UDC 004.021
DOI: 10.24025/2306-4412.3.2023.286553

# SPADE SOFTWARE AGENTS AND THEIR IMPACT ON HARDWARE RESOURCES

**Eduard Zelenko**
Ph. D. Student
Cherkasy State Technological University
460 Shevchenko Blvd., Cherkasy, 18006, Ukraine
https://orcid.org/0000-0002-9939-3830
**Yevheniia Kataieva**
Ph. D., Associate Professor
Cherkasy State Technological University
460 Shevchenko Blvd., Cherkasy, 18006, Ukraine
Slovak University of Technology in Bratislava
Vazovova 5, 812 43 Bratislava 1, Slovak Republic
https://orcid.org/0000-0002-9668-4739

**Abstract.** Eliminating the disadvantages of updating prices with a large number of products in the online store, we have found a solution in the application of the Smart Python Agent Development Environment (SPADE). The article presents the process of collecting data on SPADE and Openfire Server performance metrics in order to determine and analyze the consumption of system resources when connecting software agents with different types of behavior, in different numbers, as well as during interaction with a web application.

In current study, JMeter is used as a tool for data collection and performance testing (including load and stress). Quantitative and qualitative methods of data analysis are used. When processing the collected values of indicators for the use of hardware resources, methods of mathematical statistics have been used to identify relations between indicators. To compare the behaviors of the SPADE software agent, to determine the effectiveness of one over the other, as well as to determine the effectiveness of using the agent interface in command line mode compared to its web counterpart in the form of a graphical user interface (in terms of performance), formulas for calculating the growth rate are used.

During the study, the advantage of SPADE in the speed of program code execution; the difference in performance between agent behaviors, as well as between agent web user interface (AWUI) and command line interface (CLI) modes; features of using the CLI mode of the agent for interactive user interaction with the application in order to quickly fix errors that occur during the interaction of the agent with the web application have been determined.

Integration of SPADE agents into the pricing process has practical implications for retailers, opening up opportunities to study and develop new tools for subsequent application in solving specific problems.

**Keywords:** XMPP, Python, behavior, web, JMeter, CPU, RAM.

## Introduction

Solving the problem of pricing in the absence of up-to-date data, an integrated approach was used using correlation and data monitoring through the application programming interface (API) (Zelenko, 2022). Despite the profitability, the developed software prototype (the minimum viable product (MVP) (Graffius, 2023; Umbreen *et al*., 2022), which was used by the German company for two years (Zelenko *et al*., Classification, 2023), had several limitations, the main of which was associated with a large time spent on executing the program code for updating prices (≈100-110 s) and, as a result, an increase in the load on the web server. Other important disadvantages include the cost of additional time and financial


DOI: 10.24025/2306-4412.3.2023.286553

resources for the development and support of: a method for reporting errors that require a human decision; graphical user interface (GUI) for making these decisions as well as for interacting with the software during the price update process.

The capabilities of the technologies used did not allow eliminating the first disadvantage and made it difficult to eliminate the second one, so it was decided to make changes to the application architecture. In the process of searching for new technologies, a solution was found, which consisted in the use of SPADE.

The **goal** is to study the impact of SPADE software agents on PC hardware resources and determining the effectiveness of their use compared to the web user interface (UI) in terms of performance. The **tasks** are: collect data on XMPP server performance metrics during its testing (including load and stress testing); determine the amount and analyze the consumed hardware resources over a certain time interval when connecting SPADE software agents with different basic types of behavior, in different quantities, as well as during interaction with a web application when performing the main function – updating prices.

## Materials and methods

Software agent (hereinafter – agent) can be considered as a software object that can exhibit the above multi-dimensions or as an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its designed objectives. Researchers consider agent-oriented development as a natural and a logical evolution of the current software development paradigms, such as structured and object-orientated approaches, as it presents a higher-level of abstraction and encapsulation. Agent-oriented approaches can significantly enhance our ability to model, design and build complex, distributed software systems (Slhoub, 2018).

Agent orientation represents a higher-level abstraction that is more flexible than the prior programming paradigms, such as object abstraction. The agent technology is going to be the next significant breakthrough in software development and the new revolution in software (Slhoub, 2018). The software agents and multi-agent systems (MAS) (Holgado-Terriza *et al*., 2020) will be one of the landmark technologies in computer science that will bring extra theoretical power, methods and techniques that will broaden the field of computer applications (Ajitha *et al*., 2016).

Smart Python Agent Development Environment (SPADE) is a free MAS platform written in Python (n.d.) under MIT license and based on instant messaging (XMPP) (Spade-BDI, n.d.; Spade, n.d.; Pal *et al*., 2020; Palanca, 2018; SPADE – SPADE 3.3.0, n.d.; Donancio *et al*., 2019). And since Python is the language widely used in most machine learning framework (Lyu *et al*., 2020) as well as in web development, this will allow the software (developed using the SPADE platform) to be easily integrated into a large number of web applications, as well as to use the software in conjunction with a large number of machine learning (ML) libraries provided by Python.

Some features: flexibility, agent model based on behaviors, web-based interface, use any XMPP server, BDI support (Spade, n.d.; Pal *et al*., 2020; Palanca, 2018; SPADE – SPADE 3.3.0, n.d.; GitHub - javipalanca/spade_bdi, n.d.; Palanca *et al.*, 2022; Palanca *et al.*, 2023; Pérez, 2023). Behavior-based architectures are conceptually more capable, since they remove some of the limitations of reactive systems (Palanca *et al.*, 2022).

Five most commonly used SPADE behavior types were chosen for the experiment: Cyclic, Periodic, One-Shot, Time-Out, The Finite State Machine (FSM) (Palanca, 2018; Advanced Behaviours, n.d.; Palanca *et al*., 2020; The SPADE agent model, n.d.).

The SPADE technology was chosen for the MVP upgrade for a complex reason that includes fundamental benefits (some of which are driven by Python):


DOI: 10.24025/2306-4412.3.2023.286553

1. An increase in the speed of script execution (described in the course of the current study).
2. No need to increase the script execution time and memory limit, reducing the load on the web server.
3. The possibility of interactive interaction with the agent (in this case, in the process of updating prices), which allows a person to quickly make a decision in case of errors or situations when the decision cannot be made by the agent.
4. Reduction of time and, as a result, ceteris paribus, financial costs due to the absence of the need to develop and maintain a graphical user interface (GUI) for software agents, compared with the development and support of such for web applications. Nevertheless, the SPADE technology has a built-in GUI, which, if necessary, allows you to speed up interaction with the agent and expand its functionality.

To implement the 3rd point, when executing the price updating algorithm (PUA; which uses cost-based method (CBM) for goods in the amount of 47817 units) in PHP/Python instead of an agent, for example, through cron, it is necessary to develop an alert functionality that will be limited in capabilities compared to the potential of SPADE BDI. Also, in the case of using PHP, GUI development will be required, however, the CLI mode UI for the SPADE agent is implemented faster, providing the user with quick access as a separate, more distinct system process (both visually and programmatically; compared to the web GUI). In the case of using SPADE, the notification option may be needed only if a user needs to make a decision remotely.

The following advantages also served as a reason for choosing this technology:

1. Potential for software development using Artificial Intelligence (AI) or Distributed Artificial Intelligence systems (DAI; due to the use of Python and agent technology (agent-oriented software engineering (AOSE) (Slhoub, 2018), the link for which is SPADE). It is planned to apply in the next version of the software in the form of monitoring the prices of competitors, allowing the use of the competitive-based method (Zelenko *et al*., Overview, 2023).
2. Ability to delegate user requirements to an agent (Jubilson *et al*., 2016), allowing the agent to make decisions independently.
3. The ability to develop software at a higher level of abstraction, which is more flexible than the prior programming paradigms (Slhoub, 2018).
4. Widespread use of its programming language (Python).

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual ones. In simple terms, Software Testing means Verification of Application Under Test (AUT). Testing is important because software bugs could be expensive or even dangerous. It can be either done manually or using automated tools (Nordeen, 2020).

During manual testing, there are a lot of problems faced by testers like it is very time taking and consuming, no reusability, has no scripting feature, much human effort required, and still major or minor bugs remain unexplored. Therefore to cover all types of errors and bugs automation testing has introduced that explores all the issues exist in manual testing. Automation testing is 70% faster than the manual testing, ensures consistency, saves cost, improves accuracy, increases efficiency, et al. All automation testing tools test software in less time, produce more reliable, repeatable and reusable final product (Abbas *et al*., 2017; Nordeen, 2020).

Performance testing is defined as a type of software testing to ensure software applications will perform well under their expected workload. It is also performed in order to


DOI: 10.24025/2306-4412.3.2023.286553

determine how fast system responds under a particular workload. The goal of this testing type is not to find bugs, but to eliminate performance bottlenecks (Dilshan de Silva *et al*., 2023). It is done to provide stakeholders with information about their application regarding speed, stability and scalability (Nordeen, 2020; Mokhamd *et al*., 2023). And since these three components were considered important in the development and maintenance of the MVP, this type of testing was chosen for the experiment.

Performance testing is an important and necessary test process to ensure that there are no problems with the performance of the product, such as: long download time, poor response time and poor scalability. In addition, these types of testing activities help to identify bottlenecks that interfere with the operation of the hardware-software platform of the product resource and can lead to significant delays in performance and even to the failure or crash of the resource as a whole (Ali *et al*., 2020).

The focus of the performance test: Scalability determines maximum user load the software application can handle; Stability determines if the application is stable under varying loads; Speed determines whether the application responds fast enough (Nordeen, 2020; Mokhamd *et al*., 2023).

Methods used to implement performance testing are load and stress testing (they are performance testing techniques). Load testing is necessary to know that a software solution will perform under real-life loads (Nordeen, 2020; Mokhamd *et al*., 2023). It measures the response of a system under various load conditions. This test helps determine how the software behaves when multiple users are using the software at the same time. Load testing is required to simulate concurrent usage. Stress uses virtual users who exceed the maximum number until system/application downtime occurs and is typically used for a longer period of time. It is needed to test stability and reliability of the system or it can be called as testing system durability (Mokhamd *et al*., 2023). Both performance and therefore load testing are categorized as non-functional testing (the technique which focuses on testing of a software application for its non-functional requirements) (Nordeen, 2020).

In any case, exhaustive testing is not possible. Instead, an optimal amount of testing is needed, based on the application's risk assessment. The answer to the question "How to determine the risk?" leads to the defect clustering principle (the application of the Pareto principle) (Nordeen, 2020).

The layout and hardware/software list of the tested environment are shown in Figure 1 and in Table 1.

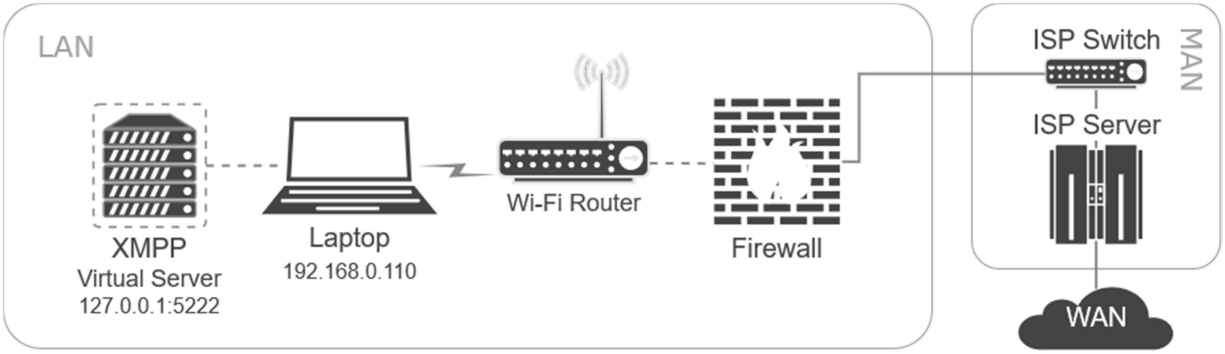


**Figure 1.** Testing environment network architecture layout (based on (Ali *et al*., 2020)

JMeter was chosen as the most suitable and effective performance (including load and stress) testing tool based on competitive analysis results and comparison testing tools


DOI: 10.24025/2306-4412.3.2023.286553

(including those for web services) (Ali *et al*., 2020; Abbas *et al*., 2017; Dilshan de Silva *et al*., 2023).

**Table 1.** Hardware/Software test environment

| # | Required hardware/software | Description/Type of hardware/software |
|---|---|---|
| 1 | Hardware | Laptop: AMD Ryzen 5 4500U with Radeon Graphics, 2.38 GHz, RAM: 16.0 GB (13.9 GB usable), SSD 512 GB. |
| 2 | Operation System | Microsoft Windows 10 Pro, Version 20H2 (OS Build 19042.1466), 64-bit. |
| 3 | An Instant Messaging (IM) server | Openfire (XMPP) Server v4.7.4, build 51b9db9. |
| 4 | Execution environment language | Java (JRE/JDK v1.8.0_202). |
| 5 | Agent Development Environment | SPADE v3.2.3 (Palanca, 2023; SPADE, n.d.) |
| 6 | Agent Development Environment programming language | Python v3.7.9. |
| 7 | Test scenario recording/execution tools | Apache JMeter v5.5 (plugins: XMPP Protocol Support (bzm) v1.5.1, 3 Basic Graphs v2.0, jp@gc - PerfMon Metrics Collector v2.1, Dummy Sampler v0.4). |
| 8 | Server metrics fetching tool | PerfMon Server Agent (JP@GC) v2.2.3. |
| 9 | Browser | Mozilla Firefox v114.0.2 (64-bit). |

The Apache JMeter™ application is open source software, a 100% pure Java application that is considered an automated tool for testing performance, load and stress as well, and along with BlazeMeter are considered sufficient automation tools to accelerate and utilize performance-testing processes (Abbas *et al*., 2017; Dilshan de Silva *et al*., 2023; Mokhamd *et al*., 2023; Ali *et al*., 2020; Apache JMeter, n.d.). And since load and stress testing are methods used to implement performance testing, JMeter can be used to implement all of them, as well as to perform Application Performance Monitoring (APM) refers to managing the performance of a software application to ensure the expected level of service, as measured by performance metrics and user experience monitoring. The APM solution aims to detect and determine application performance issues before users are actually impacted by them (Mokhamd *et al*., 2023).

It was originally designed for testing Web Applications but has since expanded to other test functions (Apache JMeter, n.d.). JMeter integration with Agile and DevOps processes improves tool utilization (Ali *et al*., 2020).

As a tool for collecting data on CPU/RAM (Central Processing Unit and Random Access Memory) metrics, in addition to Apache JMeter, the PerfMon Server Agent software agent was used (Table 1), sometimes called JP@GC Agent (where JP@GC means JMeter Plugins at Google Code) (GitHub - undera/perfmon-agent, n.d.; Documentation: JMeter-Plugins.org/wiki/PerfMon/, n.d.; Matam *et al*., 2017; Pohilko, n.d.). It is a server metrics fetching agent for server performance monitoring, written in Java and based on SIGAR (System Information Gatherer And Reporter (GitHub - hyperic/sigar, n.d.).

All tests were preceded by smoke testing (Nordeen, 2020). Objects, types, test tools and their descriptions are provided in Table 2.

Based on 10 results of the price update function execution (Figure 2), using the Dummy Sampler plugin (Documentation: JMeter-Plugins.org/wiki/DummySampler/, n.d.), the average value $\bar{t}_0 = 104.46$ was determined. Taking into account that the minimum required time for this operation is more than 100 s, it was decided to collect data within this time interval (always from the beginning of the update process; thus ensuring data normalization for their subsequent analysis).


DOI: 10.24025/2306-4412.3.2023.286553

**Table 2.** Test plan for XMPP server testing

| # | Testing object | Testing type | JMeter testing tool | Description / Features |
|---|---|---|---|---|
| 1 | XMPP server | Performance | Dummy Sampler | Difference between two conditions (when the agent is running and not). |
| 2 | XMPP server | Load | OS Process Sampler (JMeter - OS Process Support) | While connecting a large number of agents. |
| 3 | XMPP server | Load | bzm - XMPP Sampler (XMPP Protocol Support) | XMPP Connection. |
| 4 | XMPP server | Performance/Load | Dummy Sampler | Load comparison on XMPP server with different agent behaviors (outside and during the price update process). |

A comparison was made (according to the average values of the speed of the price update process) between three variants of the architecture of the software being developed, which differ in the location of the PUA: on the side of the web server with a call via cron (a test run through the browser is shown in Figure 2); with the same location, but with a call through the SPADE agent (Figures 3, 4; this architecture was used for most tests); PUA is aggregated by the SPADE agent (Figure 5). The script execution speed in the second case is approximately equal to that for the first one, which is natural, given the invariability of the PUA location. This architecture option allows you to quickly incorporate SPADE technology into a web project (approximately within 5 minutes). The script execution speed in the third case is $\bar{t}_1 = 43.24$.

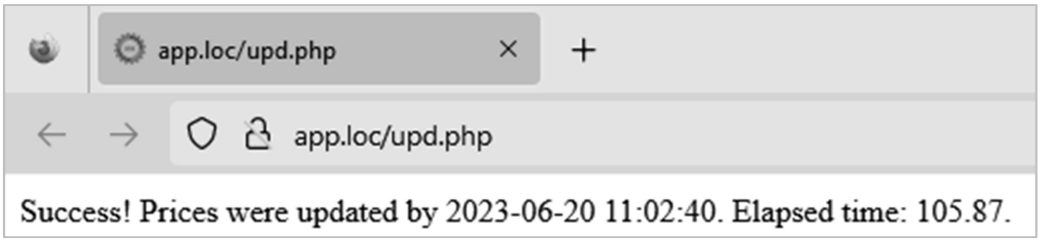


**Figure 2.** Pricing update information message displayed as a result of a request via the Mozilla Firefox web browser

```
Agent starting...
Wait until user interrupts with Ctrl+C
Starting behaviour...
Counter: 0
There's no new file!
Counter: 1
Counter: 2
```

**Figure 3.** Launching the agent

```
Counter: 321
Counter: 322
Counter: 323
Counter: 324
Counter: 325
Success! Prices were updated by 2023-06-20 11:35:35. Elapsed time: 106.729.
```

**Figure 4.** Successful price update by an agent through a request to an MVC controller aggregating PUA


DOI: 10.24025/2306-4412.3.2023.286553

```
Counter: 79
Counter: 80
Success! Prices were updated by 2023-06-20 12:13:14. Elapsed time: 42.74.
```

**Figure 5.** Successful price update by PUA aggregating agent

Let us substitute the average values of the indicators of the execution time of the price update process $\bar{t}_0, \bar{t}_1$ displayed in Figures 3 and 5 into (1) (percentage decrease, % decrease) in order to determine the efficiency. Thus, we determine $\Delta M(\%) = 58.61$. An additional run of the script outside the agent showed the same results, therefore, this growth rate is due to the difference between the speed of executing PHP and Python code, and not the work of the agent.

$$\Delta M(\%) = \frac{M_{n-1} - M_n}{M_{n-1}} \times 100\%, \quad (1)$$

where $M$ is mean, $M_n$ is n[th] sequence term ("final" value), $M_{n-1}$ is a previous term in sequence ("initial" value), $M_{n-1} - M_n$ is $\Delta M$ or decrease. (1) is based on (2) (percentage increase, % increase) (Dueñas *et al*., 2021; Mahadevan *et al*., 2022) to determine growth and growth rate (comparative performance indicator). In the case of a negative value of the denominator, it must be taken modulo.

$$\Delta M(\%) = \frac{M_n - M_{n-1}}{M_{n-1}} \times 100\%, \quad (2)$$

or $(M_n/M_{n-1} - 1) \times 100\%$, where $M_n - M_{n-1}$ is $\Delta M$ or increase. The test results are shown in Figures 8, 9.

Let us substitute the CPU and RAM load indicators displayed in Figure 9 into (2) in order to determine the increase and % increase. Results are shown in Table 3.

**Table 3.** Increase display for average CPU and RAM load (Figure 9)

| Metrics / Agent using features | CPU, % | | | RAM, MB | | |
|---|---|---|---|---|---|---|
| | $M$ | $\Delta M$ | $\Delta M(\%)$ | $M$ | $\Delta M$ | $\Delta M(\%)$ |
| Without using an agent | 0.08 | — | — | 287.32 | — | — |
| With using an agent | 0.18 | 0.1 | 125 | 289.07 | 1.75 | 0.61 |
| With using an agent while price update | 0.19 | 0.01 | 5.56 | 290.6 | 1.53 | 0.53 |

Based on the values of $\Delta M$ and $\Delta M(\%)$ from Table 3, it is concluded that the use of hardware resources (by the XMPP server during connection with the agent) is negligible in terms of modern PC computing power.

To determine the effectiveness of SPADE in terms of hardware resource usage (HRU) for user PC, the SPADE web interface was chosen as an object for comparison (Palanca *et al*., 2020) (agent web UI (AWUI) (Palanca, 2017).


**Figure 6.** Detailed measurements of Firefox's memory usage via "about:memory" pecial page (about:memory, n.d.) for the SPADE web interface tab


DOI: 10.24025/2306-4412.3.2023.286553

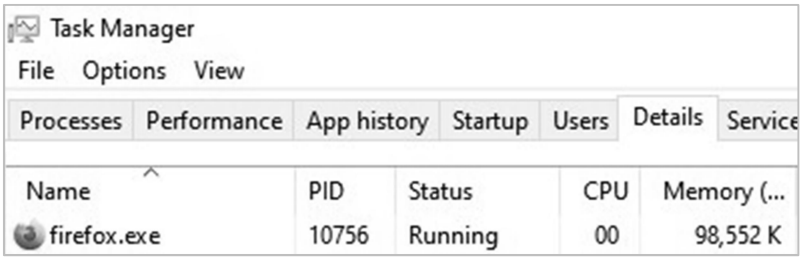


**Figure 7.** System process firefox.exe (i.e., browser tab) with PID 10756

According to Figure 10, the mean for CPU used resources is 0.16%, for RAM is 98.21 MB. Since the server does not use the hardware resources of the user's PC, only the resources used by the agent (CLI mode) were taken into account to determine the efficiency. Substituting these values (as well as those for SPADE in CLI mode: CPU – 0.2%, RAM – 85.77 MB) into (1), we get the efficiency when the SPADE agent works in CLI mode compared to AWUI: for CPU – (0.16 - 0.2) / 0.16 * 100% = -25%, for RAM – (98.21 - 85.77) / 98.21 * 100% = 12.67%. It is worth noting that AWUI was used for the example, which was not loaded with script libraries. In the case of using a more complex GUI developed using modern front-end frameworks, it is expected to increase the efficiency of using SPADE agents in CLI mode compared to the web version of the UI.

The purpose of the next load test is to show the load on the XMPP server as it grows as agents connect. An exhaustive description of the "bzm – XMPP Connection" plugin settings can be obtained from the source (XMPP Load Testing, n.d.). The test results are shown in Figure 11.

The purpose of this test is to display the difference between HRU for different types of agent behavior; the task is to collect data on performance metrics for different behaviors of the SPADE software agent using the OS Process Sampler plugin. The results are displayed in Figures 12-14.

By ordering the average HRU values (Figure 14) and substituting them into (1), we obtain the results presented in Table 4.

**Table 4.** Comparison of percentage decrease for agent behaviors

| ΔRAM (%) | | | | ΔCPU (%) | | | |
|---|---|---|---|---|---|---|---|
| Out of the price update | | During price update | | Out of the price update | | During price update | |
| Behavior type | value | Behavior type | value | Behavior type | value | Behavior type | value |
| One shot | - | FSM | - | FSM | - | Cyclic | - |
| FSM | 0.40 | One Shot | 2.14 | One Shot | 10.26 | One Shot | 8.11 |
| Timeout | 2.18 | Timeout | 1.75 | Cyclic | 5.71 | FSM | 11.76 |
| Periodic | 1.05 | Periodic | 1.45 | Periodic | 21.21 | Timeout | 3.33 |
| Cyclic | 1.68 | Cyclic | 1.66 | Timeout | 15.38 | Periodic | 24.14 |

From Table 4, we can draw conclusions about the effectiveness of one agent's behavior compared to another. For example, the FSM behavior is 0.4% (RAM) more efficient than One Shot when the agent is running outside the price update process.

Stress testing of the launch and operation of the XMPP server, as well as a large number of agents (with sending requests to the Open Server) was performed 8 times (with a duration of 100 seconds for each iteration) with two settings - with a limit on the frequency


DOI: 10.24025/2306-4412.3.2023.286553

of launching agents ($f_l = 1$ Hz, where $f$ is the frequency, $l$ is the launch (abbreviated from the "launch")) and without it. The results are presented in Table 5 and Figures 15, 16.

**Table 5.** Stress test results

| ID | Number of launches | | | Result |
|---|---|---|---|---|
| | General | Successful | Failed | |
| 1 | 166 | 127 | 39 | JMeter crash. |
| 2 | 144 | 109 | 34 | |
| 3 | 1265 | 137 | 1128 | |
| 4 | 107 | 84 | 23 | |
| 5 | 98 | 76 | 22 | |
| 6 | 116 | 97 | 19 | JMeter crash, XMPP server crash. |
| 7 | 267 | 247 | 20 | Stable performance of applications (with the ability to save CPU/RAM load metrics). |
| 8 | 98 | 95 | 3 | |

During all tests, the Open Server remained in working condition (in order to save space and due to the repetition of the result for 8 tests, this information was not indicated in the table), which can be explained by the frequency of requests by each of the agents on the Open Server – $f_r = 0.2\,Hz$, where $r$ is a request (abbreviated from "request"), which is at least 5 times less than the frequency of agent launches (the values of the "General" column, indicated in Table 5, divided by 100). In addition, the load on the XMPP server increases not only at the moment the agent starts, but also during the processing of other processes (e.g., behavior). Testing Open Server is out of the scope of this work.

Having removed the main noise at the end of the 3rd test (after about 230 agent launches), in order to avoid excessive stretching of the plot along the Y axis and in order to more clearly view the test results, we display the launch errors on a scatter plot (Figure 17).

The reason for the appearance of errors in the application is presumably due to the consumption of RAM (XMPP server: 287.32 MB + 1.75 MB (for each agent); 85.77 MB (for each SPADE agent in CLI mode)), despite the fact that 70 software agents can occupy ≈6 -6.5 GB of RAM (provided that the agent and server are running on the same PC), which is acceptable in terms of the environment under test and modern computing power in general.

Based on the RAM load metric (Figure 16), namely the lack of indicators of hardware resource consumption after the 47th second during the 7th test, we can conclude that XMPP Server has crashed. Nevertheless, the application continued to successfully execute the function of launching agents, despite the stop of data collection by CPU load metric at the 51st second.

## Results and discussion

In the course of this work, the following were carried out:

1. Data collection on XMPP server performance metrics in order to analyze and determine the amount of hardware resources consumed over a certain time interval when connecting SPADE software agents with different basic types of behavior, in different amounts, as well as during the execution of their main function – price update.
2. Comparison (according to the average values of the speed of the price update process) between the three variants of the architecture of the software being developed, which differ in the location of the price updating algorithm (PUA).


DOI: 10.24025/2306-4412.3.2023.286553

3. Stress test of launching agents, on the basis of which a scatter diagram of launch errors was built.

In the course of performing a qualitative and quantitative analysis of the data, it was found that:

1. The percentage increase in the speed of the price update process using SPADE compared to the web version was 58.61% (due to the use of Python).
2. The hardware resources used by SPADE are acceptable in terms of modern PC computing power (on average for XMPP server: 0.08% of CPU and 287.32 MB of RAM + 0.1% of CPU and 1.75 MB of RAM for each agent; 0.2% of CPU and 85.77 MB of RAM for each SPADE agent in CLI mode). Efficiency (in terms of HRU) when running a SPADE agent in CLI mode compared to AWUI: for CPU – -25%, for RAM – 12.67%.
3. Mean/min/max values, medians and modes are determined based on performance data for different types of behavior of SPADE agents. Based on ascending averages, a percentage increase is determined for RAM/CPU resource consumption during and outside of the price update process. Based on these values, one can draw a conclusion about the effectiveness of one type of agent behavior compared to another (in terms of HRU).
4. Most launch errors falls on the set of active software agents in the amount of ≈70-240 units with the result in the form of hanging OS processes and with the subsequent fall of the JMeter and XMPP server applications. The limited frequency ($f_l = 1\ Hz$) of launching agents allows you to reduce the number of errors and application crashes.
5. The trend (steady upward) in Figure 11 differs significantly from that (partially downward/absent) in Figure 16 when agents are sequentially launched with a difference in frequency ($f_l = 1\ Hz$ for the first and $f_l = 0.31\ Hz$ for the second case). Based on the current and previous paragraphs, we can conclude that with a decrease in the value of $f_l$, the stability of applications and the accuracy of the collected data increase.
6. Because of the use of SPADE and, as a result, the absence of the need to develop a GUI, the time spent on developing a UI is reduced (hence the labor intensity; compared to developing one for web applications).

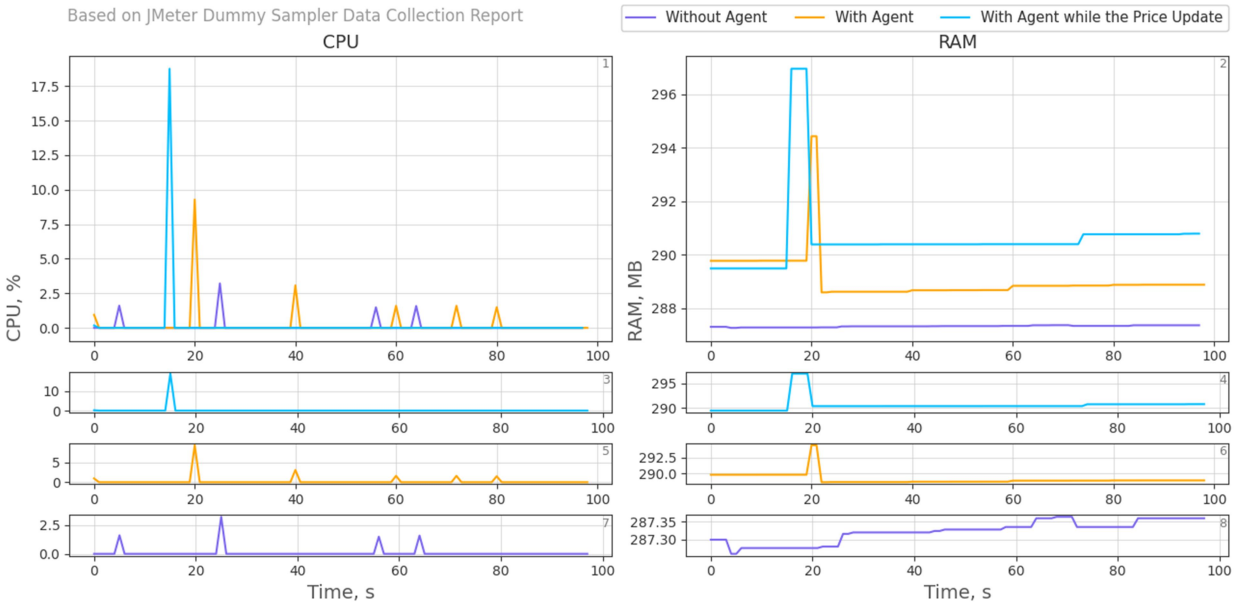


**Figure 8.** Metrics of hardware resource usage by the XMPP server process using JMeter Dummy Sampler


DOI: 10.24025/2306-4412.3.2023.286553

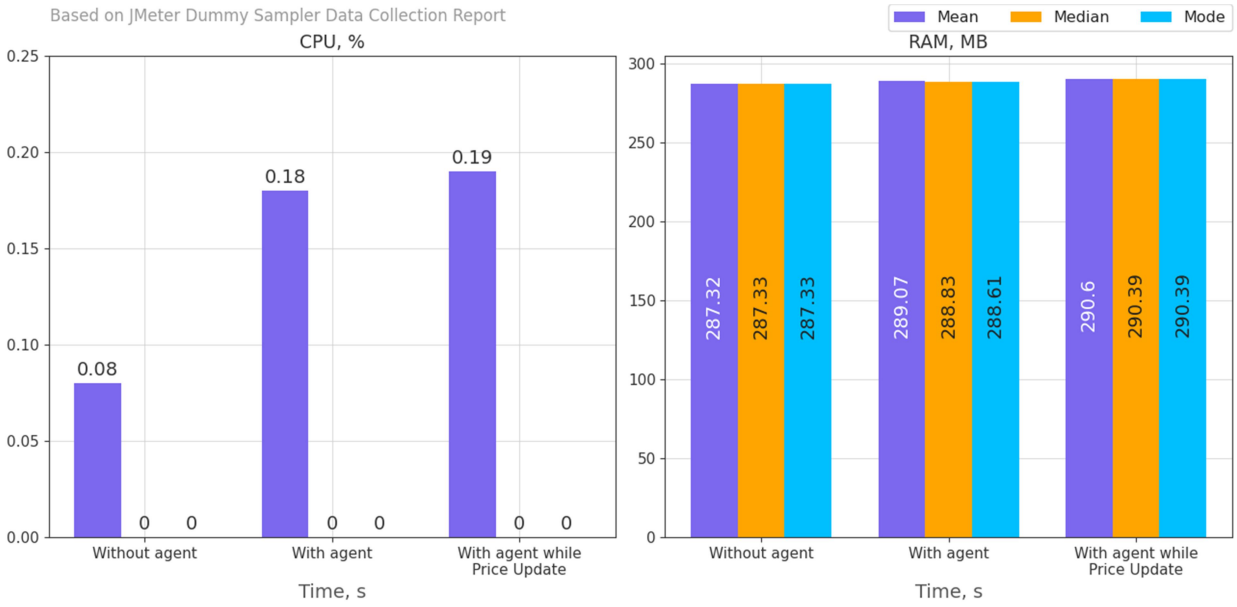


**Figure 9.** Mean, Median, Mode for the collected data shown in Figure 8

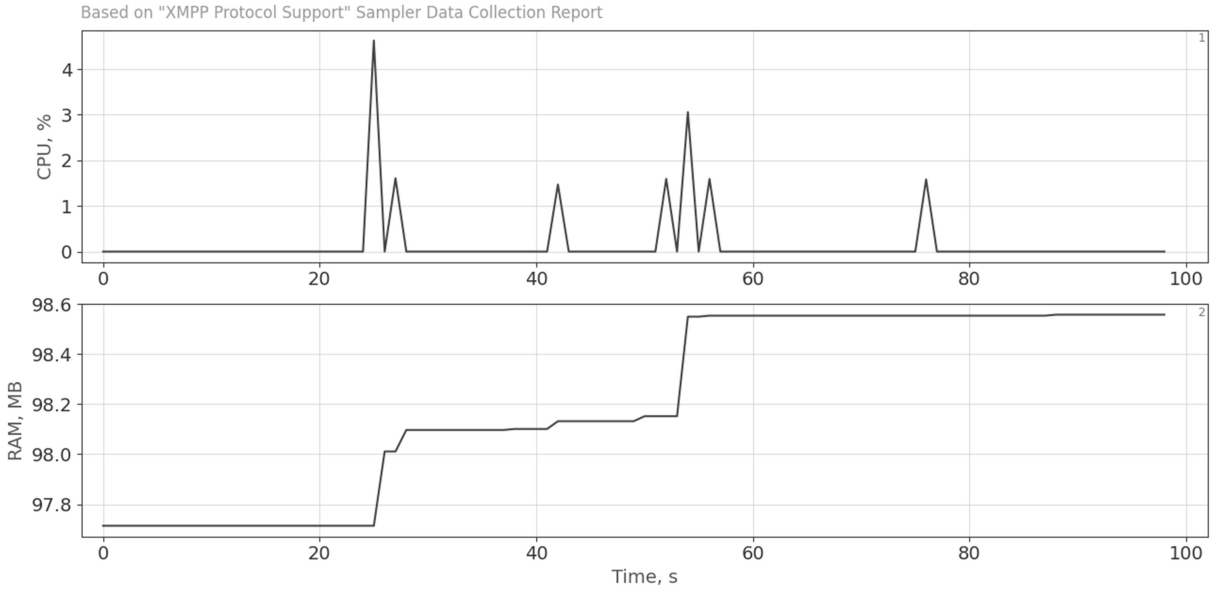


**Figure 10.** CPU and RAM usage measurements for the PID 10756 (browser tab that hosts the SPADE web GUI)


DOI: 10.24025/2306-4412.3.2023.286553

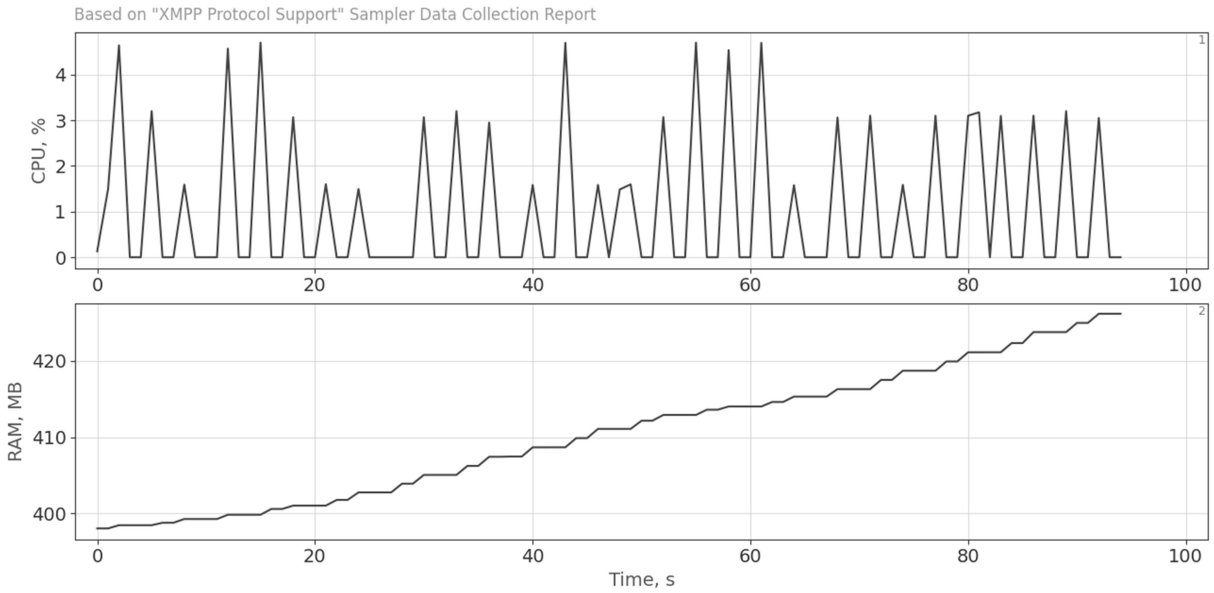


**Figure 11.** CPU/RAM performance metric for 32 agents as a line graph (each 3.2 s) with a steady upward trend in RAM consumption as the number of SPADE agents connected to the XMPP server increases

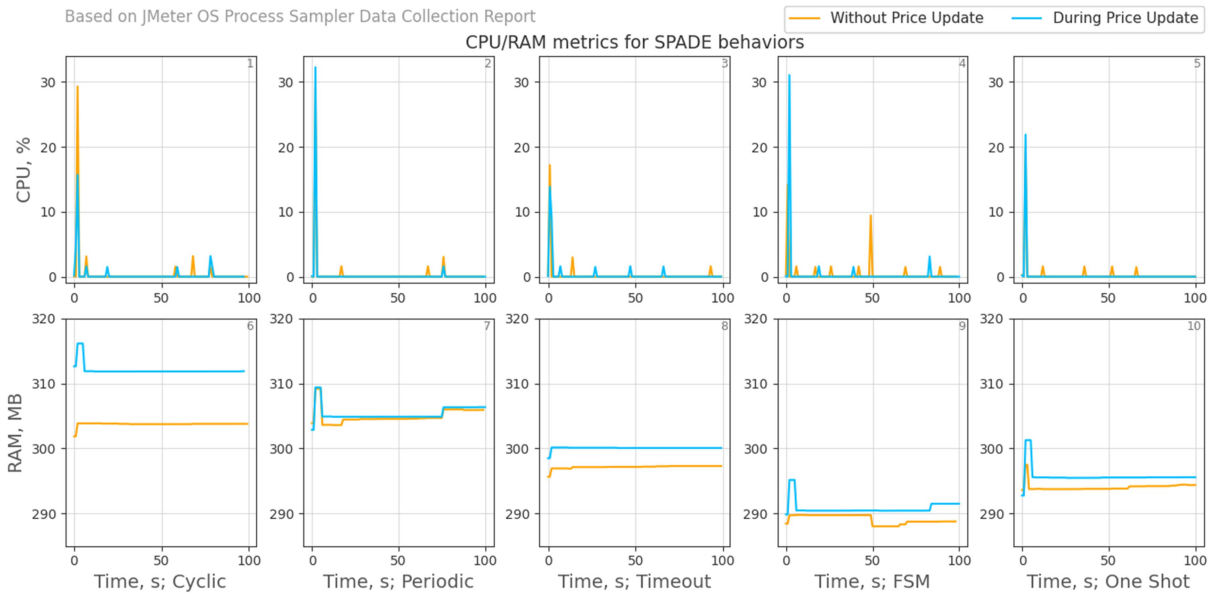


**Figure 12.** Line graphs for Comparison of HRU amount (CPU, RAM) during the price update process and outside it with different SPADE agent behaviors (FSM, One Shot, Timeout, Cyclic, Periodic)


DOI: 10.24025/2306-4412.3.2023.286553

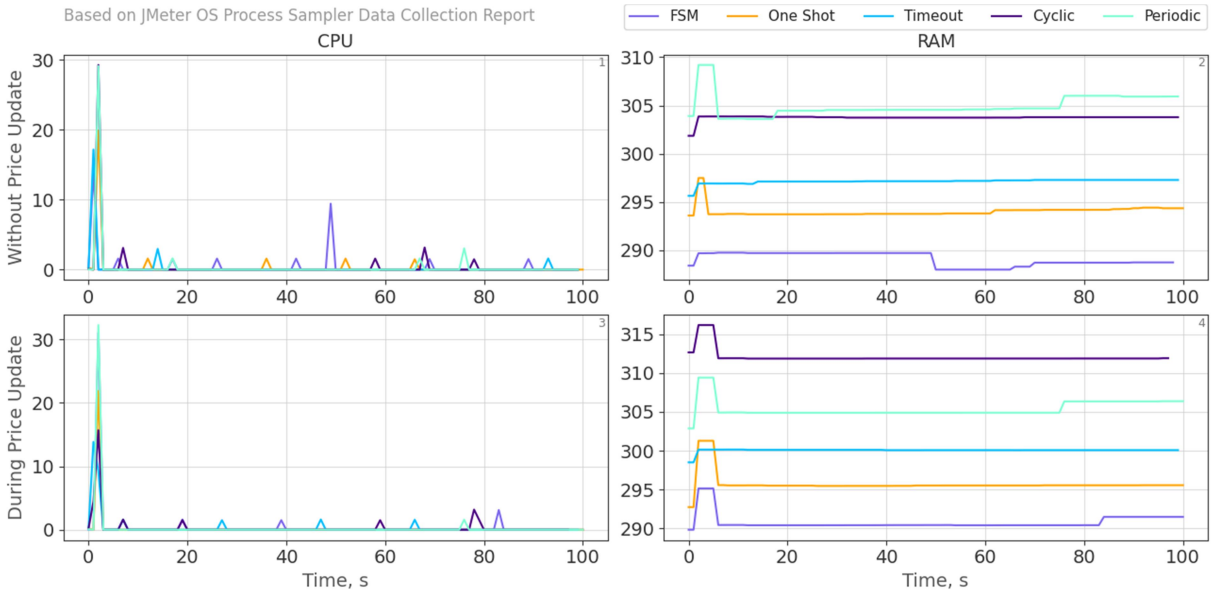


**Figure 13.** Line graphs for comparing the amount of consumption of hardware resources (CPU, RAM) for different SPADE agent behaviors (FSM, One Shot, Timeout, Cyclic, Periodic) during and outside the price update process

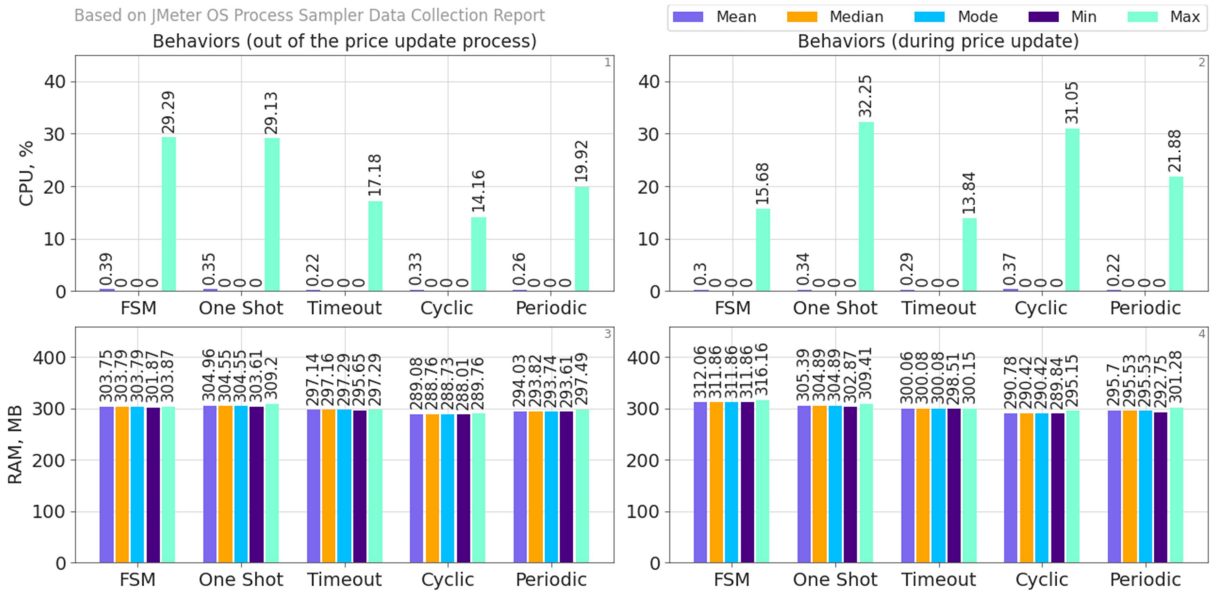


**Figure 14.** Bar graph: Mean, Median, Mode for different SPADE agent behavior (FSM, One Shot, Timeout, Cyclic, Periodic) metrics during and outside the price update process


DOI: 10.24025/2306-4412.3.2023.286553

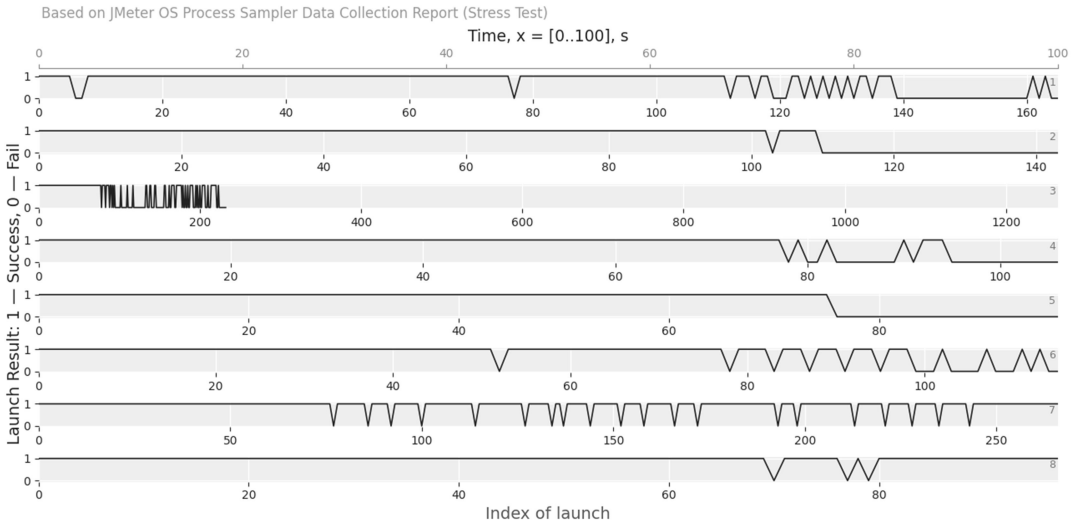


**Figure 15.** Eight tests for launch errors

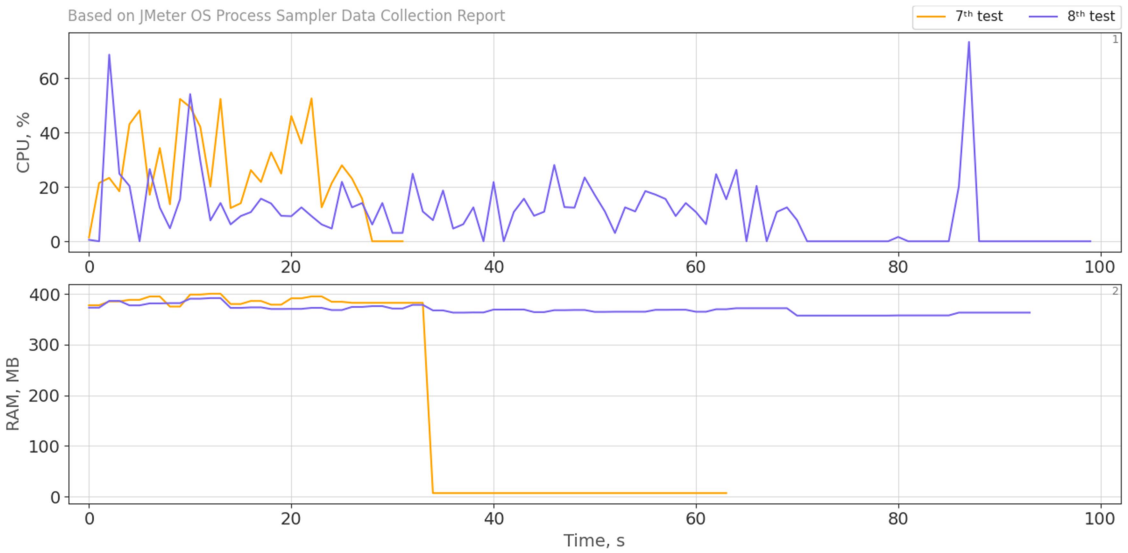


**Figure 16.** CPU and RAM metrics for 7th and 8th tests

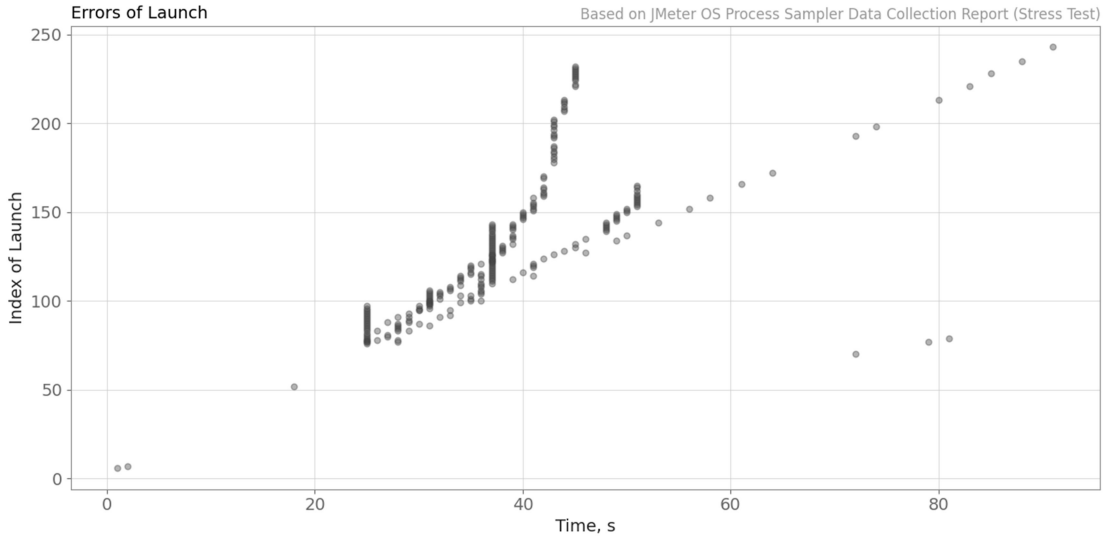


**Figure 17.** Scatter plot of launch errors


DOI: 10.24025/2306-4412.3.2023.286553

## Conclusions

During this study, performance testing of SPADE software agents was carried out (including load and stress testing). As a result of the analysis of the collected data, the performance gain (in terms of hardware resource consumption) was determined for different types of behavior and different interfaces (command line interface (CLI) and agent web user interface (AWUI)). The efficiency in the speed of execution of the price update algorithm for SPADE technologies was also determined (compared to the web).

For the first time introduced the utilization of the Smart Python Agent Development Environment (SPADE) in combination with a cost-based method (CBM) for retail price actualization in the absence of data. To the best of our knowledge, this is the first instance where SPADE agents and CBM are integrated to address pricing challenges like that.

The integration of SPADE and CBM agents into the pricing process is of practical importance for retailers, opening up opportunities for the study and development of new tools with a view to their subsequent application in solving specific problems.

Further development was found by the approach of combining SPADE agents and CBM in dropshipping. This approach can be modified to explore other pricing methods (e.g., customer value-based method, competitive-based method), respond to market dynamics (e.g., adjusting prices depending on market conditions), include additional factors (such as market demand, production costs, and competitor prices) to make intelligent decisions and adapt to changing market conditions in real time.

## Acknowledgements

None.

## Conflict of Interest

None.

## References

Abbas, R., Sultan, Z., & Bhatti, S.N. (2017). Comparative study of load testing tools: Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS), Siege. In *International Conference on Communication Technologies (ComTech)* (pp. 39-44). Rawalpindi, Pakistan. doi: 10.1109/COMTECH.2017.8065747.

about:memory – Firefox Source Docs documentation. Retrieved from https://firefox-source-docs.mozilla.org/performance/memory/about_colon_memory.html.

Advanced Behaviours – SPADE 3.3.0 documentation. Retrieved from https://spade-mas.readthedocs.io/en/latest/behaviours.html.

Ajitha, S., Mithun, G., & Kumar, T.V.S. (2016). Optimal travel management using software agent. In *International Conference on Circuits, Controls, Communications and Computing (I4C)*. Bangalore, India, 1-4. doi: 10.1109/CIMCA.2016.8053289.

Ali, S., & Chernenko, A. (2020). Performance Testing – Microsoft Dynamics 365 Finance and Operations For sales order creation web service by deploying Blazemeter and JMeter, LAP LAMBERT Academic Publishing, July 2020, 56 p. ISBN: 978-620-2-67341-9. Retrieved from https://www.researchgate.net/publication/343334368_Book_Title_Performance_Testing_-_Microsoft_Dynamics_365_Finance_and_Operations_For_sales_order_creation_web_service_by_deploying_Blazemeter_and_JMeter.

Apache JMeter. Retrieved from https://jmeter.apache.org.


DOI: 10.24025/2306-4412.3.2023.286553

Dilshan de Silva et al. (2023). Evaluating the effectiveness of different software testing frameworks on software quality. PREPRINT (V. 1). doi: 10.21203/rs.3.rs-2928368/v1.
Documentation: JMeter-Plugins.org. Retrieved from https://jmeter-plugins.org/wiki/PerfMon/.
Documentation: JMeter-Plugins.org. Retrieved from https://jmeter-plugins.org/wiki/DummySampler/.
Donancio, H., Casals, A., & Brandão, A.A. (2019). Exposing agents as web services: A case study using JADE and SPADE. Retrieved from https://gsigma.ufsc.br/wesaac2019/paper/WESAAC_2019_paper_22.pdf.
Dueñas, J.A. et al. (2021). Magnetic influence on water evaporation rate: An empirical triadic model. *Journal of Magnetism and Magnetic Materials*, 539, 168377. ISSN 0304-8853. doi: 10.1016/j.jmmm.2021.168377.
GitHub - hyperic/sigar: System Information Gatherer and Reporter. Retrieved from https://github.com/hyperic/sigar.
GitHub - javipalanca/spade_bdi: Plugin for SPADE 3 MAS platform to implement BDI Agents. Retrieved from https://github.com/javipalanca/spade_bdi.
GitHub - undera/perfmon-agent: Server metrics fetching agent, based on SIGAR. Retrieved from https://github.com/undera/perfmon-agent.
Graffius, S.M. (2023). Leverage the Power of the Minimum Viable Product (MVP), Retrieved from https://doi.org/10.13140/RG.2.2.24064.20486.
Holgado-Terriza, J.A., Pico-Valencia, P., & Garach-Hinojosa, A. (2020). A gateway for enabling uniform communication among inter-platform JADE agents, *IOS Press, Intelligent Environments*, 28, 82-91. doi: 10.3233/AISE200027.
Jubilson, A.E. et al. (2016). Revolution in e-commerce by the usage of software agents. *International Journal of Advanced Computing and Electronics Technology (IJACET)*, 3(5). Retrieved from https://troindia.in/journal/ijacet/vol3iss5/16-19.pdf.
Lyu, G., Fazlirad, A., & Brennan, R.W. (2020). Multi-agent modeling of cyber-physical systems for IEC 61499 based distributed automation. *Procedia Manufacturing*, 51, 1200-1206, ISSN 2351-9789. doi: 10.1016/j.promfg.2020.10.168.
Mahadevan, R. et al. (2022). Payday loans – blessing or growth suppressor? Machine learning analysis. *General Economics (econ.GN), Machine Learning (cs.LG)*. doi: 10.48550/arXiv.2205.15320.
Matam, S., & Jain, J. (2017). JMeter plugins. In *Pro Apache JMeter* (pp. 211-219). Apress, Berkeley, CA. doi: 10.1007/978-1-4842-2961-3_9.
Mokhamd, H., Arief, G., & Yoan, I. (2023). Analysis of application performance testing using load testing and stress testing methods in API service. *Journal of Sisfotek Global*, 13(1), 28-34. doi: 10.38101/sisfotek.v13i1.2656.
Nordeen, A. (2020). *Learn Software Testing in 24 Hours: Definitive Guide to Learn Software Testing for Beginners,* N.p., Guru99, 291 p.
Pal, C.-V. et al. (2020). A review of platforms for the development of agent systems. *Multiagent Systems (cs.MA)*, 40 p. doi: 10.48550/arXiv.2007.08961.
Palanca, J. (2017). SPADE: Agents based on XMPP. Retrieved from https://www.slideshare.net/JavierPalanca/spade-agents-based-on-xmpp-82102493.
Palanca, J. (2018). SPADE Documentation – spade.mas.pdf, v3.0.0, Aug. 17. Retrieved from https://buildmedia.readthedocs.org/media/pdf/spade-mas/feature-3.0/spade-mas.pdf.
Palanca, J. (2023). Spade Documentation, v. 3.3.0, Jun. 13. Retrieved from https://spade-mas.readthedocs.io/_/downloads/en/latest/pdf/.


DOI: 10.24025/2306-4412.3.2023.286553

Palanca, J. et al. (2022). A flexible agent architecture in SPADE. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection: 20th International Conference* (pp. 320-331), 13616. doi: 10.1007/978-3-031-18192-4_26.

Palanca, J. et al. (2023). Flexible agent architecture: Mixing reactive and deliberative behaviors in SPADE. *Electronics*, 12(3), 659. doi: 10.3390/electronics12030659.

Palanca, J., Terrasa, A., Julian, V., & Carrascosa, C. (2020). SPADE 3: Supporting the new generation of multi-agent systems. *IEEE Access*, 8, 182537-182549. doi: 10.1109/ACCESS.2020.3027357.

Pérez, S.F. (2023). *Spade-BDI Documentation*. Release 0.3.0, Jun 13. Retrieved from https://spade-bdi.readthedocs.io/_/downloads/en/latest/pdf/.

Pohilko, A., JMeter Plugins - More Powerful Load Testing with JMeter Plugins. Retrieved from https://www.methodsandtools.com/tools/jmeterplugins.php.

Python. (n.d.).Retrieved from https://www.python.org.

Slhoub, Kh.A.M. (2018). *Standardizing the Requirements Specification of Multi-Agent Systems*. Florida Institute of Technology, 143. Retrieved from https://repository.lib.fit.edu/bitstream/handle/11141/2608/SLHOUB-DISSERTATION-2018.pdf.

SPADE – SPADE 3.3.0 documentation. (n.d.). Retrieved from https://spade-mas.readthedocs.io/en/develop/readme.html.

SPADE. (n.d.). Retrieved from https://pypi.org/project/spade/.

Spade-BDI – Spade-BDI 0.3.0 documentation. (n.d.). Retrieved from https://spade-bdi.readthedocs.io/en/latest/readme.html.

The SPADE Agent Model – SPADE 3.3.0 Documentation. Retrieved from https://spade-mas.readthedocs.io/en/latest/model.html.

Umbreen, J., Mirza, M.Z., Ahmad, Y., & Naseem, A. (2022). Assessing the role of minimum viable products in digital startups. In *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 1073-1077). Kuala Lumpur, Malaysia. doi: 10.1109/IEEM55944.2022.9989653.

XMPP Load Testing - The Ultimate Guide. Retrieved from https://www.blazemeter.com/blog/xmpp-testing.

Zelenko, E. (2022). Determining the correlation between datasets for calculation of the retail price when using software agents. *Management of Development of Complex Systems*, 50, 102-105. doi: 10.32347/2412-9933.2022.50.102-105.

Zelenko, E., & Kataieva, Y. (2023). Overview of methods and software for pricing. In *Sworld-Us Conference Proceedings* (pp. 23-27), 1(usc17-01). doi: 10.30888/2709-2267.2023-17-01-023.

Zelenko, E., & Kataieva, Ye.Yu. (2023). Classification and synthesis of the main dropshipping disadvantages to eliminate them using software agents. *Electronic Modeling*, 45(2), 115-122. doi: 10.15407/emodel.45.02.115.


DOI: 10.24025/2306-4412.3.2023.286553

# ПРОГРАМНІ АГЕНТИ SPADE ТА ЇХ ВПЛИВ НА АПАРАТНІ РЕСУРСИ

**Е. В. Зеленько**
Аспірант
Черкаський державний технологічний університет
б-р Шевченка, 460, м. Черкаси, 18006, Україна
https://orcid.org/0000-0002-9939-3830
**Є. Ю. Катаєва**
Канд. техн. наук, доцент
Черкаський державний технологічний університет
б-р Шевченка, 460, м. Черкаси, 18006, Україна
Slovak University of Technology in Bratislava
Vazovova 5, 812 43 Bratislava 1, Slovak Republic
https://orcid.org/0000-0002-9668-4739

**Анотація.** Усуваючи недоліки оновлення цін при великій кількості товарів в інтернет-магазині, нами було знайдено рішення в застосуванні Smart Python Agent Development Environment (SPADE). У статті представлено процес збору даних про показники продуктивності SPADE та Openfire Server з метою визначення та аналізу споживання системних ресурсів при підключенні програмних агентів з різними типами поведінки, у різній кількості, а також під час взаємодії з веб-додатком.

У цій роботі як інструмент для збору даних та тестування продуктивності (зокрема навантажувального та стресового тестування) використано JMeter. Використано кількісні та якісні методи аналізу даних. При обробці зібраних значень показників використання апаратних ресурсів, виявлення зв'язків і закономірностей між показниками використано методи математичної статистики. Для порівняння поведінки програмного агента SPADE, визначення ефективності одного над іншим, а також з метою визначення ефективності використання інтерфейсу агента в режимі командного рядка порівняно з веб-аналогом у вигляді графічного інтерфейсу користувача (в аспекті продуктивності) використано формули для розрахунку темпу приросту.

Під час дослідження визначено: перевагу SPADE у швидкості виконання програмного коду; різницю в продуктивності між поведінкою агента, а також між режимом веб-інтерфейсу користувача агента (AWUI) і режимом інтерфейсу командного рядка (CLI); особливості використання режиму CLI агента для інтерактивної взаємодії користувача з додатком з метою швидкого виправлення помилок, що виникають під час взаємодії агента з веб-додатком.

Інтеграція агентів SPADE у процес ціноутворення має практичне значення для компаній у роздрібній торгівлі, відкриваючи можливості для вивчення та розробки нових інструментів для подальшого застосування у вирішенні специфічних проблем.

**Ключові слова:** XMPP, Python, behavior, web, JMeter, CPU, RAM.

*Дата надходження: 29.08.2023*
*Прийнято: 12.09.2023*


DOI: 10.24025/2306-4412.3.2023.286553