa bot dapat mensimulasikan pemain dengan cukup baik. Selain itu, bot dapat memungkinkan skalabilitas yang cukup tinggi. Namun bot juga memiliki beberapa kelemahan. Bot tidak dapat menunjukan dinamika yang sama dengan pemain asli. Selain itu, diperlukan perhatian khusus terhadap software client yang digunakan untuk memunkinkan eksekusi bot yang lebih efisien.

***Kata Kunci**: Game online, Pengujian otomatisasi, Pengujian stres.*

I. INTRODUCTION

ONLINE game market has grown significantly. This is led by the different levels of excitement compared to playing solo games. This is proven by the popularity of several games in the 90s, such as DOOM, Quake, and Warcraft which are played in international tournaments [1]. To provide the necessary quality to be marketed properly [2], [3], online games are required to provide servers with capacity that supports every need of all players, such as response and server capacity [4]. The server capacity of software, including online games, is an important element of a system. A system that does not have enough capacity will lead to fatal consequences [5], [6], e.g., a statistical website in Canada went down in 2016 because it was unable to accommodate a large number of traffics [5]. Therefore, the server becomes an important element for online games because the server is an interactive distributed system that works in real-time [1].

Considering the importance of a server for the quality of an online video game, it is necessary to test the load capacity to determine whether the software is a good server or not [7], [8]. Testing is important

for ensuring software quality [9], both functionally and non-functionally. Testing also validates and verifies developed systems [10], [11].

Testing generally can be done in 2 ways, namely automatic and manual testing [12], [13]. Testing the game server requires a lot of human resources and time efficiency [14], for example, players who access the online games massively and simultaneously [14]. Meanwhile, automatic testing has advantages in terms of time efficiency [15], [16] and can increase the effectiveness of testing to save human resources [17], [18]. The utilization of bots is one of the commonly used approaches to test server capacity, performance, and resource usage [19], [14].

The use of testing automation using bots or virtual users can simulate real-world conditions [20]. On the other hand, stress testing using bots will require a lot of computational resources [21]. However, previous research only focused on the capacity of the network and not the resources of the server resources itself. As server resources such as CPU and memory usage also important for the success of the online game operation, this research aim to understand the possible configuration of utilizing bots to test the performance of online video games server resource.

Therefore, the research questions (RQ) proposed in this study are as follows:

- RQ1. Which task is suitable for bots to test server capacity?
- RQ2. How many computation resources are needed for each task bot to test server capacity?

This study contributes to the software engineering body of knowledge, especially in software testing, in the form of knowledge about the characteristics of automatic testing of the quality attribute capacity. In addition, it also provides insight into tools that can be used to perform automatic testing of the capacity of video game servers.

This scientific article is arranged into four sections. Section 1, an introductory section, discusses the background of the problem that becomes the basis of this research. This study aims to find out whether using a bot can test the capacity of the server effectively. Section 2 discusses the research methods, which describe the testing procedures carried out in detail. Section 3 explains the results of tests that have been carried out to be evaluated to obtain answers related to the problems. Section 4 summarizes the results of the evaluation and what can be done for future research.

TABLE 1
SPECIFICATION OF CLIENT

Components of Client	Specification
CPU	Intel Core i7 3,5 GHz
RAM	8 GB
HDD	1 TB

TABLE 2
SPECIFICATION OF MICROSOFT AZURE SERVER

Components of Server	Specification
VM Size	B2s
vCPUs	2
RAM (GiB)	8
Temp Storage	8

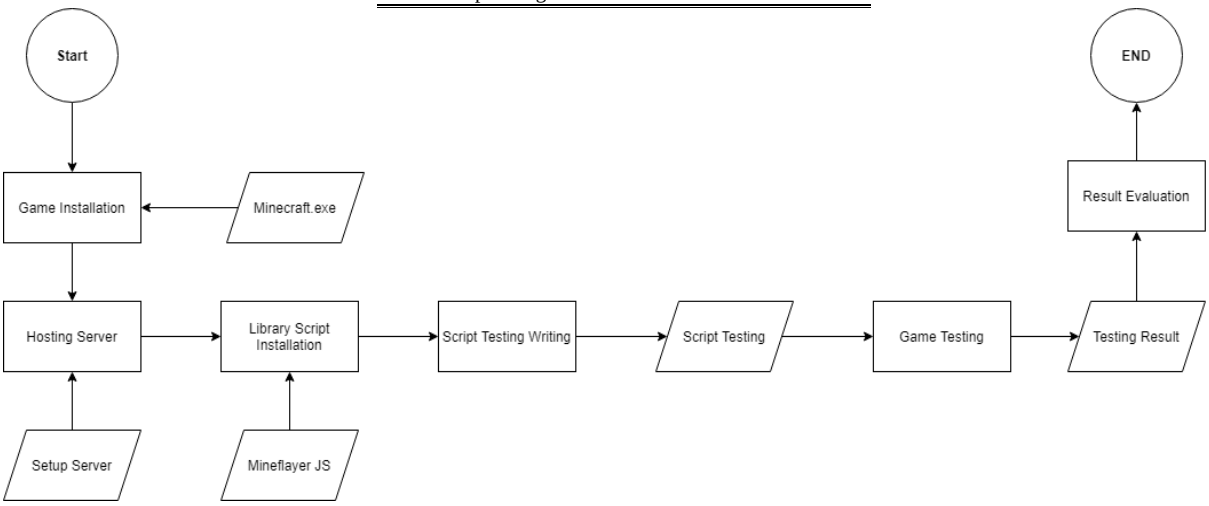

Figure. 2. Research Method

II. RESEARCH METHOD

In this study, we conducted two phases of research to answer previously stated research questions. The first one is comparing the bot to human players, which aims to answer RQ1. At the same time, the second one measures the resources needed to execute the bots to answer RQ2.

A. Bot to Human Comparison

The comparison is conducted by measuring the server's CPU usage from bots and human players. The aim of this experiment is to find out whether the bots can simulate human players correctly, both in normal and extreme conditions. The comparison is conducted with five players which already familiar with Minecraft. The CPU usage will be recorded for 30 seconds to capture the required data. Figure 1 shows the overview of this experiment.

B. Resource Utilization

This experiment aims to understand the scalability of the bot approach. Several stages were carried before employing the bots to test the game server. The research stages are illustrated in Figure 2.

1) Game Installation

The first stage in this study is game installation. In this case, the game is one of the important components to make this research obtain good results. The game used in this study is Minecraft version 1.16.14, which requires the specifications of the client as Table 1.

2) Hosting Server

The second stage is the hosting server required to host the installed game to be played online and tested for its capacity and performance. In this study, the hosting used is Microsoft Azure with the following server specifications in Table 2.

3) Library Bot Installation

The next stage is installing the library for the bot used to test the performance and capacity of the server. The bot library simplifies the testing and makes it more effective when making bot scripts for testing. The library used is mineflayer, an open-source library that is easy to access [22].

4) Script Bot Writing

After the necessary libraries are installed, the next step is writing a script for testing. The script describes the tasks to make the bot perform an activity in the game by interacting with all the elements of the game. In addition, the script has different levels for each task tested and affects the use of resources and server performance. An example of the source code of the bot used can be seen in Figure 3, while the full source code can be seen in the following link https://github.com/Letvinko/Code-Testing-Bot.

5) Game Testing

In the game server testing, the concept of task execution is as Figure 4.

Spawn Bot

The first step of executing the task is spawning the bot in the game. The bots that have been dispatched will make it easier to carry out tasks to be executed. In this testing, the bots will play a role in replacing the presence of real humans. In each spawn bot, a five seconds delay is made to avoid the rate limit on

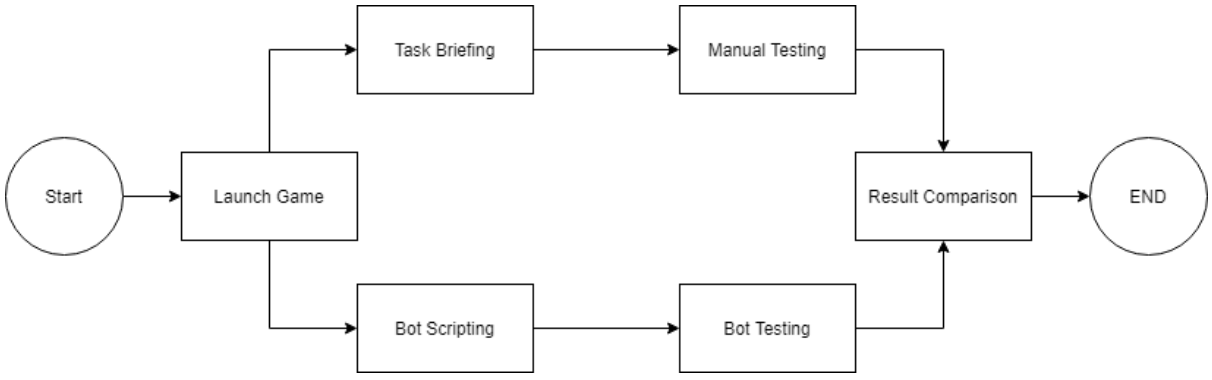

Figure. 1. Comparison Overview

```
const mineflayer = require('mineflayer')
let i = 0
function next () {
 if (i < 20) {
  i++
  setTimeout(() => {
   createBot(`mineflayer-bot${i}`)
   next()
  }, 5000)
 }
}
next()
function createBot (name) {
 mineflayer.createBot({
  host: 'amsrpl.southeastasia.cloudapp.azure.com',
  username: name
 })
}
```

Figure 3. Example of script code

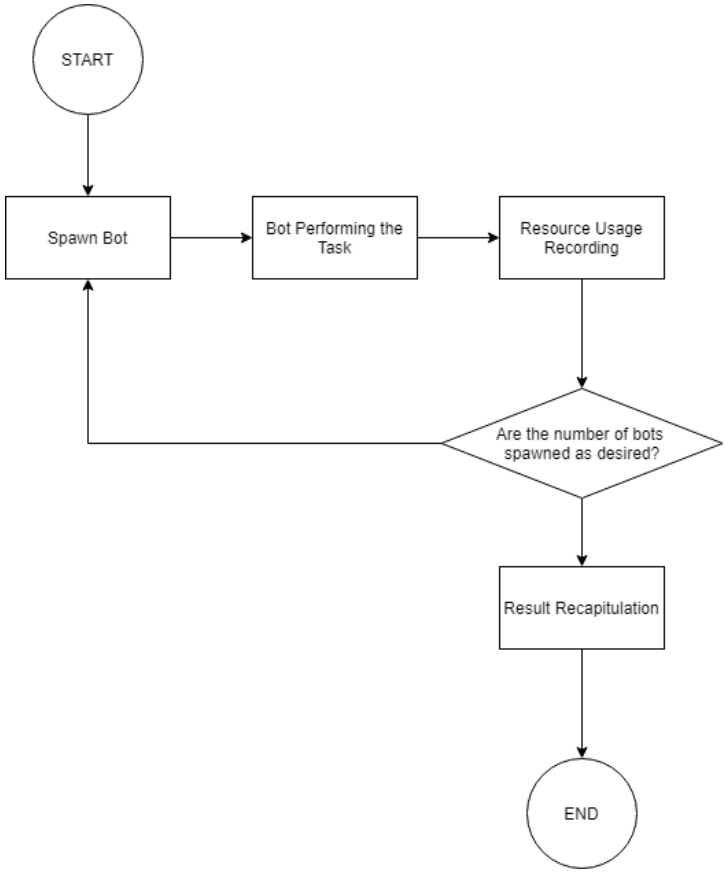

Figure 4. Workflow of task execution

the server, make it easier to observe the increase in the resources used, and observe the effect on server performance.

Bot Performing Task

After the bot spawns in the game, the next step is executing the task created in the previous script. Every bot will execute tasks according to a certain difficulty level, namely Idle (Easy), Dig (Medium), and Pathfinding (Hard) to overload the server. The scenario performed on each task is as follows.

- Idle: The task will make the bot after spawn does nothing. It is an easy task since bots just sit idle after the spawn.

- Dig: A task that will make bots dig or take blocks in the game according to what is instructed. The bot needs to find the block instructed at the closest distance. If no block is intended, the bot will give a message.
- Pathfinding: A task that will make the bot follow the player instructed continuously until a certain time. It is difficult because the bot must follow the player until a certain time with a stopping point that always changes when the player moves.

Resource Usage Recording

After the bot executes the task instructed, there will be computational resources to execute the task. From each task that is executed during the test, it will load the server as well. Thus, when testing is being carried out, there will be a record of server performance and the use of computational resources in which the results will be observed.

Results Recapitulation

After the test is complete, the results will be collected from each recording of the test results, either from the computational resources used or from the server performance during the test. In addition, the data that has been obtained will be tidied up into a document so that in the next stage, the evaluation will be easier to do.

6) Evaluation

After the data from the test results are recapitulated into a document, the next step is evaluating the results of the data obtained. The evaluation is to observe and find out the results of the server, whether when tested with the given task, the server has good performance or not and how much resource is used to carry out the tests.

III. Results and Discussion

A. Human Comparison

Figure 5 resumes the comparison between the results of bots and the manual approach for testing the resource of the server with five players online for 30 seconds. It's apparent that the median and averages between a bot and manual approaches don't differ significantly in all tasks. Dig tasks show the largest differences both in median and averages with 6.03% differences of average and 5% differences of the median. The pathfinding task shows 5.84% of differences in average and 1% in the median. The idle task has the lowest differences with only 3.6% average differences and 2% median differences.

On the other hand, the result from the manual approach shows more variance in data. The idle task shows usage of the server's CPU ranging from 28% to 99%. Pathfinding task shows usage of 3% to 57%. Finally, the dig task uses 3% to 49% of the server's CPU. On the contrary, bot testing doesn't show much variance in usage. This finding may need to be explored further for the improvement of the bot.

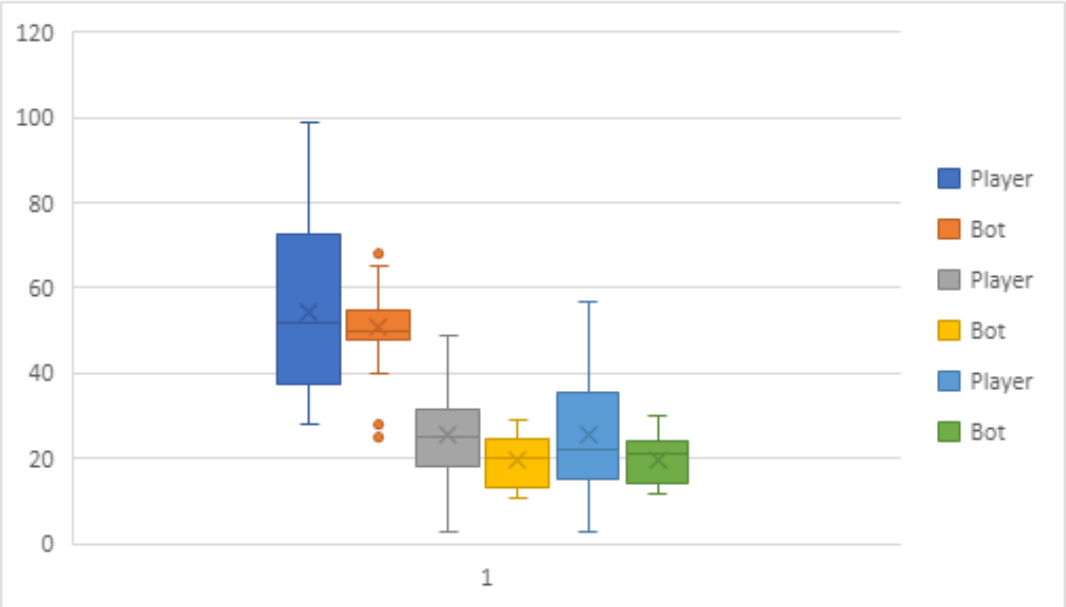

Figure 5. Comparison result

TABLE 3
AVERAGE CPU USAGE OF CLIENT AND BOT

	5 Bots	10 Bots	15 Bots	20 Bots
Idle	53.875	51.65079365	50.75294118	67.70689655
Dig	47.71875	68.77419354839	72.09411765	72.68103448276
Pathfinding	61.75	64.13114754	73.65882353	76.0862069

TABLE 4
THE AVERAGE USAGE OF RESOURCES FOR BOT

	5 Bots	10 Bots	15 Bots	20 Bots
Idle	14.26875	15.97777778	16.29176471	19.75689655
Dig	15.028125	19.18888889	19.91647059	24.08103448
Pathfinding	16.228125	21.87419355	24.40823529	24.44051724

TABLE 5
THE AVERAGE USAGE OF SERVER RESOURCES

	5 Bots	10 Bots	15 Bots	20 Bots
Idle	15.6875	39.87301587	19.15294118	21.85344828
Dig	15.8125	22.71428571	21.21176471	23.26982759
Pathfinding	13.53125	22.14754098	23.85882353	27.38793103

B.Resource Usage

The results of this study show that there are two things that greatly affect the level of use of computational resources, namely the number of bots and the complexity of the tasks. Table 3 shows resource usage in the pathfinding and dig tasks has increased linearly; however, during idle, there have been fluctuations in resource usage. From each task, it can be seen that resource usage is increased with the number of utilized bots. Therefore, it can be concluded that the more bots are performing the tasks, the greater the use of computational resources.

The results of the amount of resource code used to carry out the tasks were analyzed. As the tasks become harder, the value of the computational resource used will be higher, and it will cause the game to consume resources more heavily. As previously explained, there were three tasks tested, namely Idle, Pathfinding, and Dig tasks. Table 4 shows the resource code usage when running tasks. Pathfinding is consuming more resources other than dig and idle tasks throughout different numbers of bots employed. Furthermore, the increase in Pathfinding resource usage is quite significant from each increase in the number of bots compared to idle or dig tasks and is always the largest in their resource usage. Therefore, it can be concluded that the more complex the task for the bot, the better it is for testing the game server.

Table 5 shows the increase in resource usage of each task and the number of specific bots that pathfinding increases linearly. Unlike the other two tasks, the use of idle resources increased quite drastically from the use of five to ten bots, with a difference of 24.18551587. In addition, idle tasks decreased quite drastically from ten to fifteen bots with a difference of 20.72007469. Dig task also experienced fluctuations in resource usage but not as drastic as at idle task. However, at 20 bots, Pathfinding tasks used the most resources. Therefore, it can be said that the number of bots and the complexity of the tasks tested affect the use of server resources.

C.Discussion

For the answer to RQ1, the results indicate that bot-approach testing may be able to emulate the normal activities of real players for a long time. This is based on the findings that averages and medians of the results between bots and players are not significantly different. However, the result also suggests that players tend to do their tasks more dynamically. This dynamic approach in executing tasks is not properly emulated by our bots. Further improvement in the bot's behavior may be needed to better capture those dynamics.

For the answer of RQ2, the results indicate that the bot approach shows a linear relationship between the resource utilized for running the bot script and the number of server's CPU exhausted for the test. This means that bot-approach can be highly scalable with limited resources for running the bots. However, the resources for running the bot are also utilized for running the client of the game itself. Those clients may limit available resources for additional bots, which restricted the scalability of the bot itself. A specific client for running bots may help in ensuring the scalability of bot-approach testing.

D. Threat to Validity

There are two external threats to the validity of this research. The first one is the number of human testers for comparison between a bot and manual approach. This is caused by the availability of participants in a single time window. To help counter this, we try to ensure their experiences with playing the game used as a case study in this research.

The second one is the availability of computational resources utilized in this research. This research is conducted in a constricted testing environment, allowing limited resources to be accessed for the experiments. However, we tried to execute the measuring process as precisely as possible to achieve the most accurate results for this research.

IV. CONCLUSION

This research shows that the bot approach can be a concrete option in testing the capacity of a game server. The bot approach can emulate normal usage of the game server resources of a human. However, there are limitations in capturing the dynamics of a real human player. This limitation needs to be considered carefully for the bot approach to be implemented successfully. On top of that, given the scale of the task, the bot approach can be scalable in testing servers with high capacity. For it to be scalable, however, a special type of client specialized for testing may be needed.

In the future, there are several directions that can be taken in this research. The first one is an improvement of the bot to better emulate the dynamics of real players. Different approaches to script the bot can be researched, such as capture-and-replay and artificial intelligence-based bot. The other direction is to look further into the test environment setup itself. A proper test environment may reduce the resource usage for the execution of the bots. Developing a special type of client can be done in this direction.

REFERENCES

[1] Hsu, C. A., Ling, J., Li, Q., & Kuo, C. J., "The design of multiplayer online video game systems", Proc. SPIE 5241, Multimedia Systems and Appli-cations VI , 2003.
[2] S. Ariyurek, A. Betin-Can and E. Surer, "Automated Video Game Testing Using Synthetic and Humanlike Agents," in *IEEE Transactions on Games*, vol. 13, no. 1, pp. 50-67, March 2021.
[3] R. Ramadan and B. Hendradjaya, "Development of game testing method for measuring game quality," *2014 International Conference on Data and Software Engineering (ICODSE)*, 2014, pp. 1-6.
[4] Q. Zhou, C. J. Miller and V. Bassilious, "First Person Shooter Multiplayer Game Traffic Analysis," *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 195-200.
[5] S. M. Shariff, H. Li, C. Bezemer, A. E. Hassan, T. H. D. Nguyen and P. Flora, "Improving the Testing Efficiency of Selenium-Based Load Tests," *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, 2019, pp. 14-20.
[6] ISO 25010. (n.d.). ISO 25000 PORTAL. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.
[7] Garousi, V., Briand, L. C., & Labiche, Y, "Traffic-aware stress testing of distributed systems based on UML models", *In Proceedings of the 28th international conference on Software engineering (ICSE '06)*. Association for Computing Machinery, New York, NY, USA, 2006, 391–400.
[8] Jiang, Z. M, "Automated analysis of load testing results", *In Proceedings of the 19th international symposium on Software testing and analysis (ISSTA '10)*, Association for Computing Machinery, New York, NY, USA, 2010, 143–146.
[9] Ghalistan, G., Widowati, S., & Husen, J. H., "Pengembangan Kakas Pengujian Formal Detail Otomatis untuk Aplikasi Permainan Mobile", 2020.
[10] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," *Future of Software Engineering (FOSE '07)*, 2007, pp. 85-103.
[11] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 77-81.
[12] S. Thummalapenta, S. Sinha, N. Singhania and S. Chandra, "Automating test automation," *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 881-891.
[13] Sagi, B. R., & Silvestrini, R., "Application of combinatorial tests in video game testing. Quality Engineering", 29(4), 2017, 745-759.
[14] C. Cho, D. Lee, K. Sohn, C. Park and J. Kang, "Scenario-Based Approach for Blackbox Load Testing of Online Game Servers," *2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2010, pp. 259-265.
[15] E. Enoiu, D. Sundmark, A. Čaušević and P. Pettersson, "A Comparative Study of Manual and Automated Testing for Industrial Control Software," *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2017, pp. 412-417.
[16] I. Dobles, A. Martínez and C. Quesada-López, "Comparing the effort and effectiveness of automated and manual tests," *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, 2019, pp. 1-6.
[17] Divya Kumar, K.K. Mishra, "The Impacts of Test Automation on Software's Cost, Quality and Time to Market", Procedia Computer Science, Volume 79, 2016, Pages 8-15.
[18] Patidar C. P, Dongre A. D., "An Automated Testing Model using Test Driven Development Approach. Orient", J. Comp. Sci. and Technol, 10(2), 2017, 385-390.
[19] M. Cocar, R. Harris and Y. Khmelevsky, "Utilizing Minecraft bots to optimize game server performance and deployment," *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2017, pp. 1-5.
[20] T. Alstad et al., "Minecraft computer game performance analysis and network traffic emulation by a custom bot," *2015 Science and Information Conference (SAI)*, 2015, pp. 227-236.
[21] T. Alstad et al., "Game network traffic simulation by a custom bot," *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, 2015,

pp. 675-680.
[22] Andrewrk. et al., (2016) Prismarinejs/mineflayer. GitHub. [Online]. Available: https://github.com/PrismarineJS/mineflayer