УДК 004.41
DOI https://doi.org/10.26661/2786-6254-2025-1-15

ВИКОРИСТАННЯ ТЕОРІЇ ЧЕРГ ТА ІМОВІРНІСНОГО АНАЛІЗУ В АВТОМАТИЗОВАНОМУ ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

Чузов Д. В.
аспірант
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0009-0006-3879-6536
chuzov.d@gmail.com

Чопорова О. В.
доктор філософії, доцент
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0000-0003-3167-7869
o.choporova@znu.edu.ua

Ключові слова:
автоматизоване тестування, вебдодатки, теорія черг, інформаційні технології, імовірнісний аналіз.

У статті розглядається актуальна проблема автоматизованого тестування вебзастосунків, що стає все більш важливим з огляду на стрімке зростання складності сучасних вебсистем і вимог до їх надійності, якості та продуктивності. Пропонований підхід базується на застосуванні методів імовірнісного аналізу та теорії черг, що дає змогу моделювати процеси взаємодії користувачів із вебзастосунками і прогнозувати їх поведінку в умовах різноманітних сценаріїв використання. Особлива увага приділяється розробці аналітичних моделей, які враховують випадковий характер виникнення запитів до сервера, а також можливі затримки у їх обробці, що виникають через обмежені ресурси системи.

У досліджені побудовано модель на основі теорії черг, яка дає можливість описати процеси обробки запитів у вебзастосунках. З використанням методів імовірнісного аналізу можна оцінювати ймовірності різних сценаріїв, що виникають під час тестування, а саме моделювати збої системи за перевищення граничного навантаження або виникнення черг на обслуговування запитів. Це дає змогу краще розуміти потенційні проблеми продуктивності та надійності вебсистем, а також розробляти ефективні стратегії їх тестування.

Запропонована методологія автоматизованого тестування уможливлює оптимізацію процесу виявлення вразливостей і збоїв у роботі вебзастосунків. Описані концептуальні моделі можуть бути інтегровані в існуючі системи автоматизованого тестування, забезпечуючи їхню більшу точність і надійність. Крім того, застосування теорії черг у контексті тестування дає змогу вирішувати завдання оптимізації розподілу ресурсів під час роботи з великими обсягами даних, що також підвищує ефективність роботи вебзастосунків у режимах високого навантаження. Отже, результати дослідження можуть бути корисними для розробників і тестувальників вебсистем, а також для науковців, що займаються проблемами моделювання процесів у сфері інформаційних технологій.

USING QUEUE THEORY AND PROBABILITY ANALYSIS IN AUTOMATED WEB APPLICATION TESTING

Chuzov D. V.
Postgraduate Student
Zaporizhzhia National University
Universytetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0009-0006-3879-6536
chuzov.d@gmail.com

Choporova O. V.
Doctor of Philosophy, Associate Professor
Zaporizhzhia National University
Universytetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0003-3167-7869
o.choporova@znu.edu.ua

Key words: *automated testing, web applications, queueing theory, information technology, probabilistic analysis.*

The article addresses the pressing issue of automated testing of web applications, which is becoming increasingly important due to the rapid growth in the complexity of modern web systems and the demands for their reliability, quality, and performance. The proposed approach is based on the application of probabilistic analysis methods and queueing theory, allowing the modeling of user interaction processes with web applications and predicting their behavior under various usage scenarios. Special attention is given to the development of analytical models that take into account the random nature of server requests and the possible delays in their processing due to the system's limited resources.

Within the framework of the study, a model based on queueing theory was constructed, enabling an description of the request processing processes in web applications. The use of probabilistic analysis methods allows for the assessment of the likelihood of various scenarios occurring during testing, in particular, simulating system failures when the load exceeds the limit or when queues form for request processing. This helps to better understand potential performance and reliability issues of web systems and to develop effective testing strategies.

The proposed methodology for automated testing allows for the optimization of the process of identifying vulnerabilities and failures in the operation of web applications. The described conceptual models can be integrated into existing automated testing systems, ensuring greater accuracy and reliability. Moreover, the use of queueing theory in the context of testing helps solve resource allocation optimization tasks when working with large amounts of data, which also improves the efficiency of web applications under high-load conditions. Ultimately, the results of the research can be useful to web system developers and testers, as well as to researchers involved in mathematical modeling problems in the field of information technology.

Вступ. Автоматизоване тестування вебзастосунків зазвичай є окремим етапом у забезпеченні якості програмного забезпечення, особливо в умовах зростаючої складності вебархітектур і збільшення числа одночасних користувачів. Складність полягає в тому, що традиційні методи тестування не завжди можуть ефективно передбачити можливі збої чи аномалії в поведінці системи під час пікових навантажень. Часто ігноруються критичні аспекти продуктивності, як-от час відповіді системи, кількість оброблених запитів та стійкість до відмов.

На сьогодні є актуальною потреба в розробці аналітичних підходів, які могли б адекватно описувати динаміку роботи вебзастосунків у реальних умовах експлуатації. Теорія черг, імовірнісні моделі та методи оцінки ризиків можуть допомогти спрогнозувати поведінку системи та підвищити ефективність автоматизованого тестування, особливо в умовах непередбачуваних навантажень.

Метою дослідження є аналіз ефективності застосування методів моделювання у процесі автоматизованого тестування вебзастосунків в умовах реальних навантажень. Особлива увага приділяється оцінці продуктивності систем за допомогою теорії черг та імовірнісних методів на етапі автоматизованого тестування, що дає змогу підвищити точність прогнозування поведінки вебдодатків, зменшити ризики перевантаження та покращити якість тестування.

Огляд літератури. Автоматизоване тестування вебзастосунків є актуальною технологією у забезпеченні якості програмного забезпечення, що підтверджується значним зростанням кількості досліджень у цій сфері. Однією з фундаментальних праць у цій галузі є робота Февстера, де розглянуто ефективне використання інструментів автоматизації тестування [1], що заклала основу для подальших досліджень у напрямі автоматизованого аналізу вебдодатків. Робота Тріведі [2] стала основоположною для використання імовірнісного аналізу в оцінці надійності комп'ютерних систем, що дає можливість ефективно моделювати випадкові процеси, як-от запити користувачів до серверів вебзастосунків. Зі свого боку, Клейнрок [3] одним із перших застосував теорію черг для аналізу комп'ютерних систем, заклавши базу для розробки підходів, що описують обробку запитів у системах з обмеженими ресурсами. Його ідеї були продовжені в роботах, наприклад, Бушері та Ван Дейка [4], які запропонували методи для ефективнішого прогнозування завантаження систем, оптимізацію їхньої продуктивності та визначення пропускної здатності, необхідної для забезпечення стабільної роботи мережі.

Дослідження останніх років свідчать про підвищену увагу до поєднання ймовірнісних методів і теорії. Зокрема, професор Вашингтонського університету в Сент-Луїсі Радж Джайн у своїй роботі наводить детальний аналіз методів продуктивності комп'ютерних систем, зокрема вебсерверів, і описує важливість моделювання прогнозування поведінки систем за різних типів навантажень [5]. Детальне дослідження продуктивності вебсерверів на основі моделей черг провели дослідники університету Коннектикуту й довели, що ці моделі дають змогу значно підвищити точність передбачення часу обробки запитів і допомагають виявляти критичні точки перевантаження [6].

Однак, попри значні досягнення в цій галузі, залишаються відкритими питання, пов'язані з інтеграцією аналітичних моделей і автоматизованих інструментів тестування у єдину систему. Більшість сучасних підходів зосереджені на окремих аспектах тестування, як-от моделювання навантажень або оцінка продуктивності, але комплексні рішення, що поєднують імовірнісний аналіз, теорію черг та автоматизацію, залишаються недостатньо розробленими. Саме на ці аспекти спрямовані сучасні дослідження, зокрема, у контексті моделювання взаємодії вебзастосунків із користувачами за допомогою імовірнісних методів.

Таким чином, аналіз останніх публікацій демонструє важливість подальшого розвитку концептуальних моделей для автоматизованого тестування вебсистем, що базуються на імовірнісному аналізі та теорії черг. Це дасть можливість вирішувати нові виклики, що виникають через збільшення складності вебдодатків і підвищені вимоги до їх продуктивності та надійності.

Виклад матеріалу дослідження. Для побудови описових моделей автоматизованого тестування вебзастосунків поєднано кілька базових підходів: теорія черг та ймовірнісне моделювання. Основна увага приділялася моделям обслуговування запитів і поведінці системи під час обробки великих обсягів даних.

Моделювання вебтрафіку за допомогою теорії черг є одним з основних підходів для аналізу продуктивності вебдодатків під час обробки запитів користувачів. У цій статті розглядається використання моделі M/M/1, яка є класичною моделлю масового обслуговування [7]. Позначення M/M/1 походить із теорії черг і розшифровується таким чином: перша «M» (Memoryless) означає, що запити надходять у систему згідно з пуассонівським процесом (марківський вхідний потік), друга «M» вказує на експоненційний розподіл часу обслуговування, а «1» означає, що в системі є лише один канал обслуговування (сервер).

Теорія черг використовується для опису систем, у яких заявки (у нашому випадку – запити від користувачів) надходять до системи (серверу), що їх обслуговує. У моделі M/M/1 припускається, що:

а) запити надходять до системи згідно з пуассонівським розподілом із середньою інтенсивністю λ (кількість запитів за одиницю часу) [8];

б) час обробки запиту на сервері є випадковою величиною, яка підпорядковується експоненціальному розподілу із середньою інтенсивністю μ (кількість запитів, які сервер може обробити за одиницю часу);

в) у системі є один канал обслуговування (сервер).

Модель M/M/1 дає змогу оцінити середній час перебування запиту в системі, довжину черги, імовірність того, що користувачі стикнуться з затримками, а також час очікування на обслуговування.

Запропонований підхід дає можливість не тільки оцінити основні параметри продуктивності, але й визначити критичні точки перевантаження вебсистеми. Це є важливим кроком у вдосконаленні методів автоматизованого тестування, оскільки тестувальники зможуть адаптивно змінювати параметри тестових сценаріїв відповідно до прогнозованих навантажень.

Ключовими метриками моделі M/M/1 є:

1. **Імовірність затримки або відмови в обслуговуванні**. Це найголовніший показник, що визначає імовірність того, що черга зросте занадто великою і запити будуть затримуватися. Модель M/M/1 дає змогу оцінити цю ймовірність, особливо в умовах наближення до пікового навантаження, коли інтенсивність надходження запитів λ стає близькою до інтенсивності обслуговування μ.

Згідно з моделлю M/M/1, продуктивність вебдодатка може погіршуватися в разі збільшення кількості користувачів або під час пікових навантажень. Основний показник, що характеризує навантаження на систему, – це коефіцієнт використання сервера (ρ), який обчислюється як:

$$\rho = \frac{\lambda}{\mu}. \tag{1}$$

Цей коефіцієнт вказує, яку частку часу сервер зайнятий обробкою запитів. Якщо ρ наближається до 1, система стає перевантаженою, що веде до зростання часу очікування запитів у черзі та підвищеного ризику збоїв.

2. **Середня довжина черги**: це очікувана кількість запитів, що перебувають у черзі на обслуговування. Вона обчислюється як очікуване значення кількості заявок у черзі:

$$L_q = \sum_{n=1}^{\infty}(n-1)P_n, \tag{2}$$

де P_n – імовірність того, що в системі є n запитів (разом із тими, що в черзі та на обслуговуванні). Відомо, що для стаціонарного стану система задовольняє балансові рівняння:

$$P_n = P_{n+1}\lambda. \tag{3}$$

Це означає, що ймовірність переходу від стану n до стану $n+1$ (надходження нової заявки) дорівнює імовірності переходу від стану $n+1$ до стану n (обслуговування заявки сервером). Для нульового стану (коли в системі немає заявок):

$$P_1\lambda = P_0\mu. \tag{4}$$

Звідси знаходимо:

$$P_1 = \frac{\lambda}{\mu}P_0, \tag{5}$$

$$P_n = \left(\frac{\lambda}{\mu}\right)^n P_0 = p^n P_0. \tag{6}$$

Щоб отримати P_0, використовуємо умову, що сума всіх імовірностей дорівнює 1:

$$\sum_{n=0}^{\infty} P_n = 1. \tag{7}$$

Це дає суму геометричної прогресії:

$$P_0 \sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n = 1, \tag{8}$$

$$P_0 * \frac{1}{1-p} = 1. \tag{9}$$

Повертаючись до формули (2), підставимо у неї значення p та P_n з виразу (6):

$$L_q = \sum_{n=1}^{\infty}(n-1)p^n P_0. \tag{9}$$

Оскільки , маємо:

$$L_q = (1-p)\sum_{n=1}^{\infty}(n-1)p^n. \tag{10}$$

Тепер потрібно знайти суму:

$$S = \sum_{n=1}^{\infty}(n-1)p^n. \tag{11}$$

Диференціюючи суму геометричної прогресії, отримуємо такі формули:

$$\sum_{n=0}^{\infty} np^n = \frac{p}{(1-p)^2}, \text{ для } |p| < 1. \tag{12}$$

Запишемо оригінальний ряд таким чином:

$$S = \sum_{n=1}^{\infty} np^n - \sum_{n=1}^{\infty} p^n. \tag{13}$$

Підставимо відомі суми:

$$S = \frac{p}{(1-p)^2} - \frac{1}{1-p} = \frac{p-(p-1)}{(1-p)^2} = \frac{2p-1}{(1-p)^2}, \tag{14}$$

$$L_q = (1-p)S = (1-p)\frac{p^2}{(1-p)^2} = \frac{p^2}{1-p}. \tag{15}$$

Підставимо $\rho = \frac{\lambda}{\mu}$:

$$L_q = \frac{\left(\frac{\lambda}{\mu}\right)^2}{1-\frac{\lambda}{\mu}} = \frac{\lambda^2}{\mu(\mu-\lambda)}. \tag{16}$$

Отже, формула середньої довжини черги:

$$L_q = \frac{\lambda^2}{\mu(\mu-\lambda)}. \tag{17}$$

Звідси маємо, що якщо інтенсивність запитів λ зростає, то довжина черги L_q збільшується квадратично. Якщо λ наближається до μ, то $L_q \to \infty$, що означає перевантаження системи. А чим більша швидкість обслуговування μ, тим менша черга.

3. **Середній час перебування в системі**. Цей показник визначає загальний час, який запит проводить у системі, включно з часом очікування в черзі та часом обслуговування сервером. Середній час перебування заявки в системі складається із середнього часу очікування в черзі W_q – скільки часу заявка чекає початку обслуговування, та середнього часу обслуговування W_s, який дорівнює 1/ μ, оскільки час обробки однієї заявки має експоненційний розподіл із параметром μ. Отже, загальний час перебування заявки в системі можна описати такою формулою:

$$W = W_q + W_s. \quad (18)$$

Згідно з формулою Літтла, котра є однією з ключових теорем теорії масового обслуговування, середня кількість заявок у системі L_q пов'язана із середнім часом перебування W_q [9]:

$$L_q = \lambda W_q . \quad (19)$$

Середня кількість заявок у системі L_q вже відома з формули (17). Маючи це, середній час очікування в черзі можна описати таким чином:

$$W_q = \frac{L_q}{\lambda} = \frac{\lambda}{\mu(\mu - \lambda)}. \quad (20)$$

Підставимо W_q у вираз $W = W_q + W_s$:

$$W = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu} = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{\mu-\lambda}{\mu(\mu-\lambda)} = \frac{\mu}{\mu(\mu-\lambda)} = \frac{1}{\mu-\lambda}. \quad (21)$$

Отже, середній час перебування в системі W обчислюється за формулою:

$$W = \frac{1}{\mu - \lambda}, \quad (22)$$

де μ – інтенсивність обслуговування, а λ– інтенсивність надходження запитів.

З використанням описаних метрик модель M/M/1 дає змогу проводити симуляції та прогнозувати стабільність роботи системи за різних рівнів навантаження, що є корисним для автоматизованого тестування вебдодатків. Завдяки цій моделі можна прогнозувати поведінку системи під час пікових навантажень, виявляти проблеми з продуктивністю, оцінювати ризики перевантаження й потенційні збої.

Для отримання практичних результатів модель було інтегровано в процес автоматизованого тестування. Проведено серію експериментів на вебзастосунках, розгорнутих на серверній інфраструктурі (операційна система Linux (версія Ubuntu 24.04), сервер на базі Apache з процесором 3.2 GHz і 16 GB оперативної пам'яті, база даних MySQL) з використанням одного сервера для обслуговування запитів.

Моделювання проводилося на основі M/M/1 моделі, а також за допомогою інструментів для автоматизованого тестування, зокрема JMeter [10] та Selenium [11]. Експерименти було налаштовано на кілька рівнів навантаження для детального аналізу продуктивності системи за різних умов, а саме експерименти передбачали низьке, середнє, високе та пікове навантаження, інтенсивність надходження запитів (λ) варіювалася від 50 до 250 запитів на секунду, а інтенсивність обслуговування (μ) була встановлена на 200 запитів на секунду, що відповідало максимальній обчислювальній потужності сервера. У підсумку було виміряно середній час перебування запиту в системі, коефіцієнт завантаження сервера (ρ) та середню довжину черги для кожного з варіантів навантаження (див. табл. 1).

Результати показали, що за низьких і середніх навантажень ($\rho \leq 0{,}5$) сервер працює ефективно, із середньою довжиною черги близько нуля. Імовірність затримки становить від 25 до 50 %, що означає, що лише частина запитів може потрапити в чергу. Отже, система функціонує стабільно, з мінімальними затримками. Подальші заходи оптимізації можуть бути необов'язковими, оскільки сервер здатний справлятися з навантаженням.

За високих навантажень ($\rho=0{,}75$) сервер працює на 75 % своєї пропускної здатності, що призводить до помітного збільшення черги до 2,25 запиту. Імовірність затримки досягає 75 %, що означає, що більшість запитів будуть змушені чекати. За такого рівня навантаження система ще стабільна, але користувачі вже можуть відчувати затримки. У таких випадках рекомендується запровадити балансування навантаження або оптимізацію обслуговування, щоб зменшити час очікування.

За пікових навантажень ($\rho=0{,}95$) система майже повністю завантажена. Середня довжина черги зростає до 9 запитів, що може значно збільшити час очікування для кожного запиту. Пікове навантаження робить систему вразливою до перевантажень і збоїв. Щоб уникнути проблем із продуктивністю, важливо розглянути додавання додаткових серверів, покращення алгоритмів кешування або впровадження більш ефективного розподілу навантаження.

А за екстремальних навантажень ($\rho>1{,}0$) сервер досягає своєї максимальної пропускної здатності. Середня довжина черги теоретично прямує до нескінченності, імовірність затримки досягає 100 %, а система ризикує не обробляти нові запити своєчасно. Для вирішення проблеми потрібно оптимізувати роботу сервера, наприклад, за допомогою

Таблиця 1

Результати експерименту залежно від рівня навантаження сервера

Варіант навантаження	Інтенсивність надходження (λ), запити/с	Інтенсивність обслуговування (μ), запити/с	Коефіцієнт використання сервера (ρ)	Середня довжина черги	Середній час перебування запиту в системі	Імовірність відмови або затримки
Низьке	50	200	0,25	0,083	0,267 с	25 %
Середнє	100	200	0,5	0,5	0,5 с	50 %
Високе	150	200	0,75	2,25	1 с	75 %
Пікове	190	200	0,95	9,025	5 с	95 %
Екстремальне	250	200	1,25	теоретично нескінченна черга	система перевантажена	100 %

балансування навантаження. Розподіл трафіку між декількома серверами може зменшити навантаження на кожен окремий сервер, що дасть змогу зменшити затримки і знизити ймовірність збою. Крім того, додавання додаткових серверів або збільшення обчислювальної потужності існуючих серверів допоможе підвищити значення інтенсивності обслуговування μ, що знизить коефіцієнт завантаження. Зі свого боку, застосування асинхронного оброблення запитів або оптимізація часу обробки запиту дає можливість збільшити пропускну здатність μ і зменшити середню довжину черги.

Таким чином, **результати** дослідження продемонстрували ефективність запропонованих моделей у прогнозуванні часу відповіді системи, ймовірності збоїв та оптимізації процесу тестування. Моделі теорії черг дають змогу покращити управління вебтрафіком, а ймовірнісні моделі – виявити критичні точки, де система може бути вразливою до збоїв.

Таке моделювання роботи системи може бути використано для створення більш стійких і надійних вебдодатків, що витримують навантаження та забезпечують швидкий час відповіді, що важливо для користувачів і розробників систем. Їх інтеграція у процес автоматизованого тестування вебзастосунків дає можливість створювати ефективні системи для симуляції навантажень та аналізу продуктивності й надійності. Ці моделі також можуть бути інтегровані з популярними інструментами автоматизованого тестування, як-от Selenium та JMeter, для проведення симуляцій, що відображають реальні сценарії використання вебзастосунків.

Моделі черг дають змогу точно моделювати потік запитів, що надходять до вебдодатка. У процесі тестування ці моделі допомагають оцінювати, як система реагує на різні сценарії навантажень, які можуть передбачати як поступове збільшення кількості запитів, так і різкі пікові навантаження. Наприклад, інструмент JMeter може бути налаштований для симуляції великих потоків запитів відповідно до моделі M/M/1. Він дає змогу задавати різні параметри навантаження, як-от кількість користувачів і частота запитів, щоб відтворити поведінку системи у стані високої завантаженості. Зі свого боку, інструмент Selenium, котрий часто використовується для функціонального тестування вебдодатків, можна застосовувати для симуляції поведінки окремих користувачів або навіть групи користувачів одночасно, за різних умов навантаження, щоб виявити проблеми в продуктивності інтерфейсу користувача.

Крім аналізу продуктивності, інтеграція таких моделей у процес тестування дає можливість створювати комплексні сценарії для оцінки стійкості системи. Ці сценарії можуть відображати, як система поводитиметься за різних умов. Наприклад, тестування системи за максимального навантаження із застосуванням моделі M/M/1 або випадки, коли інтенсивність запитів λ наближається до інтенсивності обслуговування μ. Це дає змогу визначити точки, у яких система починає перевантажуватися. До того ж, використовуючи ймовірнісні моделі збоїв, можна протестувати стійкість системи до відмов серверів або інших компонентів. Це допомагає визначити оптимальні стратегії для відновлення роботи системи в разі збою.

Результати, отримані під час тестування з використанням математичних моделей, можуть бути використані для оптимізації як самої системи, так

і процесу тестування. Наприклад, якщо моделювання показує, що за певних умов навантаження система зазнає значних затримок, це може підказати, де потрібно оптимізувати алгоритми або інфраструктуру.

Висновки. У статті було розглянуто застосування математичних моделей у процесі автоматизованого тестування вебдодатків, акцентовано увагу на моделях черг та імовірнісних підходах. Завдяки впровадженню моделей черг можна точно оцінити продуктивність вебдодатків за різних умов навантаження, а ймовірнісні моделі дають змогу моделювати різноманітні сценарії збоїв та пікових навантажень і оцінити надійність вебдодатків. Інтеграція наведених моделей у процес автоматизованого тестування за допомогою інструментів, як-от JMeter та Selenium, відкриває нові можливості для аналізу продуктивності. Цей підхід допомагає створювати реалістичні сценарії навантаження, а також перевіряти стійкість системи до збоїв у реальних умовах експлуатації. Важливо зазначити, що в майбутньому можна розширити застосування математичних моделей на будь-які типи вебдодатків та інші технології, як-от мікросервіси та хмарні рішення. Отже, використання математичних моделей у процесі автоматизованого тестування вебдодатків потребує подальшого дослідження, оскільки це є перспективним напрямом, який дає можливість покращити якість і надійність програмного забезпечення, а також забезпечити кращий досвід для користувачів.

ЛІТЕРАТУРА

1. Fewster M., Graham D. Software Test Automation: Effective Use of Test Execution. Addison-Wesley, 1999. 574 p.
2. Trivedi K.S. Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd Edition. Wiley, 2001. 830 p.
3. Kleinrock, L. Queueing Systems. Volume 1: Theory. Wiley-Interscience, 1975. 417 p.
4. Boucherie R.J., van Dijk N.M. Queueing Networks: A Fundamental Approach. New York: Springer, 2010. 824 p.
5. Jain R. The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling. URL: https://www.researchgate.net/publication/259310412_The_Art_of_Computer_Systems_Performance_Analysis_Techniques_For_Experimental_Design_Measurement_Simulation_and_Modeling_NY_Wiley (дата звернення: 19.10.2024).
6. Lu J. Performance Analysis of a Web Server. Режим доступу: https://www.researchgate.net/publication/220156026_Performance_Analysis_of_a_Web_Server (дата звернення: 16.10.2024).
7. Shortle J.F., Thompson J.M., Gross D., Harris C.M. Fundamentals of Queueing Theory. 2018. 548 p. DOI: 10.1002/9781119453765.
8. Карташов М. В. Імовірність, процеси, статистика. Київ : ВПЦ Київ. ун-т, 2007. 504 с.
9. Leon-Garcia A. Probability, Statistics, and Random Processes For Electrical Engineering. Pearson, 2007. 832 p.
10. Selenium [Електронний ресурс]. Режим доступу: https://www.selenium.dev (дата звернення: 02.11.2024).
11. Apache JMeter [Електронний ресурс]. Режим доступу: https://jmeter.apache.org (дата звернення: 02.11.2024).

REFERENCES

1. Fewster, M., Graham, D. (1995). Software Test Automation: Effective Use of Test Execution. Addison-Wesley. 574 p.
2. Trivedi, K.S. (2001). Probability and Statistics with Reliability, Queuing, and Computer Science Applications. Wiley. 830 p.
3. Kleinrock, L. (1975). Queueing Systems. Volume 1: Theory. Wiley-Interscience. 417 p.
4. Boucherie, R.J., van Dijk, N.M. (2010). Queueing Networks: A Fundamental Approach. New York: Springer. 824 p.
5. Jain, R. (1991). The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling. URL: https://www.researchgate.net/publication/259310412_The_Art_of_Computer_Systems_Performance_Analysis_Techniques_For_Experimental_Design_Measurement_Simulation_and_Modeling_NY_Wiley (access date: 19.10.2024).
6. Lu, J. (2008). Performance Analysis of a Web Server. URL: https://www.researchgate.net/publication/220156026_Performance_Analysis_of_a_Web_Server.
7. John F. Shortle, James M. Thompson, Donald Gross, Carl M. Harris (2018). Fundamentals of Queueing Theory. 548 p. DOI: 10.1002/9781119453765.
8. Kartashov, M.V. (2007). Imovirnist, protsesy, statystyka. Kyiv: VPTs Kyiv University. 504 p. [in Ukrainian].
9. Leon-Garcia, A. (2007). Probability, Statistics, and Random Processes For Electrical Engineering. Pearson. 832 p.
10. Selenium. URL: https://www.selenium.dev (access date: 02.11.2024).
11. Apache JMeter. URL: https://jmeter.apache.org (access date: 02.11.2024).