Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: **Методи і засоби комплексного тестування комп'ютерних інформаційних систем**

Виконав: студент(ка) 6 курсу, групи СІм-61
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Слюз І. Б. (прізвище та ініціали)

Керівник (підпис) Жаровський Р.О. (прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) Готович В.А. (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« » грудня 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Слюзу Івану Богдановичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методи і засоби комплексного тестування комп'ютерних інформаційних систем*

Керівник роботи *Жаровський Руслан Олегович, к.т.н.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*06*» *грудня 2022* року № *4/7-986*

2. Термін подання студентом завершеної роботи *22.12.2022 р.*

3. Вихідні дані до роботи *Характеристики КІС, життєвий цикл проектування КІС, програмне забезпечення для тестування ІС, методи тестування компонентів КІС.*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Теоретичні основи тестування комп'ютерних інформаційних систем
2. Аналіз методів проведення комплексного тестування комп'ютерної інформаційної системи
3. Апробація методів тестування комп'ютерної інформаційної системи
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета дослідження.
2. Задачі дослідження, об'єкт і предмет, наукова новизна і практична цінність дослідження.
3. Життєвий цикл тестування КІС
4. Сценарій комплексного тестування КІС
5. Процес навантажувального тестування з використанням моделей
6. Апробація методів тестування комп'ютерної інформаційної системи
7. Результати експериментального дослідження.
8. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека в надзвичайних ситуаціях			
Охорона праці			

7. Дата видачі завдання *15.12.2022*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Аналіз сучасних технічних проблем тестування комп'ютерних інформаційних систем*	*14.11.2022-20.11.2022*	*виконано*
2	*Аналіз методів проведення комплексного тестування комп'ютерної інформаційної системи*	*20.11.2022 – 27.11.2022*	*виконано*
3	*Розробка сценарію комплексного тестування комп'ютерної інформаційної системи*	*27.11.2022 – 04.12.2022*	*виконано*
4	*Апробація методів тестування комп'ютерної інформаційної системи*	*04.12.2022– 08.12.2022*	*виконано*
5	*Охорона праці та безпека в надзвичайних ситуаціях*	*08.12.2022-12.12.2022*	*виконано*
6	*Оформлення пояснювальної записки і графічного матеріалу*	*12.12.2022-14.12.2022*	*виконано*
7	*Попередній захист кваліфікаційної роботи магістра*	*14.12.2022*	*виконано*
8	*Захист кваліфікаційної роботи магістра*	*22.12.2022*	

Студент _____ *Слюз Іван Богданович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Жаровський Р.О.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи і засоби комплексного тестування комп’ютерних інформаційних систем // Кваліфікаційна робота // Слюз Іван Богданович // ТНТУ, комп’ютерна інженерія, група СІм-61 // Тернопіль, 2022 // с. – 92, рис. – 47, табл. – 8, аркушів А1 – 8, додат. – 6 , бібліогр. – 21.

Ключові слова: тестування, комп’ютерна інформаційна система, алгоритм, автоматизація.

У кваліфікаційній роботі магістра досліджено методи і засоби тестування комп’ютерних інформаційних систем. Для розглянутих методів визначено основні концепції та етапи проведення тестування.

В рамках навантажувального тестування розглянуто підхід, заснований на моделях. Показано його ефективне застосування для тестування прикладного програмного забезпечення в ІТ- середовищі функціонування, що включає СУБД, бази даних, системне ПЗ, розгорнуті на обладнанні комплексу технічних засобів ІС.

Докладно досліджено технологію навантажувального тестування з використанням моделей, сформовано вимоги до проведення навантажувального експерименту та властивості об'єкта тестування, що охоплюють усі аспекти планування, проведення навантажувального експерименту та аналізу його результатів.

Розглянуто низку інструментальних засобів, що надають можливості для реалізації автоматизованого тестування. Наведено результати аналізу щодо основних із запропонованих інструментів. Показано, що автоматизація процесу тестування допомагає компаніям скорочувати час, що витрачається на тестування

ABSTRACT

Methods and means of computer information systems comprehensive testing // Master thesis // Slyuz Ivan Bogdanovich // TNTU, computer engineering, group CIm-61 // Ternopil, 2022 // p. – 92, fig. – 47, tab. – 8, sheets A1 – 8, add. – 6, bibliography. - 21.

Keywords: testing, computer information system, algorithm, automation.

The master's thesis researched the methods and means of testing computer information systems. The main concepts and stages of testing are defined for the considered methods.

An approach based on models was considered as part of the load testing. Its effective application for testing application software in the IT operating environment, which includes DBMS, databases, system software, deployed on the equipment of a complex of IS technical means, is shown.

The technology of load testing using models was studied in detail, the requirements for carrying out the loading experiment and the properties of the test object were formed, covering all aspects of planning, carrying out the loading experiment and analyzing its results.

A number of tools that provide opportunities for the implementation of automated testing are considered. The results of the analysis regarding the main of the proposed tools are given. Automating the testing process has been shown to help companies reduce the time spent on testing.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

HTTP англ. HyperText Transfer Protocol протокол передачі даних

ІС інформаційна система

QA англ. Quality Assurance забезпечення якості

SLA англ. Service Level Agreement угода про рівень послуг

STLC англ. Software Testing Life Cycle життєвий цикл тестування програмного забезпечення

ІТ інформаційні технології

КІС комп’ютерна інформаційна система

ОС операційна система

ПЗ програмне забезпечення

СУБД система управління базами даних

ВСТУП

Актуальність теми. Інтенсивний розвиток і все більше впровадження інформаційних технологій (ІТ) вимагає від проектувальників та розробників інформаційних систем (ІС) створення ефективних та якісних програмних, апаратних продуктів. Це говорить про те, що перевірка якості є важливим процесом у створенні ІС, при цьому питання та проблеми тестування висвітлені у доступній літературі менше, ніж будь-який інший аспект розробки програмно - апаратних систем.

Аналіз публікацій показав, що сенс терміна «тестування» та визначення його передається різними авторами як: «тестування є процесом, що демонструє відсутність помилок» [15], «мета тестування - показати, що ІС коректно виконує передбачені функції» [16], є недостатньо коректними з точки зору економічної складової (тобто ціни розробки КІС). Найчастіше результати своєї роботи розробник неспроможний оцінити з погляду ефективності ІС, де вона використовується, тобто. він практично нічого не може сказати про те, наскільки повно протестований код програми і яким чином використовуються технічні засоби ІС. Крім того, дуже часто інженерам необхідно вирішувати питання, пов'язані зі скороченням витрат на проектування КІС при цьому забезпечивши стабільність і якість її роботи. Основним способом вирішення цих проблем є тестування ІС. Процес тестування включає вирішення питань не тільки технічного характеру (організація ефективного процесу тестування, визначення часу тестування, використання або невикористання інструментальних засобів і т.д.), але і питань економічного характеру.

Аналіз досліджень та публікацій у галузі тестування показав, що розгляд таких аспектів тестування, як технічні, економічні представляє значний практичний інтерес. Відповідно до вищесказаного, метою є дослідження системи тестування для перевірки КІС з погляду економічного та технічного підходів. Великий внесок у вирішення різних аспектів проблеми тестування зробили: Позін Б.А., Савін Р., Шмейлін Б.З., Бородін А.М., Мирвода С.Г., Andrews A., Offut J., Changyou Xing, Heiskanen H., Maunumaa M., Honlin Zh., Wenbo X., Makinen M. та інші.

Мета кваліфікаційної роботи. Метою роботи є аналіз методів тестування та розробка сценарію комплексного тестування для підвищення продуктивності комп’ютерних інформаційних систем.

Для того, щоб досягти мети, необхідно вирішити наступні задачі:

– розглянути теоретичні засади тестування інформаційних систем, що дозволяють здійснювати аналіз наявних методів тестування інформаційних систем;

– проаналізувати з урахуванням розробленої теорії практичні методи тестування програмного забезпечення інформаційних систем;

– розробити сценарій комплексного тестування, що дозволяє з одного боку оцінити продуктивність інформаційної системи;

– підтвердити шляхом експерименту ефективність реалізованого сценарію та результати його впровадження, використовуючи процес апробації.

Об'єкт дослідження: методи комплексного тестування комп’ютерних інформаційних систем.

Предмет дослідження: розробка сценарію комплексного тестування для підвищення продуктивності та надійності комп’ютерних інформаційних систем.

Методи дослідження: методи системного аналізу та дослідження операцій; методи комплексного тестування комп’ютерних інформаційних систем для підвищення їхньої продуктивності.

Наукова новизна дослідження полягає в тому, що в ньому визначено основні критерії ефективності тестування, які застосовуються для оцінки продуктивності та надійності КІС; розроблений сценарій комплексного тестування КІС, що дозволяє на основі визначення типів помилок, що найчастіше зустрічаються, задавати якісні тести, що забезпечують мінімізацію втрат у бізнес-середовищі організації.

Практичне значення результатів кваліфікаційної роботи полягає у розробці теоретичних положень концепції аналізу та порівняння методів тестування інформаційних систем, що дозволяють здійснювати вибір найбільш відповідного методу для практичного використання, покладеного в основу розробки сценарію комплексного тестування, що дозволить суттєво скоротити витрати на тестування КІС та підвищити якість її функціонування.

Публікації. Результати дослідження апробовано на X науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології», XI міжнародній науково-технічній конференція молодих учених та студентів «Актуальні задачі сучасних технологій» (7-8 грудня 2022 року), у вигляді тез конференцій.

1. Слюз І., Жаровський Р. Принципи та основні етапи комплексного тестування комп'ютерної інформаційної системи. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. С. 93.

2. Слюз І., Жаровський Р. Критерії ефективності тестування комп'ютерної інформаційної системи. Матеріали XI Міжнародна науково-технічна конференція молодих учених та студентів «Актуальні задачі сучасних технологій» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. С. 174.

Структура роботи. До складу кваліфікаційної роботи магістра входить розрахунково-пояснювальна записка та графічний матеріал. Розрахунково-пояснювальна записка містить вступ, 4 розділи, загальні висновки, список використаної літератури і додатки. Обсяг роботи: розрахунково-пояснювальної записки – 92 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТЕСТУВАННЯ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Комп'ютерні інформаційні системи (КІС) з погляду системного аналізу відносяться до складних систем, оскільки вони побудовані з безлічі різних підсистем: серверів БД, серверів додатків, серверів кешування, серверів резервного копіювання, безпеки і т.д. [15]. У добре спроектованій системі великий ступінь взаємної інтеграції, що збільшує ризик виходу з ладу однієї або кількох підсистем КІС, за яким слідує з ладу інших раніше працездатних підсистем і так далі аж до повного руйнування всієї КІС. Без спеціальних засобів обробки нештатних ситуацій КІС виявиться нестійкою до помилок і не зможе гарантувати закладені у неї технічні вимоги, а також вимоги щодо часу відновлення до працездатного стану у разі виникнення передбачуваних та непередбачених помилок [7].

Тестування КІС охоплює основні стадії життєвого циклу, аналогічні до послідовності процесів розробки програмного забезпечення: постановка задачі для тесту, проектування, написання тестів, тестування тестів, виконання тестів та вивчення результатів тестування [12]. У зв'язку з цим розробка методики (сценаріїв) тестування КІС, що проводиться на всіх етапах її життєвого циклу, є актуальною [2].

1.1. Життєвий цикл тестування комп'ютерної інформаційної системи

Так як тестування КІС ґрунтується на основних етапах життєвого циклу тестування програмного забезпечення (STLC), то спершу розглянемо і формалізуємо основні етапи тестування програмних продуктів. Крім того комплексне тестування КІС включає в себе також тестування ПЗ.

У процесі STLC для покращення якості продукту виконуються різні дії. У життєвому циклі тестування програмного забезпечення (STLC) передбачені такі етапи зі своїми критеріями входу та результатами:

1. Аналіз вимог (табл. 1.1) - на цьому етапі команда забезпечення якості (QA) розуміє вимоги з точки зору того, що ми тестуватимемо, і з'ясуємо вимоги, що тестуються. Вимоги можуть бути функціональними або нефункціональними, такими, як продуктивність, тестування безпеки [6].

Таблиця 1.1

Вхідні критерії та результат етапу «Аналіз вимог»

Вхідні критерії	Заходи	Результат
Мають бути доступні наступні документи: - Технічні вимоги. - Архітектура системи. Поряд з вищезазначеними документами мають бути чітко визначені критерії прийому.	Підготуйте список запитань чи запитів та отримайте відповіді від Business Analyst, System Architecture, Client, Technical Manager / Lead тощо. Складіть список того, що виконували всі типи тестів, такі як функціональність, безпека, продуктивність і т.д. Визначте напрямок та пріоритети тестування. Перерахуйте деталі середовища тестування, де проводитимуться дії з тестування. При необхідності перевірте техніко-економічне обґрунтування автоматизації та підготуйте техніко-економічне обґрунтування автоматизації.	Список питань з усіма відповідями, які мають бути вирішені, тобто тестові вимоги

2. Планування випробувань (табл. 1.2) - на цьому етапі визначаються зусилля та оцінки витрат для всього проекту.

Таблиця 1.2

Вхідні критерії та результат етапу «Планування випробувань»

Вхідні критерії	Заходи	Результат
Документи з вимогами (оновлена версія нечіткої чи відсутньої вимоги). Автоматизація техніко- -економічного обґрунтування.	Визначити мету та обсяг проекту. Перерахуйте типи тестування, що застосовуються у STLC. Оцінка зусиль тестування та планування ресурсів. Вибір інструменту тестування, якщо потрібно. Визначте процес тестування. Визначте середовище тестування, необхідне для всього проекту. Підготуйте розклад випробувань. Визначте контрольні процедури. Визначте ролі та обов'язки. Перерахуйте результати тестування. Визначте вхідні критерії, критерії призупинення, критерії поновлення та критерії виходу. Визначте ризики, якщо такі є.	План тестування чи документ про стратегію тестування. Документ щодо оцінки зусиль із тестування.

3. Розробка тестового прикладу (таб. 1.3) - цьому етапі STLC команда тестування записує докладні тестові приклади. Поряд із тестовими наборами команда тестування також готує тестові дані, якщо такі необхідні для тестування.

Таблиця 1.3

Вхідні критерії та результат етапу «Розробка тестового прикладу»

Вхідні критерії	Заходи	Результат
Документи з вимогами (оновлена версія нечіткої чи відсутньої вимоги). Автоматизація техніко-економічного обґрунтування.	Підготовка тестових випадків. Підготовка сценаріїв автоматизації тестування (за потреби). Підготовка необхідних тестових даних до виконання тестових випадків.	Тестові випадки. Тестові дані. Тестування сценаріїв автоматизації (якщо потрібно).

4. Налаштування тестового середовища (табл. 1.4), яке визначає умови тестування ІС.

Таблиця 1.4

Вхідні критерії та результат етапу «Налаштування тестового середовища»

Вхідні критерії	Заходи	Результат
Документи з вимогами (оновлена версія нечіткої чи відсутньої вимоги). Автоматизація техніко-економічного обґрунтування.	Підготовка тестових випадків. Підготовка сценаріїв автоматизації тестування (за потреби). Підготовка необхідних тестових даних до виконання тестових випадків.	Тестові випадки. Тестові дані. Тестування сценаріїв автоматизації (якщо потрібно).

5. Виконання тесту (табл. 1.5) – на цьому етапі команда тестування починає виконання тестових прикладів на основі підготовленого тест плану та підготовлених тестових прикладів на попередньому етапі. Якщо якийсь із тестових випадків заблокований через якийсь дефект, то такі тестові випадки можуть бути позначені як заблоковані, тому ми можемо отримати звіт на основі того, скільки тестових прикладів пройшло, провалилося, заблоковано або не виконано тощо. Після

усунення дефектів, ті ж випробування з помилками або блокуванням можуть бути виконані знову для повторного тестування функціональності [6].

Таблиця 1.5

Вхідні критерії та результат етапу «Виконання тесту»

Вхідні критерії	Заходи	Результат
План тестування або документ про стратегію тестування. Тестові випадки. Тестові дані.	На основі тест плану виконайте контрольні приклади. Позначте стан тестових випадків, таких як Пройдено, Збій, Заблоковано, Не виконано тощо. Призначте ідентифікатор помилки для всіх невдалих та заблокованих тестових випадків. Зробіть повторне тестування, коли дефекти усунуті. Слідкуйте за дефектами до закриття.	Звіт про виконання тесту. Звіт про дефекти.

6. Закриття циклу випробувань (табл. 1.6) - цьому етапі аналізуються помилки, розподіляються дефекти на кшталт і складності.

Таблиця 1.6

Вхідні критерії та результат етапу «Закриття циклу випробувань»

Вхідні критерії	Заходи	Результат
Виконання тестового прикладу завершено Звіт про виконання тесту Звіт про дефекти	Оцініть критерії завершення циклу тестування на основі охоплення, якості, вартості, часу, критичних бізнес-цілей та програмне забезпечення. Підготуйте показники тесту на основі вищезазначених параметрів. Підготувати звіт про закриття тесту Поділитись передовим досвідом для будь-яких подібних проектів у майбутньому	Звіт про закриття тесту Тест метрики

Сценарії тестування STLC [15] повинні сприяти пошуку непередбачених помилок/проблем у роботі КІС на всіх етапах життєвого циклу, що виражається замкненою послідовністю дій (рис. 1.1):

- стадія 1 - загальне планування та аналіз вимог - для визначення методів та видів тестування, обмежень та ризиків які належить тестувати; наявність необхідного інструментарію тощо;
- стадія 2 - уточнення критеріїв приймання - для визначення/уточнення

метрик та ознак можливості/необхідності початку, зупинення та відновлення тестування, завершення або припинення тестування;

- стадія 3 – уточнення стратегії тестування – для розгляду та уточнення актуальних для поточної ітерації частин стратегії тестування;
- стадія 4 – розробка тест-кейсів – для розробки, перегляду, уточнення, доопрацювання, переробки тощо з тест-кейсами/тестовими сценаріями;
- стадія 5 - виконання тест-кейсів - для безпосереднього виконання сценарію тестування;
- стадія 6 - фіксація знайдених дефектів - для формування розуміння проблеми та уточнення важливості та терміновості проведення тестування;
- стадія 7 - аналіз результатів тестування - для обробки отриманих результатів, щоб приймати рішення щодо ще однієї ітерації;
- стадія 8 - звітність - для фіксування проміжних та кінцевих варіантів тестування.

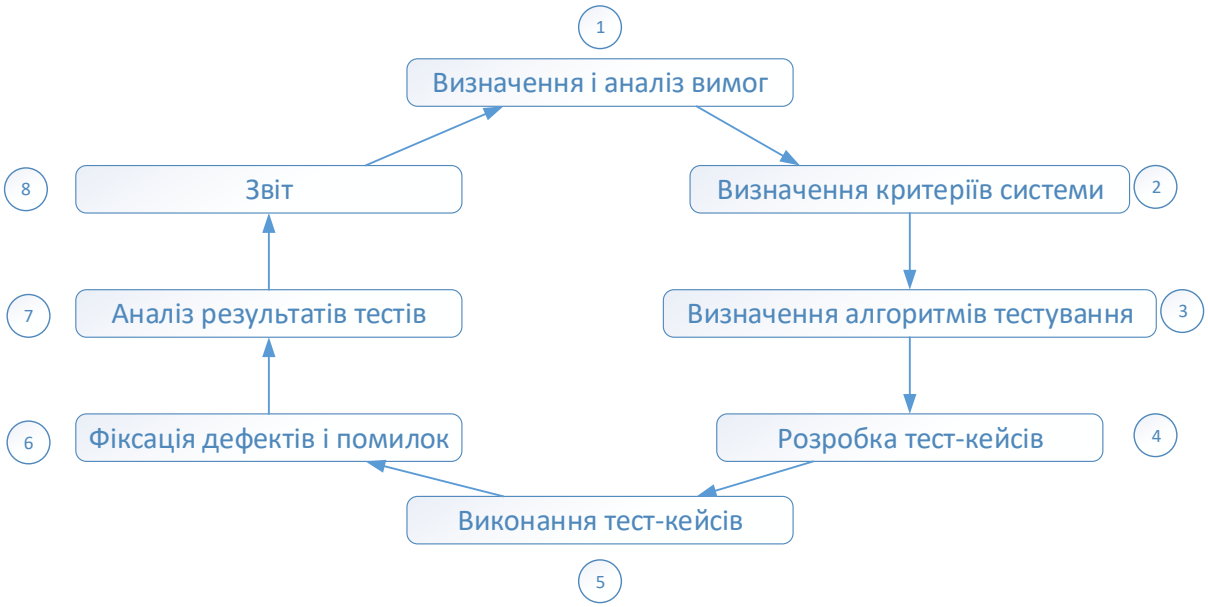


Рис. 1.1. Життєвий цикл тестування КІС

Кожна з виділених стадій тестування може розглядатися як така, що забезпечує створення певного проміжного продукту - функціональної групи чи частини ІС з деякими обмеженими характеристиками якості. Ці характеристики

виділяються та деталізуються на основі первинного технічного завдання та специфікації вимог на ІС. У процесі проектування ІС вони уточняються та конкретизуються у специфікаціях вимог на групи компонент ІС [17]. В результаті створюється сукупність еталонів, що мають набори значень показників якості, яким повинні відповідати компоненти, що налагоджуються і випробовуються, на кожній стадії тестування ІС.

Таким чином, пройшовши основні стадії життєвого циклу тестування ІС, можна з упевненістю стверджувати про працездатність програмного продукту.

1.2. Принципи та основні етапи комплексного тестування комп'ютерної інформаційної системи

Щоб орієнтуватися у тестах, варто розглянути два основні підходи.

Перший підхід орієнтований вивчення логіки програмного забезпечення КІС під час проектування тестів. У процесі проектування тестів необхідно передбачити, щоб кожна команда умовного переходу виконувалася хоча б раз, тобто. Необхідно перевірити кожну частину ІС [14]. При цьому зовсім (чи майже) не цікавляться специфікаціями.

Другий підхід тестування ґрунтується на виконанні фундаментального принципу функціонування КІС: віддача має перевищувати витрати. Ця віддача вимірюється ймовірністю того, що тест виявить невиявлену раніше помилку. Витрати вимірюються часом та вартістю підготовки, виконання та перевірки результатів тесту [11]. Кожен тест повинен бути представлений деяким класом вхідних значень таким чином, щоб його правильне виконання створювало переконаність у тому, що ІС працюватиме правильно для певного класу вхідних даних.

Якщо врахувати, що КІС - це не тільки програмні компоненти, що використовуються в її складі, але й апаратне та організаційне забезпечення, то і в результатах її випробувань повинні бути відображені показники обраних серверів, робочих станцій, мережевого обладнання (їх надійність та продуктивність), а також

ефективність розробленого регламенту експлуатації системи У зв'язку з цим виникає необхідність у проведенні комплексного тестування на відповідність усім вимогам, що висуваються.

Комплексне тестування КІС складається з наступних етапів [10]:

1. Детальне вивчення проекту системи та її експлуатаційних документів.
2. Створення та впровадження автоматизованої системи відстеження помилок (bug tracking).
3. Безпосереднє тестування роботи ІС, яке включає:
   a. тестування роботи всіх модулів ІС,
   b. перевірка внутрішньо системних зв'язків,
   c. тестування обладнання, у тому числі на наявність необхідних ліцензій,
   d. перевірка програмного забезпечення, у тому числі перевірка коду,
   e. тестування продуктивності та максимальних навантажень ІС,
   f. тестування на відмови: несподівані програмні збої, вихід з ладу модулів системи, людський фактор та ін.,
   g. тестування на захищеність ІС - включає комплекс досліджень з виявлення способів злому системи і витоків інформації,
   h. загальна перевірка роботи системи.
4. Тестування роботи системи управління ІТ -структурою.
5. Тестування персоналу.
6. Аналіз отриманих даних та вироблення стратегії з виправлення виявлених помилок.

Комплексне тестування не призначене для тестування всіх функцій повністю зібраної системи . Воно спрямовано на пошук невідповідності системи її вихідним цілям [14]. Таким чином, у комплексному тестуванні бере участь КІС, опис її можливостей та вся документація, яка поставлятиметься разом із системою.

1.3. Методологія тестування комп’ютерних інформаційних систем

Методологія тестування є стратегією та підходом для тестування ІС, щоб гарантувати, що ІС придатна для експлуатації.

Методики тестування включають тестування того, що ІС працює відповідно до специфікації, не має небажаних побічних ефектів при використанні методів, що виходять за межі проектних параметрів і в гіршому випадку буде стійкою до відмови [14].

Методології тестування ІС – це різні підходи та способи забезпечення того, щоб ІС була повністю протестована. Методології тестування охоплюють усі: від модульного тестування окремих модулів, інтеграційного тестування всієї ІС, до спеціалізованих форм тестування, таких як безпека та продуктивність [7].

Оскільки ІС стають все більш складними та взаємопов'язаними, а також з великою кількістю різних платформ і пристроїв, які необхідно протестувати, важливо мати надійну методологію тестування, щоб переконатися, що ІС, що розробляється, були повністю протестовані, що вони відповідають зазначеним вимогам і можуть успішно працювати у всіх очікуваних середовищах з необхідною зручністю використання та безпекою.

На рис. 1.2 представлено загальну схему основних видів тестування ІС, передбачених методологіями тестування.

Рис. 1.2. Загальна схема основних видів тестування

Відповідно до даної схеми методологія тестування включає [13]:

1. Функціональне тестування виконується з використанням функціональних специфікацій, наданих клієнтом, або з використанням специфікацій проектування, таких як сценарії використання, які надає команда розробників.

Функціональне тестування в методології тестування розбите на чотири компоненти: модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування.

Модульне тестування - тестування окремих програмних модулів чи компонентів, що входять до складу програми чи ІС. Модульні тести зазвичай пишуться розробниками модуля, і в методології розробки, заснованої на тестуванні (такі як Agile, Scrum або XP), вони фактично пишуться до того, як модуль буде створений як частина специфікації [16]. Кожна функція модуля тестується спеціальним модульним тестовим приладом, написаним тією ж мовою програмування, що сам модуль (рис. 1.3).

Рис. 1.3. Фрагмент модульного тесту

Інтеграційне тестування - тестування різних модулів/компонентів, які були успішно протестовані при інтегруванні разом для виконання конкретних завдань та заходів (також відомі як тестування сценарію). Це тестування зазвичай виконується за допомогою комбінації автоматичних функціональних тестів та ручного тестування.

Системне тестування включає тестування системи на наявність помилок. Даний вид тесту проводиться шляхом взаємодії з апаратними та програмними

компонентами всієї системи, а потім здійснюється перевірка загалом (рис. 1.4). Для цього типу тестування застосовується метод «чорної скриньки», де програмне і апаратне забезпечення перевіряється на відповідність до очікуваних робочих умов, і навіть можливих граничних умов.

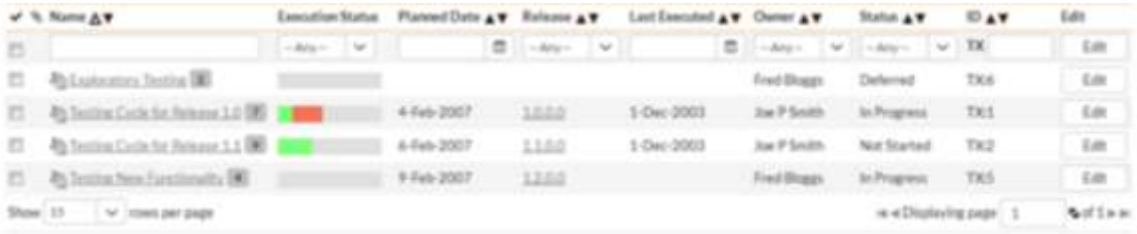


Рис. 1.4. Демонстрація роботи системного тестування

Приймальне тестування (рис. 1.5) є заключним етапом функціонального тестування ІС і включає перевірку того, що всі вимоги до продукту виконані, і що кінцеві користувачі та клієнти протестували систему, щоб переконатися, що вона працює належним чином і відповідає всім пред'явленим вимогам [5].

Рис. 1.5. Демонстрація роботи приймального тестування системи

2. Нефункціональне тестування включає тестування ІС на відповідність нефункціональним вимогам, які зазвичай включають вимірювання / тестування ІС на відповідність певним технічним якостям, наприклад: вразливість, масштабованість, зручність використання [7, 11].

У більшості методологій тестування існує кілька різних типів тестування продуктивності (рис. 1.6), наприклад [16]:

- тестування продуктивності - це вимір поведінки системи при зростаючому навантаженні (як числа користувачів, і обсягів даних),
- навантажувальне тестування – перевірка того, що система може працювати у потрібному режимі,
- стрес-тестування – це визначення точки відмови в системі, коли протестоване навантаження перевищує те, яку вона може підтримувати.

Рис. 1.6. Демонстрація роботи тестування продуктивності

Тестування сумісності (рис. 1.7) перевіряє сумісність ІС з вказаними операційними системами, апаратними платформами, мобільними пристроями. Тести сумісності потрібні [18], щоб переконатися, що ІС працює, як очікується, у всіх різних комбінацій апаратних/програмних, і що всі функції послідовно підтримуються.

Рис. 1.7. Демонстрація роботи тестування сумісності

Тестування безпеки перевіряє ІС на конфіденційність, цілісність, автентифікацію, доступність та недоторканність. Індивідуальні тести проводяться для запобігання будь-якому несанкціонованому доступу до ІС.

Юзабіліті-тестування розглядає аспект зручності використання програмно-апаратного забезпечення ІС для кінцевого користувача. Простота, з якою користувач може отримати доступ до продукту, формує основну точку тестування. Юзабіліті-тестування розглядає п'ять аспектів тестування:

−навчання,

−ефективність,

−задоволеність,

−запам'ятовування та помилки.

Таким чином, сукупність розглянутих тестових випробувань може бути представлена у вигляді комплексного тестування (integration tests), яке проводиться під час процесу інтеграції технічного та програмного забезпечення до валідації комп'ютерної системи з метою перевірки сумісності програмного забезпечення та технічного забезпечення ІС.

Усі комплексні тести мають бути підготовлені на основі документації для користувача. Вони пишуться у формі сценаріїв, що представляють низку послідовних дій користувачам і складаються з трьох основних компонентів [12]:

1. власне сценарію комплексного тестування із зазначенням дій, які мають бути досконалі під час виконання тесту
2. вхідних даних,
3. очікуваних вихідних даних.

У комплексному тестуванні крім фахівців можуть брати участь і користувачі КІС, ґрунтуючись на одному з методів [12]:

1. Тестова експлуатація, коли система тестується на робочому місці. Це дозволяє побачити КІС у роботі до того, як її почнуть експлуатувати.

2. Використання КІС в організації розробника. Дозволяє усунути безліч помилок, але не дає змоги побачити деякі помилки інтеграції.

Таким чином, комплексне тестування КІС може бути процесом як контролю, так і випробування, при якому можна побачити її роботу в реальному середовищі користувача або в обстановці, яка створена, щоб максимально відповідати середовищу користувача.

1.4. Види тестів під час виконання комплексного тестування комп’ютерних інформаційних систем

Компонентами комплексного тесту є вхідні цілі, документація, інструкції для користувачів та сама система. Усі комплексні тести мають бути підготовлені з урахуванням інструкцій для користувача (а не зовнішніх специфікацій). До зовнішніх специфікацій слід звертатися тільки для того, щоб розібратися в протиріччях між системою та інструкціями до неї. Виділяють такі підвиди комплексного тестування:

1. Стрес тестування або тестування з навантаженням - спроба піддати систему максимальному "тиску" (наприклад, спробу одночасно підключити до системи максимальну кількість терміналів).
2. Навантажувальне тестування - спроба завантажити системі великі обсяги даних протягом тривалого часу, щоб визначити співвідношення об’єму даних які система встигає опрацювати і даних зазначених у специфікації.
3. Тестування конфігурації - спроба перевірити роботу мінімальної та максимальної конфігурації системи з будь-якою апаратною платформою або програмою, з якою КІС буде взаємодіяти.
4. Тестування сумісності - спроба знайти несумісності нових та старих версій КІС, при цьому взаємодія користувачів з попередньою версією має повністю зберегтися.
5. Тестування захисту – спроба порушити захист системи.
6. Тестування вимог до пам'яті - спроба показати, що система не використовує максимальний об’єм пам'яті, який прописано у документації.
7. Тестування продуктивності - спроба продемонструвати, що компоненти, чи

в загальному КІС, відповідають заявленим характеристикам, таким як час відгуку, рівень пропускної здатності при певному навантаженні.

8. Тестування процесів налаштування системи – спроба визначити якість та ергономічність роботи системи.

9. Тестування надійності/готовності - спроба довести, що система не задовольняє вихідним вимогам надійності (середній час між відмовами, кількість помилок, здатність до виявлення, виправлення помилок та стійкість до помилок).

10. Тестування засобів відновлення - спроба перевірити здатність системи до відновлення (особливо у сфері роботи операційної системи, СУБД, систем передачі).

11. Тестування зручності обслуговування - спроба проаналізувати очима обслуговуючого персоналу документи, що описують внутрішню логіку системи, щоб зрозуміти, як швидко і точно визначити причину помилки, якщо відомі лише деякі її ознаки.

12. Тестування документації - спроба повірити точність усієї документації.

13. Тестування зручності установки - спроба перевірити та усунути недоліки встановлених процедур системи.

14. Тестування зручності експлуатації – спроба вирішити проблеми ергономіки щодо взаємодії користувача із системою.

Аналіз видів комплексного тестування показав, що вони не зводяться до перевірки окремих функцій системи, а перевіряють працездатність КІС загалом. Найчастіше вони пишуться у формі сценаріїв, у яких вказуються дії, які здійснюються під час виконання тесту. Усі тестові сценарії мають бути методологічно та систематично опрацьовані. При тестуванні повинні бути використані вихідні дані та послідовності команд, які в документації користувача явно не рекомендуються або оголошуються забороненими. На рис. 1.8 зображено сценарій комплексного тестування.

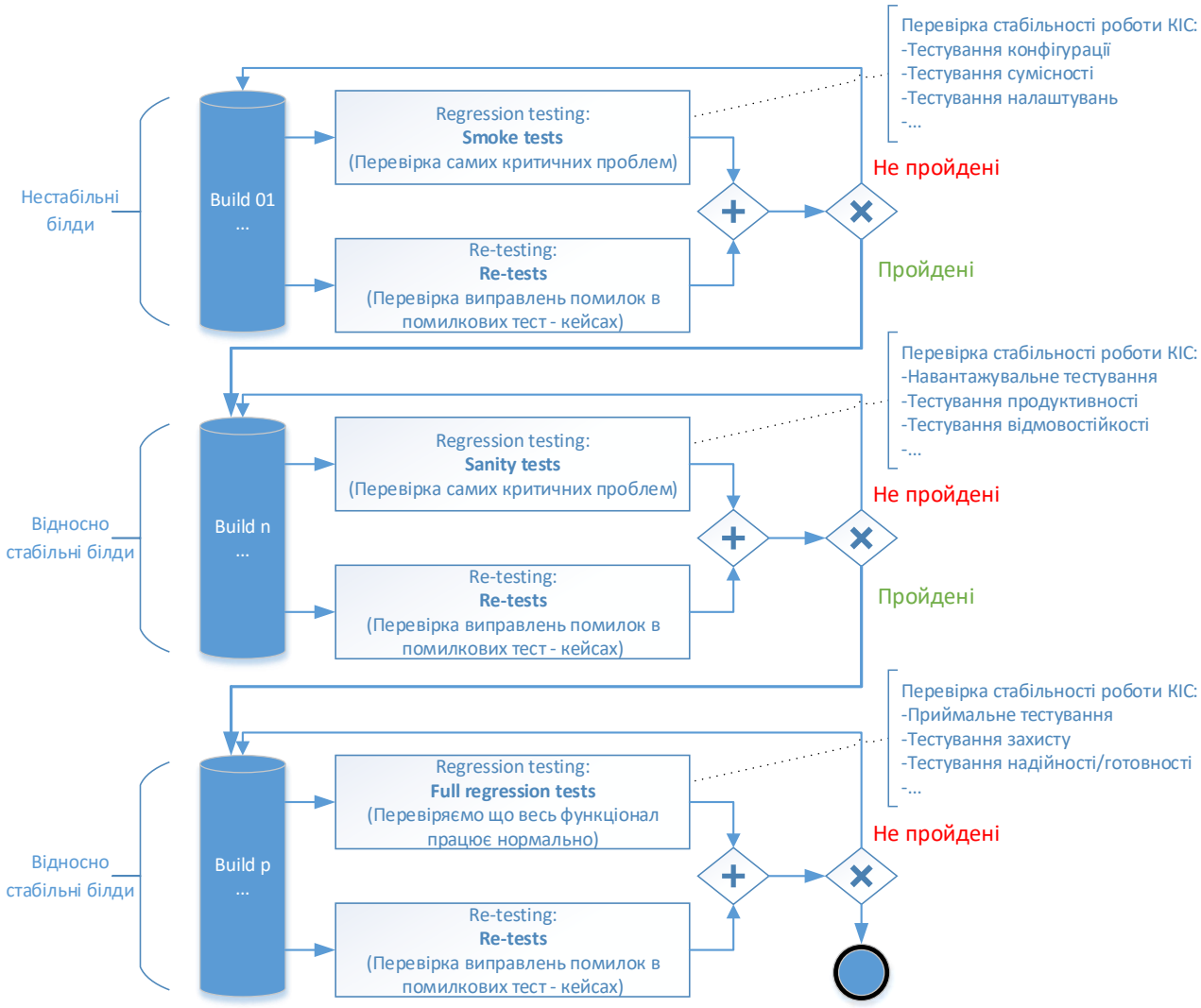


Рис. 1.8. Сценарій комплексного тестування КІС

У ході аналізу теоретичних основ комплексного тестування комп'ютерних систем та аналізу їх основних видів було зроблено висновок про те, що комплексне тестування КІС не зводиться до перевірки лише її функціоналу, а орієнтовано на оцінку якості та продуктивності. Для проведення комплексного тестування має бути розроблений сценарій, який має передбачити всі можливі варіанти, що можуть призвести до нестабільної роботи КІС.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ ПРОВЕДЕННЯ КОМПЛЕКСНОГО ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Основні критерії ефективності тестування комп'ютерної інформаційної системи

Під тестуванням розуміють процес дослідження виконання функцій інформаційної системи на кінцевій множині вхідних даних X, отримання відгуку Y та його порівняння з еталонною множиною вихідних значень $Y_{\text{ет}}$ з метою виявлення помилок і дефектів в ІС.

Пара $(x, y_{\text{ет}})$, $x \in X, y_{\text{ет}} \in Y_{\text{ет}}$ називається тестовим випадком (тест-кейс), а всі тестові випадки, згруповані за певною ознакою, називаються тестовим комплектом. Рішення про наявність помилки у функціонуванні КІС приймається або при розбіжності результатів на одному з тестових випадків, або якщо відрізняються закони розподілу вихідних даних [13].

Було помічено, що у міру виявлення найбільш серйозних помилок і недоліків, ефективність низьковитратних методів падає разом з кількістю помилок, що виявляються. Тому всі методи тестування в межах своїх задач мають більшу ефективність в порівнянні з іншими. Можна виділити вимоги до оптимальних критеріїв тестування: достатній, повний, надійний, легко перевіряється.

Існують такі класи критеріїв тестування [17]:

1. Структурні критерії.
2. Функціональні критерії.
3. Критерії стохастичного тестування:
   - статистичні методи;
   - метод оцінки швидкості виявлення помилок.
4. Мутаційні критерії (на основі підходу Монте-Карло).

Для розробки тестів потрібно обирати ті методи, що в основному реалізують критерії функціональних тестів.

2.2. Методи комплексного тестування комп’ютерної інформаційної системи

Комплексне тестування КІС передбачає застосування методів тестування програмних продуктів, апаратних засобів та аналізу предметної галузі.

1. Виділимо серед методів тестування: методи функціональних діаграм та попарного тестування [18, 13], а для аналізу предметної галузі розробимо онтологічну модель.

Функціональна діаграма є формальною мовою, якою здійснюється трансляція специфікації програми і яка написана природною мовою. Побудова тестів з використанням методу функціональних діаграм здійснюється поетапно.

На першому етапі визначаються вхідні дані і здійснюється їх розбиття на класи еквівалентності.

На другому етапі визначаються причинно наслідкові зв’язки у відповідності до специфікації.

На третьому етапі з визначених зв’язків формується булевий граф, який і представляє собою функціональну діаграму.

На четвертому етапі до діаграми додають примітки, які описують комбінації причин або наслідків і задають різного роду обмеження.

На п’ятому етапі з діаграми формують таблицю рішень стовбці якої відповідають тесту.

Під час розробки тестів доводиться аналізувати роботу інформаційних систем з великою кількістю параметрів. Одним із підходів для оптимального підбору кількості тестів є метод ортогональних масивів.

Розроблені набори комбінаторних стратегій що дозволяють обрати таку підмножину вхідних комбінацій, яка дозволяє максимально збільшити ймовірність виявлення дефектів. Парне тестування є одним з оптимальних серед них. Парне тестування використовує аналіз граничних значень та розподіл еквівалентності, які припускають, що для кожної пари вхідних параметрів системи повинні бути всі можливі дискретні комбінації цих параметрів.

Для аналізу предметної галузі розробки тестів застосовується онтологічний

підхід [5]. Під онтологією розуміється впорядкована трійка виду:

$$O = (C, R, F), \quad (2.1)$$

де C - кінцева множина концептів предметної області; R - кінцева множина відносин між концептами заданої предметної області; F - кінцева множина функцій інтерпретації (аксіоматизації), заданих на концептах та/або відносинах онтологій O.

Тестування слід здійснювати з погляду користувача, що передбачає повне розуміння того, для чого система застосовуватиметься.

На рис. 2.4 представлена онтологічна модель тестового випадку.

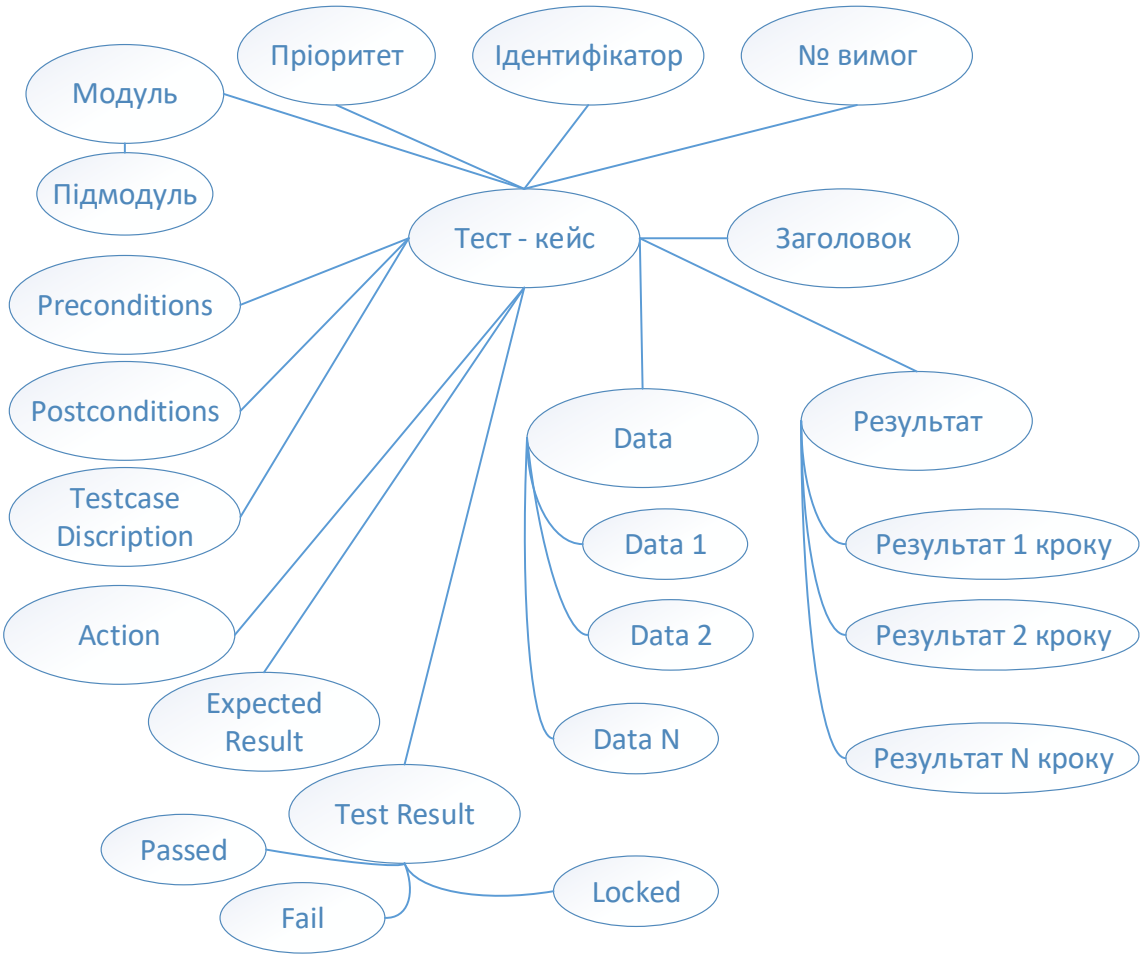


Рис. 2.4. Онтологічна модель тестового випадку

2. У зв'язку з тим, що сучасному класу комп’ютерних інформаційних систем притаманний модульний принцип побудови, виділимо два підходи до комбінування модулів [5]: покрокове та монолітне.

Покроковий метод тестування передбачає, що модулі тестуються не

ізольовано, а підключаються по черзі для виконання тесту доки до набору протестованих модулів не буде підключений останній модуль.

Монолітний метод, або метод «великого удару», застосовується під час тестування та збирання програми.

Аналіз виділених методів модульного тестування КІС показав:

– Монолітне тестування трудомістке, однак менша витрата машинного часу.

– При монолітному тестуванні можлива паралельна організація роботи на початковій фазі тестування.

– При покроковому тестуванні вже на початкових етапах виявляються помилки між модулями, а також легше здійснювати налагодження програм.

– Результати покрокового тестування більш повні і дають кращий результат при визначенні дефектів КІС.

Все це дозволяє зробити висновок, що покрокове тестування є кращим при розробці та апробації КІС .

3. Інтеграційне тестування спрямоване на перевірку взаємодії між частинами (модулями) програмного забезпечення КІС.

4. Щоб процес тестування КІС мав виправдану з економічної погляду трудомісткість, необхідно заздалегідь виробити ряд стратегій. Найбільш поширені:

- тестування методом «чорної» скриньки;
- тестування методом «білого» скриньки.

5. Для комплексного тестування КІС також застосовуються такі методи [4]:

Альфа-тестування – це ручне тестування потенційними користувачами, замовниками чи незалежною командою тестування на стенді розробки. Альфа-тестування часто використовується як форма внутрішнього приймального тестування перед проведенням бета-тестування.

Альфа-тестування дозволяє фільтрувати, уточнювати і передавати розробникам дефекти, що надходять, з докладним описом, що значно скорочує час, а також дозволяє скорочувати трудовитрати розробників на пошук причини дефекту і його виправлення.

У межах проведення альфа-тестування вирішуються такі:

- підготовка розкладу тестування;
- організація учасників тестування;
- відбір і уточнення зауважень, що надходять;
- реєстрація дефектів.

Бета-тестування проводиться після альфа-тестування та може використовуватись як приймальне тестування зовнішніми користувачами. Бета-версія системи передається групі користувачів поза командою розробки, щоб знизити кількість дефектів. Іноді версія передається кільком командам, щоб отримати зворотний зв'язок від якнайбільшої кількості майбутніх користувачів.

Ключові переваги:

- отримання відгуків та побажань від потенційних користувачів КІС;
- підвищення якості проведеного тестування у задані терміни та ліквідація проблем, пов'язаних із тестовим середовищем.

Таким чином, використання разом розглянутих методів сприяє більш якісному проведенню комплектного тестування КІС.

2.3. Комплексне тестування комп’ютерної інформаційної системи на її продуктивність

Сучасні комп’ютерні інформаційні системи часто змушені працювати з великим навантаженням. Тестування продуктивності дозволяє оцінювати та планувати продуктивність КІС, а також керувати нею в умовах планового, підвищеного та пікового навантаження. Навантажувальне тестування виконується з метою оцінки поведінки системи за нормальних умов і на прогнозованому піку навантаження. Стресове тестування передбачає ретельну перевірку в екстремальних умовах з метою оцінки стабільності КІС.

2.3.1.Тестування продуктивності КІС

Тестування продуктивності (Performance Testing) - це тестування, яке виконується визначення того, як компоненти системи працюють у конкретній

ситуації [2]. Основна мета тестування продуктивності включає встановлення еталонної поведінки системи. Тестування продуктивності не спрямовано пошук дефектів у додатку. Успішний тест продуктивності має спровокувати більшість проблем продуктивності, які можуть бути пов'язані з базою даних, мережею, програмним забезпеченням, обладнанням тощо.

Нижче наведено загальний процес тестування продуктивності [4]:

1. Визначте своє середовище тестування – ознайомтеся з деталями апаратного, програмного забезпечення та мережевих конфігурацій, використаних під час тестування.

2. Визначте критерії прийнятності продуктивності - сформулюйте цілі та обмеження пропускної спроможності, часу відгуку та розподілу ресурсів, критерії успіху проекту поза цією метою та обмеженнями.

3. Плануйте та проектуйте тести продуктивності - визначте, як використання може змінюватись серед кінцевих користувачів, та визначте ключові сценарії для тестування у всіх можливих випадках використання.

4. Налаштування середовища тестування – підготуйте середовище тестування перед виконанням, організуйте інструменти та інші ресурси.

5. Реалізувати тестовий дизайн – створіть тести продуктивності відповідно до тестового дизайну.

6. Запустити тести – виконайте та оцініть результати тесту.

7. Аналізуйте, налаштовуйте та повторно тестуйте - об'єднуйте, аналізуйте та ділитесь результатами випробувань; виконайте точне налаштування та перевірте знову, щоб побачити, чи є покращення або зниження продуктивності.

Створювані сценарії тестування продуктивності мають бути близькими до реальних умов. Технологія тестування продуктивності має підтримувати сценарії, які не лише збільшують кількість користувачів, але і імітують їх поведінку. Типовою поведінкою може бути, наприклад, відвідування користувачем домашньої сторінки, вхід до системи, перехід за посиланням на статтю, додавання товару в кошик та здійснення покупки. У наступному фрагменті коду показаний опис простого моделювання для Gatling на Scala (рис. 2.5).

```
import io.getling.core.Predef/_
import io.getling.core.structure.ScenarioBuilder
import io.getling.http.Predef/_
import io.getling.http.protocol.HttpProtocolBuildder
import scala.concurrent.duration._

class CarCreationSimulation extends Simulation {
    val httpConf: HttpProtocolBuildder = http
    .baseURL(http://test.car-manufacture.example.com/car-
manufacture/resources)
        .acceptHeader("*/*")

    val scn: ScenarioBuilder = scenario("create_car")
        .exec(http('request_1")
            .get("/cars"))
        .exec(http('request_1")
            .post("/cars"))
            .body(StringBody("""{"id": "X123A234", "color": "RED",
                        "engine": "DIESEL"}""")).asJSON
            .check(header("Location").saveAs("locationHeader")))
        .exec(http('request_1")
            .get("${locationHeader}"))

    Pause(1 second)

Setup(
    scn.inject(rampUsersperSec(10).to(20).during(10 second))
        ).protocols(httpConf)
        .constantPauses
)
```

Рис. 2.5. Фрагмент опису обробки клієнтських запитів

Сценарій create_car включає три клієнтські запити, які читають список всіх автомобілів, створюють автомобіль і підключаються до створеного ресурсу. Сценарії настроюють кілька віртуальних користувачів. Кількість користувачів починається з 10 і збільшується до 20 протягом 10 секунд. Тестування продуктивності дозволяє визначити максимальну інтенсивність операцій, коли система задовольняє вимогам щодо час відгуку.

На ринку є безліч інструментів для тестування продуктивності, серед них є:

NeoLoad – це платформа для тестування продуктивності, розроблена для DevOps. З NeoLoad команди тестують у 10 разів швидше, ніж з традиційними інструментами, щоб відповідати новому рівню вимог протягом усього життєвого

циклу розробки програмного забезпечення Agile – від компонентів до повних тестів навантаження в масштабі всієї системи;

LoadView Testing – платформа для тестування інфраструктури в будь-якому масштабі. LoadView пропонує навантажувальне тестування на основі хмарних обчислень.

HP LoadRunner – це найпопулярніший на сьогоднішній день інструмент для тестування продуктивності. Цей інструмент здатний моделювати сотні тисяч користувачів, піддаючи додатки реальному навантаженню, щоб визначити їх поведінку при визначеному навантаженні.

Jmeter - один з провідних інструментів, що використовуються для навантажувального тестування веб-серверів і серверів додатків.

Тестування продуктивності рекомендується проводити для нової версії програмного забезпечення, що планується до впровадження. Воно дозволяє виявити та запобігти відмовим КІС перед впровадженням в експлуатацію; визначити максимальну кількість одночасно працюючих користувачів у системі та максимальна кількість операцій, що одночасно виконуються без втрати якості обслуговування.

2.3.2.Навантажувальне тестування КІС

Навантажувальне тестування (load testing) - даний тип тестування дозволяє оцінити поведінку системи при зростаючому навантаженні. Метою тесту навантаження є визначення максимального навантаження, яке може витримати система [15].

У ролі навантаження може бути кількість користувачів, а також кількість операцій на сервері.

Продуктивність при цьому визначається такими факторами:

−швидкістю роботи програмного забезпечення;

−швидкістю роботи апаратного забезпечення;

−швидкістю роботи мережі.

Під час тестування можуть здійснюватися такі операції, що дозволяють більш

точно виміряти продуктивність та визначити «вузьке місце» системи:

−вимірювання часу і інтенсивності виконання окремих функцій інформаційної системи;

−визначення максимальної кількості користувачів, які можуть одночасно використовувати ресурси КІС;

−визначення меж допустимої продуктивності при збільшенні навантаження від виконання операцій чи одночасного підключення користувачів.

Приклад такого навантажувального тесту показаний на рис. 2.6.

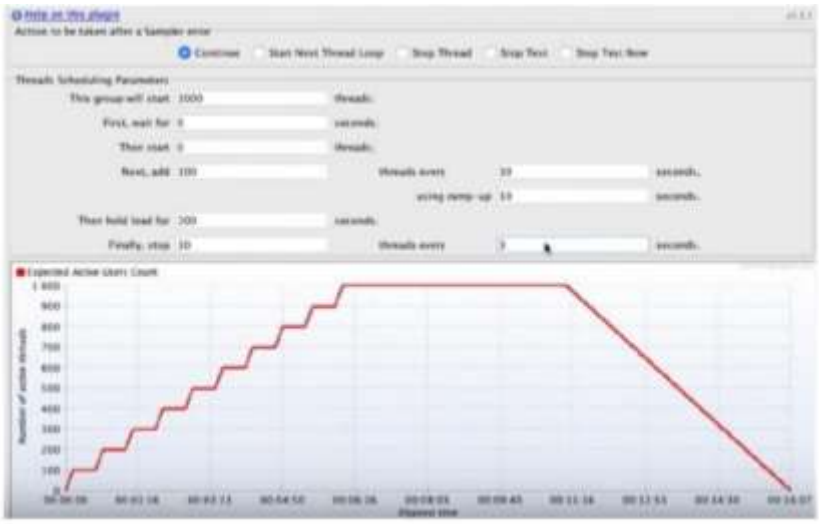


Рис. 2.6. Приклад виконання навантажувального тесту

Цей тест проводиться для визначення кількості користувачів, із якими може в нормальному режимі працювати КІС. Умови тесту наступні:

−через кожні 30 секунд до системи приєднується по 100 користувачів,

−максимальне навантаження 1000 користувачів,

−крок підключення 30 секунд, після чого Jmeter чекає 30 секунд, перш ніж розпочати наступний крок,

−при навантаженні 1000 потоків, час їх одночасної роботи 300 секунд (5 хвилин),

−на останньому етапі кожні 3 секунди зупиняємо по 10 потоків.

Інший приклад навантажувального тесту показаний на рис. 2.7, де зображено пікове навантаження від 7000 користувачів.

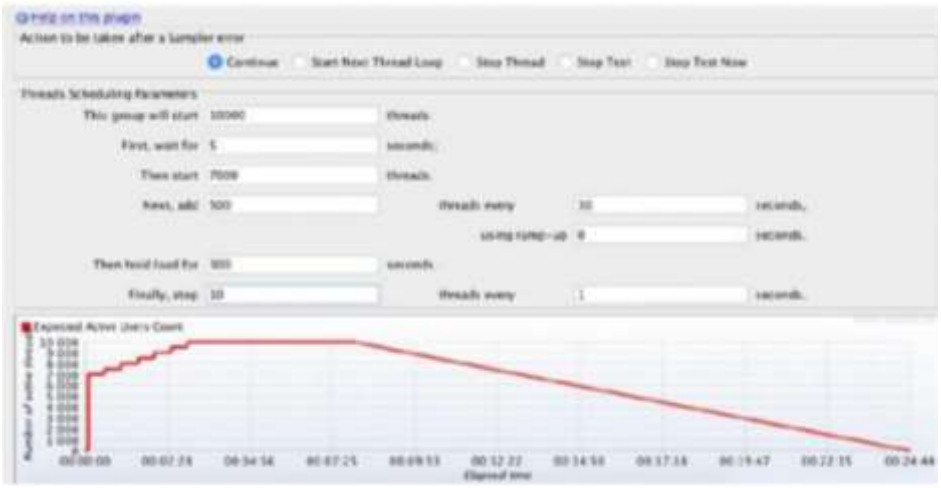


Рис. 2.7. Приклад виконання навантажувального тесту

Навантажувальне тестування по суті дозволяє визначити запас міцності КІС.

2.3.3.Навантажувальне тестування, засноване на моделях

Розглянемо підхід, заснований на моделях, стосовно навантажувального тестування КІС, в рамках якого здійснюється тестування прикладного програмного забезпечення (ПЗ) в ІТ- середовищі функціонування, що включає СУБД, бази даних, системне ПЗ, розгорнуті на обладнанні комплексу технічних засобів ІС [16, 17, 18].

При навантажувальному тестуванні КІС у процесі експлуатації головною метою є оцінка експлуатаційних характеристик інформаційної системи, у складі якої функціонує прикладне ПЗ. Зазвичай це звані «Показники призначення», чи нефункціональні вимоги, тобто пропускна здатність/продуктивність ІС, реактивність під час вирішення функціональних завдань та інших.

При формалізації предметної області (для класу систем) формуються вихідні дані [17]:

1. Інформація про обраний вид навантажувального тестування;
2. Інформація про вимірювані характеристики та показники продуктивності;
3. Інформація про структуру системи з погляду варіантів подачі навантаження та способів вимірювань;
4. Інформація про завантаження в структурованому вигляді.

Представимо такі моделі, за допомогою яких при постановці завдання навантажувального експерименту здійснюється вибір необхідних характеристик, показників та вимірюваних величин, що адекватно характеризують процес функціонування ІС, що тестується (рис. 2.8):

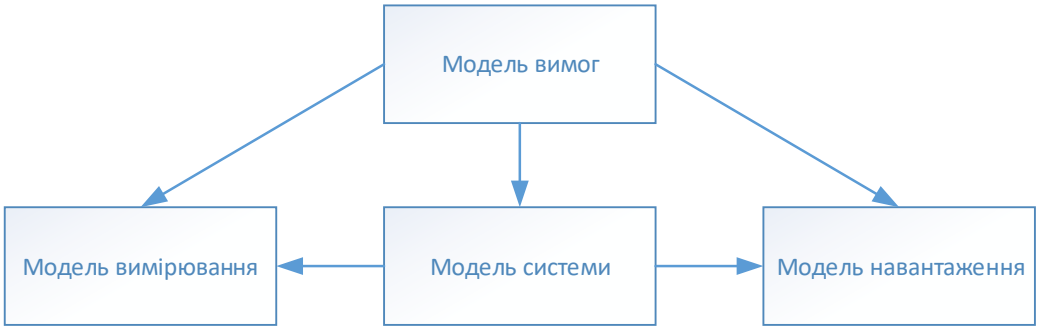


Рис. 2.8. Взаємозв'язок моделей

1. Модель вимог – характеризує тип тестованої системи, склад дисфункційних вимог (бізнес-правила та технічні вимоги) тестованої інформаційної системи [18].

Модель вимог може бути подана у такому вигляді:

$$R = B \cup T \tag{2.2}$$

де R - множина вимог до системи; B - множина бізнес-правил, пов'язаних із технологічними процесами, корпоративними регламентами, політиками, стандартами, законодавчими актами, алгоритмами обчислень тощо; T - множина технічних вимог, що встановлюють технічні властивості, якими повинна мати система, наприклад, характеристики продуктивності, надійності та доступності.

Модель вимог представляє собою опис нефункціональних вимог, згрупованих за типами і об'єктами, до яких вони застосовні. Ця модель містить також критерії оцінки одержуваних у результаті навантажувального тестування характеристик та розрахункових показників.

2. Модель системи – визначає структуру системи як мережу систем масового обслуговування (включаючи склад елементів типу «ресурс») [9].

Модель системи може бути представлена у такому вигляді:

$$\Sigma = \{U(p)\}, \{S\}, K_S, K_U \tag{2.3}$$

де$\{U(p)\}$ - множина пристроїв об'єкта тестування з характеристиками продуктивності p ; $\{S\}$ – множина програмних комплексів (компонент, сервісів); K_S- матриця зв'язків між програмними комплексами, рядками якої є джерела, стовпцями - приймачі, а в осередках вказується наявність зв'язку між ними; K_U - матриця зв'язків програмних комплексів із пристроями, що характеризує кількість виділених пристроєм ресурсів для програмного комплексу. Рядками цієї матриці є програмні комплекси, стовпцями – обчислювальні комплекси. Елементами матриці є вектори виділених ресурсів.

Матриці K_S та K_Uдля представленої схеми виглядають так:

$$K_S = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad K_U = \begin{pmatrix} p_1 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 \\ 0 & 0 & p_3 & 0 \\ 0 & 0 & 0 & p_4 \\ 0 & 0 & 0 & p_4 \end{pmatrix},$$

де p_i - Вектор характеристик продуктивності, що характеризує кількість виділених на обчислювальному комплексі ресурсів для заданого програмного комплексу.

Модель системи являє собою опис об'єкта тестування, до якого входять програмні та технічні засоби, їх зв'язки та ресурси, що виділяються (пам'ять, процесор і т.l.). На рис. 2.9 у вигляді графічної схеми представлений приклад моделі системи взаємодії програмних та обчислювальних комплексів.

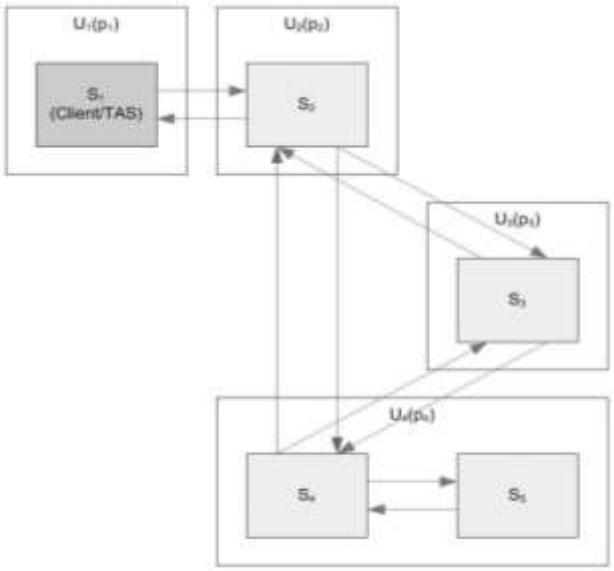


Рис. 2.9. Приклад моделі системи у вигляді графічної схеми

3. Модель навантаження - це опис кількості та типів вимог обслуговування до системи, закон розподілу вимог обслуговування в часі експерименту, правила надходження вимог обслуговування до системи, точки входу вимог обслуговування до системи (логічний рівень) [16].

Модель навантаження може бути представлена у такому вигляді:

$$L = \{ F, M, I \}, \quad (2.4)$$

де F – множина функцій, що характеризують розподіл навантаження, що вводиться в систему; M - багатовимірна матриця, розмірностями якої можуть бути види навантаження: джерела потоків навантаження, найменування та типи потоків, а елементами - їх кількісні характеристики; I - мноожина інтерфейсів для введення навантаження (як посилання на модель системи).

Потік навантаження структурується за його динамічними (F) та статичними (M) властивостями.

Модель навантаження є описом тестових даних і правил їх надходження в систему. Фактично за моделлю навантаження формуються вихідні дані для генерації навантаження інструментальними засобами.

4. Модель вимірювань - визначає склад характеристик, показників і величин, що збираються, інтерфейс введення вимог до системи, метод їх збору та алгоритми перетворення, а також критерії оцінки отриманих результатів [11].

Модель вимірювання призначена для уніфікації опису:

- способів отримання вимірюваних величин при навантажувальному тестуванні ІС;
- типових способів оцінки показників та характеристик;
- загальних властивостей інструментальних засобів для аналізу можливостей їх використання для автоматизації вимірювань.

Модель вимірювань може бути представлена в наступному вигляді:

$$\Delta = \{|U|\}, \tau, \mu, R, \omega , \quad (2.5)$$

де $\{| U |\}$ - перелік вимірюваних величин кожного типу пристроїв інформаційної системи чи її частини; τ - періодичність та шпаруватість вимірювань; μ - множина розрахункових показників та їх взаємозв'язок з вимірюваними величинами; R -

типові правила та алгоритми отримання розрахункових показників; ω – типові критерії оцінки отриманих результатів.

Для планування навантажувального експерименту ключову роль відіграє вибір переліку вимірюваних характеристик продуктивності (табл. 2.1). На основі отриманих значень цих характеристик робляться висновки про результати експерименту.

Таблиця 2.1

Види характеристик продуктивності

Характеристики	Розрахункові показники
Реактивність	Середній час відгуку
	Середній час очікування
	Середній час обслуговування
Продуктивність	Пропускна здатність
	Вироблення
Використання	Утилізація ресурсу
	Відносна пропускна здатність

Реактивність важлива для КІС, експлуатаційними характеристиками ІС можуть бути час відгуку (реакції) системи на запит користувача або час очікування розв'язання задачі (наприклад, може вирішуватись задача мінімізації часу очікування виконання бізнес-транзакції). Характеристики реактивності визначаються як час між отриманням системою вхідних даних та появою відповідних вихідних даних. Реактивність оцінюють або розраховують на основі наступних показників продуктивності:

- середній час відгуку;
- середній час обслуговування;
- середній час очікування.

Час відгуку розраховується як період між початком транзакції та завершенням виконання останнього етапу транзакції.

Час відгуку (Tr) зазвичай складається з часу обслуговування (час, за який проводиться обробка) (Ts) та часу очікування (час очікування ресурсу) (Tw):

$$Tr = Ts + Tw.$$

Показники реактивності залежать від типу системи, структури її точок входу та виходу.

Для КІС важлива їхня інтегральна пропускна здатність, що оцінюється як кількість бізнес-транзакцій, оброблюваних системою в одиницю часу (продуктивність) [15].

При оцінці продуктивності зазвичай використовують безпосередньо виміряні або розрахункові значення наступних показників продуктивності:

- напрацювання (V);
- пропускна здатність (або абсолютна пропускна здатність)

Напрацювання (V) визначається як кількість завершених транзакцій за певний період часу:

$$V_i = \frac{N_i}{T}, \tag{2.6}$$

де $i = 1, ..., n$ - порядковий номер періоду часу; Ni - кількість завершених транзакцій; T - період часу.

Пропускна здатність (C) - максимально можлива кількість завершених транзакцій за одиницю часу:

$$C = max(V1, \ldots, Vn). \tag{2.7}$$

Характеристики використання призначені для того, щоб описати, якою мірою використовуються ресурси системи, що тестується при заданому навантаженні.

До них належать такі показники продуктивності:

- коефіцієнт використання ресурсу;
- відносна пропускна здатність.

Коефіцієнт використання ресурсу показує, яку частину часу ресурс використовується для обслуговування транзакцій. Загальна формула:

$$U = \frac{\sum_{i=1}^{n} t_{Si}}{T} 100\%, \tag{2.8}$$

де n – кількість оброблених транзакцій; t_{Si} - час обслуговування i -ї транзакції; T - період обслуговування.

Характеристики використання ресурсів та їх кількість істотно залежать від

використовуваного в системі обладнання та входять до його складу ресурсів: процесорів, оперативної пам'яті, зовнішніх носіїв, каналів введення-виведення тощо.

Технологія навантажувального тестування, наведена на рис. 2.10.

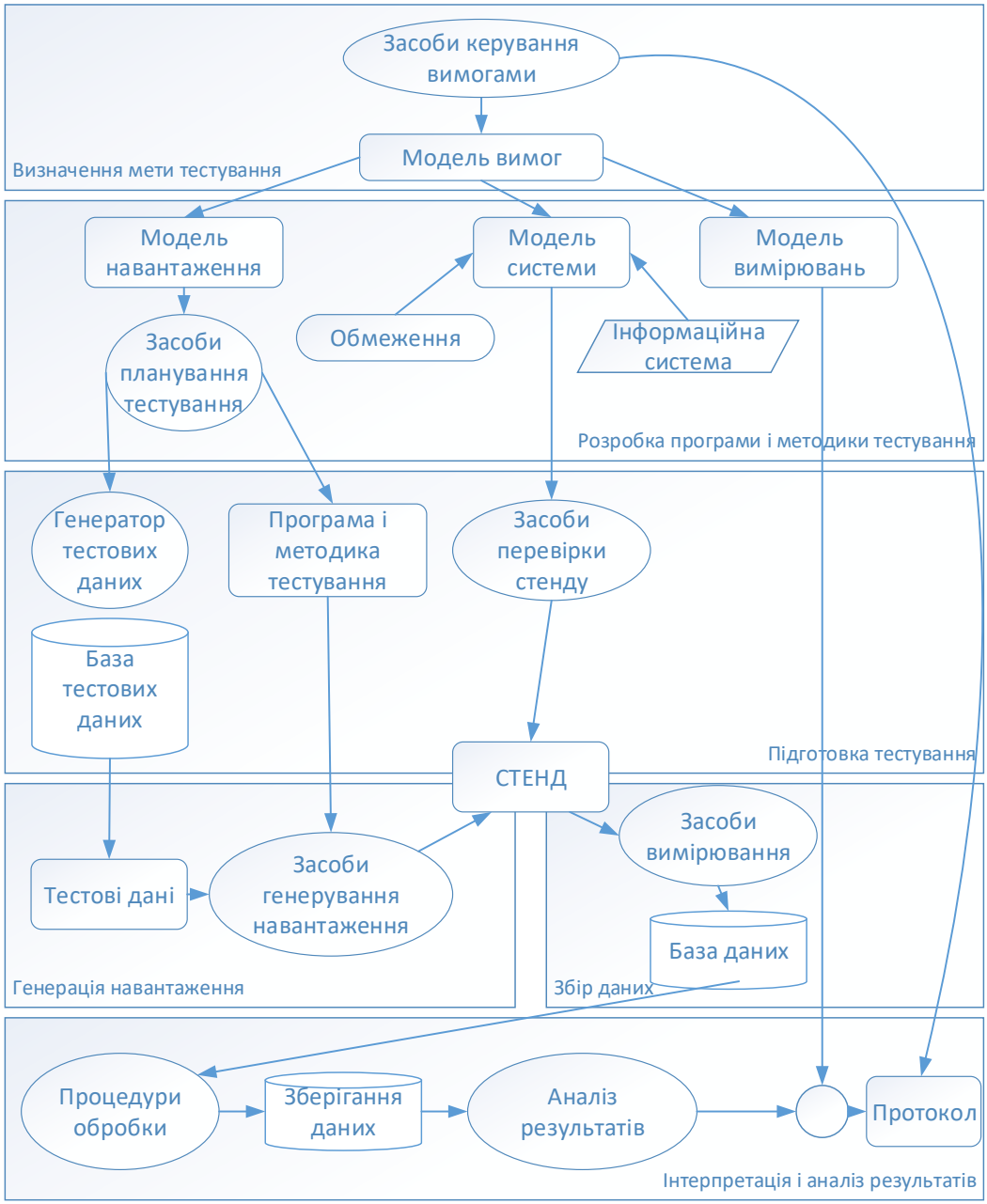


Рис. 2.10. Процес навантажувального тестування з використанням моделей

Основні етапи навантажувального тестування наступні [16]:

- визначення цілей тестування - за допомогою моделі вимог відбувається визначення правил формалізації вимог до експлуатаційних характеристик системи та формується модель вимог;

- розробка програми та методики випробувань - модель навантаження використовується для визначення можливої статичної та динамічної структури та складу потоку навантаження;
- підготовка до тестування - на основі моделі навантаження виконується налаштування засобів планування тестування та генератора тестових даних;
- подача навантаження - засоби автоматизації виконують процедури подачі навантаження точки входу, задані для кожного типу навантаження в моделі системи;
- збирання даних - виконується автоматизований збирання значень вимірюваних характеристик;
- інтерпретація та аналіз результатів - засобами автоматизації використовуються всі чотири моделі: модель вимог для зіставлення результатів експерименту з вимогами до системи; модель навантаження та модель системи забезпечують формування звіту про умови проведення експерименту.

Технологія заснована на використанні моделей, що описують вимоги до проведення експерименту навантаження і властивостей об'єкта тестування, охоплює всі аспекти планування, проведення експерименту навантаження і аналізу його результатів.

Використання моделей істотно (у кілька разів) знижує трудомісткість навантажувального тесту і забезпечує можливість повторного використання підготовленого експерименту, наприклад, при контролі деградації інформаційної системи в її життєвому циклі.

2.3.4.Стрес-тестування КІС

При стрес-тестуванні виконуються дії з метою перевантажити ресурси інформаційної системи з метою виявлення слабких місць [19]. Також проводять стрес тести з використанням методів відключення окремих компонент системи. Даний метод дозволяє визначити межі роботи системи з мінімальними ресурсами.

Мета таких тестів полягає в тому, щоб встановити за яких обставин система відмовить. Також одним із результатів стрес тесту є інформація по відновленню системи (наприклад як буде впливати відключення окремих компонентів системи

від електромережі і за який час система відновить свою роботу після ввімкнення живлення, і чи відновиться взагалі).

Головний результат стрес-тестування полягає в тому, що є можливість отримати звіти після аварії, щоб визначити поведінку інформаційної системи після збою. При цьому КІС не має ставити під загрозу безпеку даних після збою. При успішному стрес-тестуванні КІС має відновитись до нормального стану разом із усіма її програмно-апаратними підсистемами навіть після повного свого відключення і буде в подальшому працювати в штатному режимі.

У додатку Б представлені основні відмінності між цими типами тестування.

В рамках тестування продуктивності збирається статистика використання системи, на основі якої складається профіль навантаження. Після цього обчислюється початкова точка та розмір кроку для збільшення інтенсивності виконання операцій.

У ході виконання тестування визначається максимальна продуктивність системи:

−найбільша інтенсивність виконання операцій з необхідною якістю обслуговування (SLA);

−пікова продуктивність системи, коли він відбувається погіршення показників якості обслуговування операцій (час виконання, відмови).

2.4. Методи та сценарій приймального тестування комп’ютерних інформаційних систем

Приймальний тест - це комплексне тестування, необхідне для визначення рівня готовності системи до подальшої експлуатації. Тестування проводиться на підставі набору тестових сценаріїв, що охоплюють основні бізнес-операції системи.

Ключові переваги:

- дозволяє виявити системні порушення;
- дозволяє виявити дефекти, пов'язані із зручністю та простотою використання.

- залучення досвідчених компетентних фахівців дозволяє якісно та у задані терміни провести процес приймання тестування.

Основні етапи приймального тестування:

1. Підготовка - включає розробку програми та методики випробувань та підготовку приймальних тестів.

2. Проведення – супровід клієнта під час проведення приймальних тестів (заведення дефектів, відстеження коректності та швидкості виконання тестування).

3. Звіт – клієнту надається докладний звіт із переліком помилок, які потрібно усунути перед запуском системи в експлуатацію.

Напрями приймального тестування:

- операційне тестування – перевірка системи на здатність виконувати свою роль у середовищі експлуатації відповідно до бізнес-моделі;
- альфа-тестування - перевірка незалежною командою тестування;
- тестування користувача - перевірка придатності системи для впровадження кінцевими користувачами;
- бета-тестування - тестування зовнішніми користувачами, потенційними клієнтами.

Сценарій приймального тестування повинен мати чітко сформульовані розділи, представлені на рис. 2.11.

Найбільш повним та різностороннім приймальним випробуванням має піддаватися перша базова версія КІС. Комплексне тестування КІС гарантує перевірку виконання всіх вимог технічного завдання. Запропонований сценарій проведення приймального тестування дозволяє фіксувати умови неправильної роботи КІС та характер прояву дефектів.

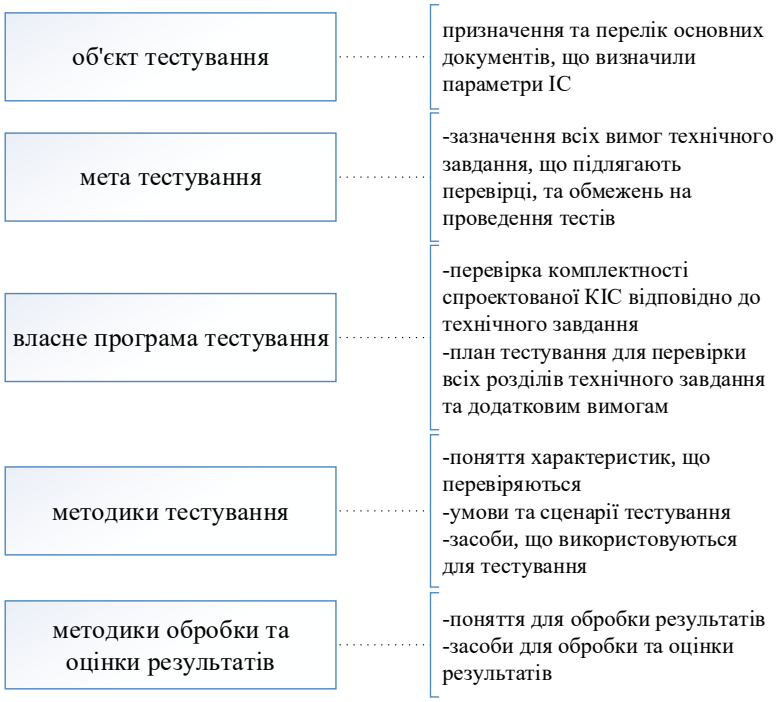


Рис. 2.11. Сценарій приймального тестування

При завершальних випробуваннях основна увага, крім перевірок функціональної придатності, має зосереджуватись на підготовці стресових тестів, тестуванні в режимах граничного використання ресурсів, надійності функціонування КІС. Для цього необхідно проводити випробування у два послідовні етапи - Альфа- і Бета-тестування [8, 13]. Обсяг тестів та тривалість обох етапів тестування залежатимуть від складності комплексу програмного забезпечення КІС та інтенсивності потоку змін.

РОЗДІЛ 3
АПРОБАЦІЯ МЕТОДІВ ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Сценарій навантажувального тестування з використанням різних інструментальних засобів

Розглянемо навантажувальне тестування для оцінки продуктивності простого HTTP -запиту GET з 20 потоків з 100 000 ітерацій.

Сторона сервера (додаток, що тестується): Процесор: 4x Xeon L5520 @ 2,27 ГГц; RAM: 8 ГБ; ОС: Microsoft Windows Server 2008 R2 x64; Сервер програм: IIS 7.5.7600.16385.

Клієнтська сторона (генератор навантаження): Процесор: 4x Xeon L5520 @ 2,27 ГГц; RAM: 4 ГБ; ОС: Ubuntu Server 12.04 64-бітна.

Для проведення тестування будемо використовувати інструментальні засоби, кожен інструмент надсилатиме запити так швидко, як може:

1. Grinder - це безкоштовне середовище тесту навантаження на основі Java, доступне за ліцензією BSD в стилі open-source.

На рис. 3.1 представлений фрагмент проведення тесту навантаження, виконаного з використанням Grinder.

Рис. 3.1. Навантажувальне тестування за допомогою Grinder

2. Gatling Project - це інструмент для тестування продуктивності з відкритим вихідним кодом, який в основному розроблений та підтримується Stephane Landelle.

В додатку В наведено приклад звіту Gatling для сценарію завантаження.

3. Tsung - інструмент для тестування продуктивності з відкритим кодом.

Tsung використовує Erlang. Tsung не надає графічного інтерфейсу для розробки або виконання тестів.

Виклик tsung виводить результат, представлений рис. 3.2.

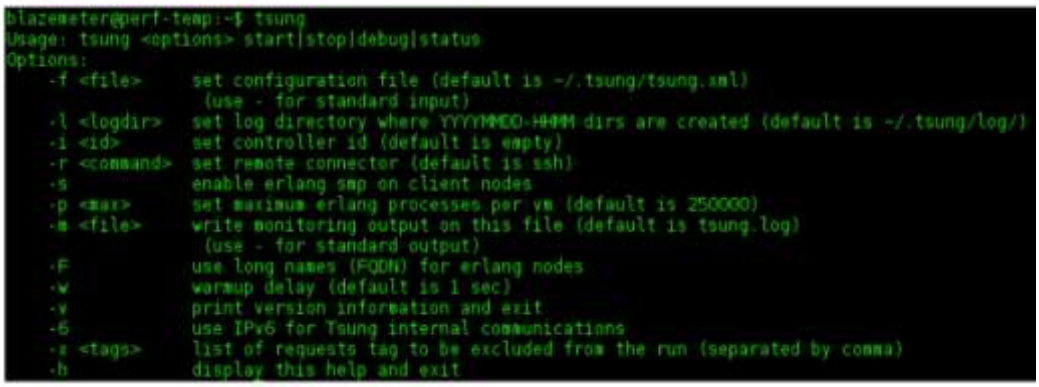


Рис. 3.2. Виклик сценарію tsung

На рисунках 3.3 наведено статистичний звіт навантажувального тестування, 3.4 - графічний звіт.

Рис. 3.3. Статистичний звіт навантажувального тестування

Рис. 3.4. Графічний звіт навантажувального тестування

4. Apache JMeter™ - має зручний графічний інтерфейс, що значно полегшує розробку тестів та налагодження. Додаток JMeter з агрегованим звітом за сценарієм навантаження (рис. 3.5).

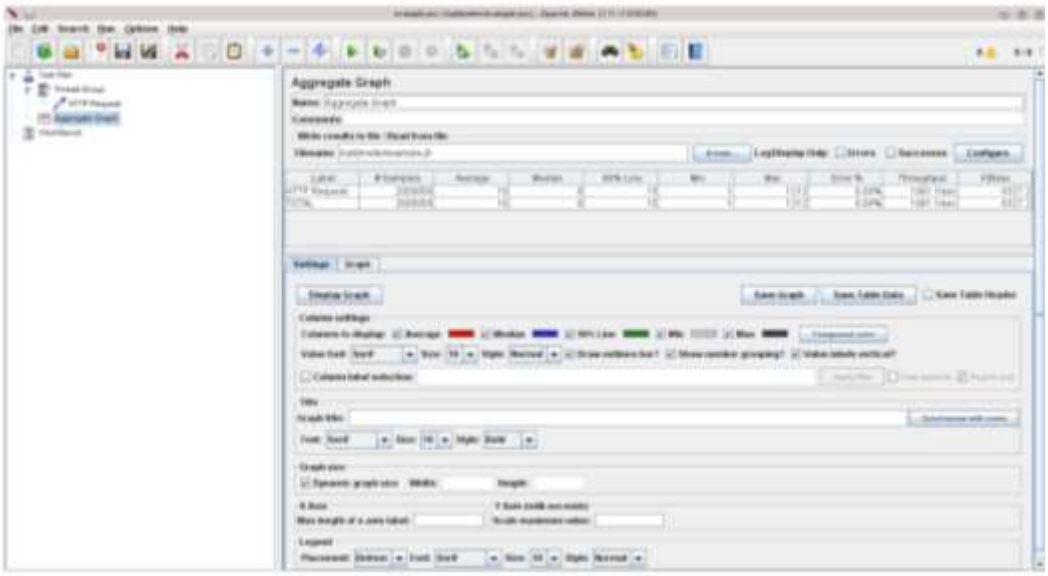


Рис. 3.5. Агрегований звіт сценарію навантаження JMeter

5. Locust засноване на Python середовищі з відкритим вихідним кодом, яке дозволяє писати скрипти продуктивності мовою Python. Приклад базового тестового сценарію з висновком представлено рис. 3.6.

```
from locust import HttpLocust, TaskSet, task
class SimpleLocustTest(TaskSet):
 @task
 def get_something(self):
 self.client.get("/")
class LocustTests(HttpLocust):
 task_set = SimpleLocustTest
```

```
→ JMeterVsLocust locust -f simple_locust_script.py --host=http://192.168.1.170:8080
[2022-11-18 14:46:38,458] Yuris-MBP-2.home/INFO/locust.main: Starting web monitor at *:8089
[2022-11-18 14:46:38,459] Yuris-MBP-2.home/INFO/locust.main: Starting Locust 0.8.1
```

Рис. 3.6. Приклад тестового сценарію Locust

Після виконання скрипта відображається докладна звітність (рис. 3.7).

Порівняємо результати тесту навантаження цих інструментів з наступними показниками:

1. Середній час відгуку (мс).
2. Середня пропускна здатність (запитів/сек).

3. Загальний час виконання тесту (хвилин).

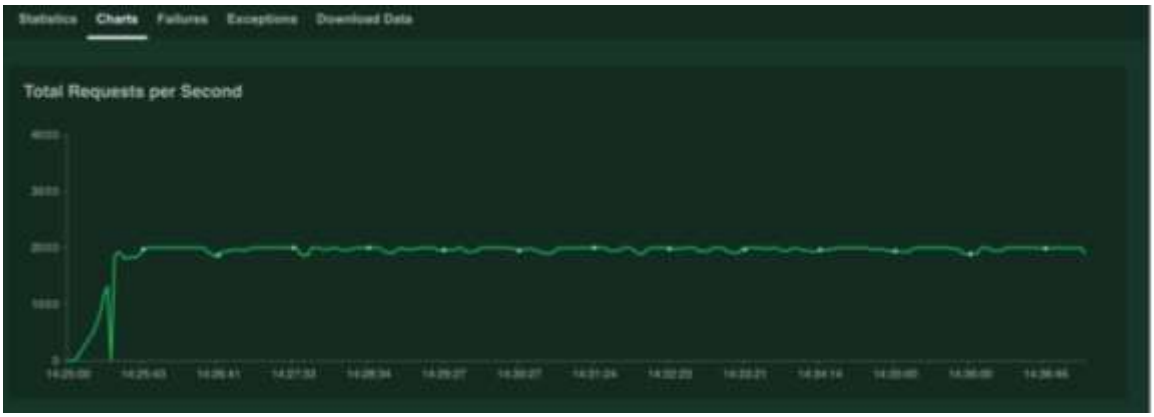


Рис. 3.7. Звітність тестового сценарію Locust

На рисунках 3.8, 3.9 представлені діаграми, що відображають середню відповідь та загальний час виконання тесту.

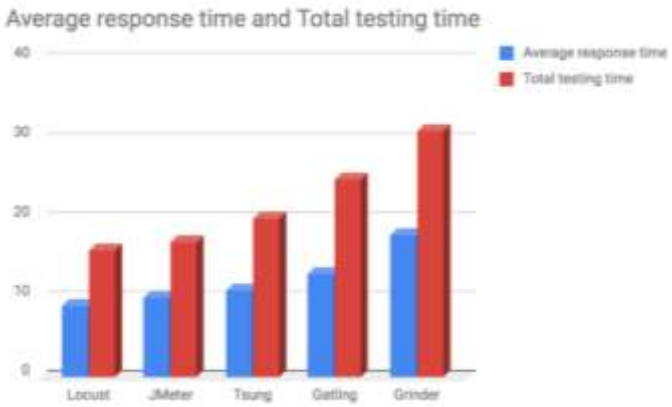


Рис. 3.8. Процес навантажувального тестування з використанням моделей

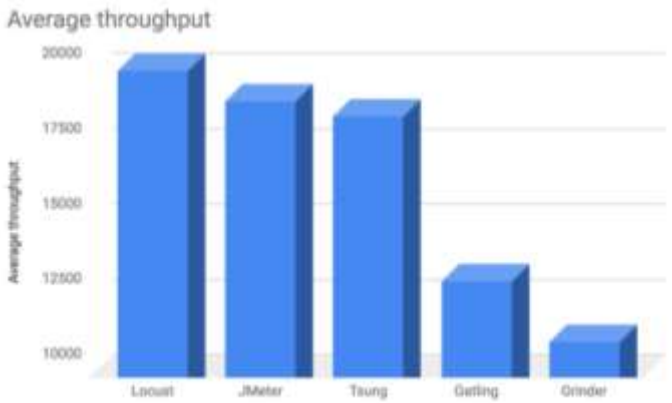


Рис. 3.9. Процес навантажувального тестування з використанням моделей

Як показано на графіках, у Locust найшвидший час відгуку з найвищим середнім значенням, за яким йдуть JMeter, Tsung і Gatling. У Grinder найповільніший час із найнижчою середньою пропускною здатністю.

У додатку Г наведено порівняння ключових функцій, інструментів тестування.

Застосувавши вибрані інструментальні засоби для проведення навантажувального тесту, були отримані наступні результати: час вибірки постійно знаходиться між 128 і 164 мс. Географічна відстань є найважливішим фактором, що впливає на час вибірки, якщо ваш сервер має достатні ресурси. У тестовому випадку сервер добре відреагував на 50 користувачів, які надсилають запити протягом 10 секунд.

Тепер настав час спробувати збільшити навантаження та подивитися, наскільки добре реагує система. Навантаження збільшується до 80 за 10 секунд. Нижче наведено результати (рис. 3.10).

Label	Sample Time(ms)	Status	Bytes	Latency
HTTP Request	953		55065	546
HTTP Request	880		55065	479
HTTP Request	941		55065	529
HTTP Request	924		55065	502
HTTP Request	815		55065	398

Рис. 3.10. Результати тестування зі збільшенням навантаження

Сервер явно починає обтяжуватись запитами (рис. 3.11). Запустимо тест JMeter для оцінки використання ресурсів VPS (рис. 3.12).

```
top - 16:52:25 up 5 days, 23:18,  1 user,  load average: 0.06, 0.16, 0.13
Tasks:  74 total,   1 running,  73 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa   0.0 hi   0.0 si,  0.0 st
KiB Mem:    501868 total,   409232 used,    92636 free,    28132 buffers
```

Рис. 3.11. Результати тестування зі збільшенням навантаження без використання JMeter

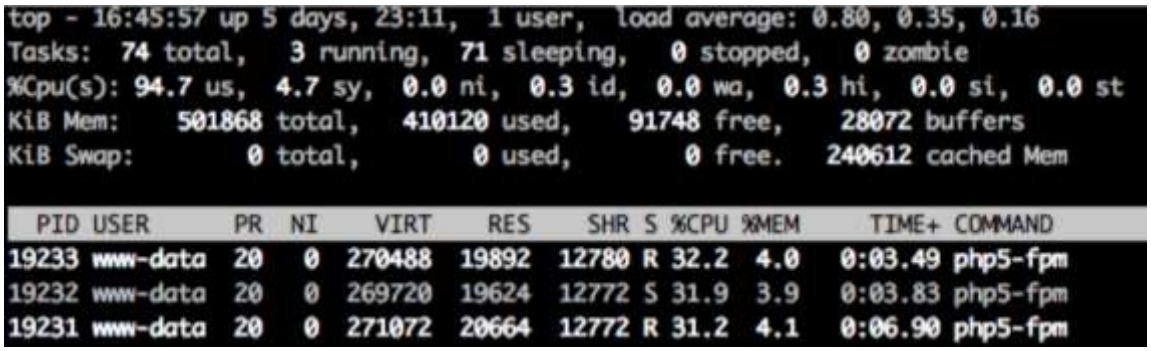


Рис. 3.12. Результати тестування зі збільшенням навантаження з використанням JMeter

Очевидно, що є суттєве споживання ресурсів процесора та оперативної пам'яті. Для задоволення зростаючих робочих навантажень сервери повинні бути оптимізовані, або потрібно збільшувати ЦП. Також можна розмістити базу даних на іншому сервері.

Таким чином, можна підтримувати навантаження доти, доки не відбудеться істотне зниження продуктивності. Для перевірки продуктивності можна використовувати інструментальні засоби, наприклад, JMeter має багато застосувань у контексті підвищення продуктивності та оптимізації серверів.

3.2. Тестування продуктивності та надійності бази даних КІС

Розглянемо тестування надійності та продуктивності КІС. Характеристики та список обладнання для проведення навантажувального тестування наведено у додатку Д.

Загальними можливостями СУБД, що використовується під час роботи КІС, є:

- підтримка реляційної чи об'єктно-реляційної моделі бази даних;
- підтримка міжнародного стандарту ANSI SQL-92 та вище;
- наявність засобів створення індексів та кластерів даних;
- автоматичне відновлення бази даних;
- підтримка мережевих протоколів TCP/IP;
- можливість контролю доступу до даних;

- централізоване керування обліковими записами користувачів; - Оптимізація запитів.

Тестування здійснюється ітераційно, збільшуючи на кожній ітерації кількість сесій тестів, що одночасно працюють. Тестування припиняється за одним із перерахованих вище критеріїв. На кожній ітерації фіксуються показники продуктивності. Після закінчення тестування з знятим показниками робляться висновки про кількість користувачів, що одночасно працюють.

Для результативного проведення процедури тестування продуктивності КІС необхідне певне технічне та програмне забезпечення завдання. Проводити збір та оцінку характеристик функціонування СУБД не потрібно.

Визначено види застосовуваного тесту продуктивності:

1. Визначення максимальної продуктивності – реалізується сценарієм №2. Тест завершується, коли час відгуку перевищив допустимі межі; кількість неуспішних операцій збільшилася до критичного (понад 10%) рівня; вичерпано системні чи апаратні ресурси. Результатом тестування є максимальний досягнутий рівень навантаження (позначається *L max*).

2. Тест надійності – реалізується сценарієм № 1. Критерієм успішності тестування є: відсутність деградації продуктивності системи в ході тесту (інтенсивності та часу відгуку); відсутність витоку ресурсів протягом тесту.

3. Тест відмовостійкості виконується на рівні типового навантаження, який зазвичай встановлюється на рівні 70% від максимальної *Lmax*. Критеріями успішного проходження системою тесту є:

- система зберегла доступність у функціоналі, не пов'язаному з функціями суміжної системи;
- система відновилася протягом необхідного часу відновлення доступності.

Критеріями успішного завершення навантажувальне тестування є виконання всіх запланованих тестів і отримання даних моніторингу.

Для проведення навантажувального тестування розроблено два сценарії тестування. Алгоритм процедури представлений стандартним чином (рис . 3.13).

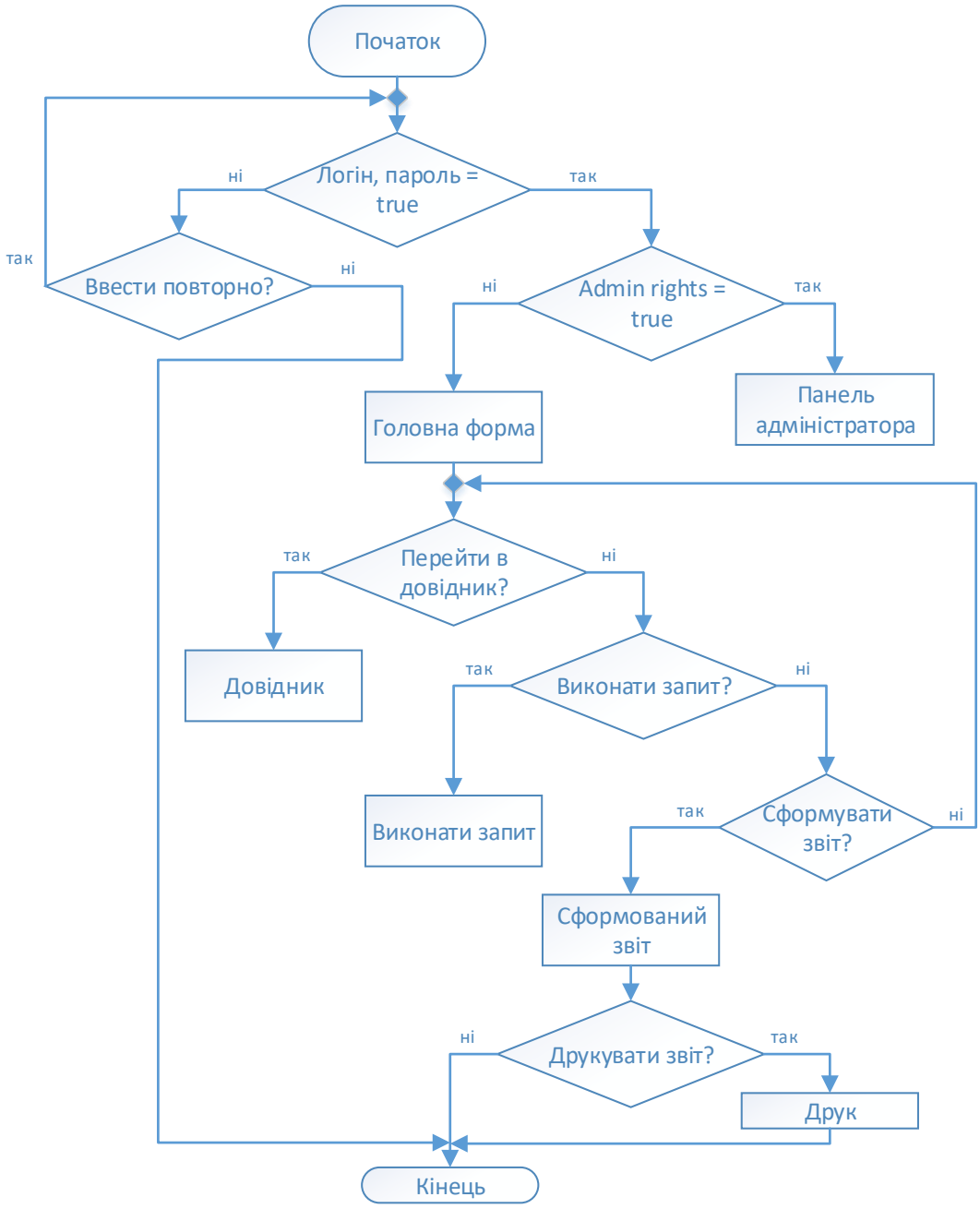


Рис. 3.13. Алгоритм процедури тестування за сценарієм 1

Імітація DDoS атаки (рис. 3.14). Одночасна робота в одному з довідників системи максимальної кількості користувачів – 30-50.

За умови успішності сценарію кількість користувачів збільшується на 10 од. і тестування триває. Визначається межа можливостей КІС.

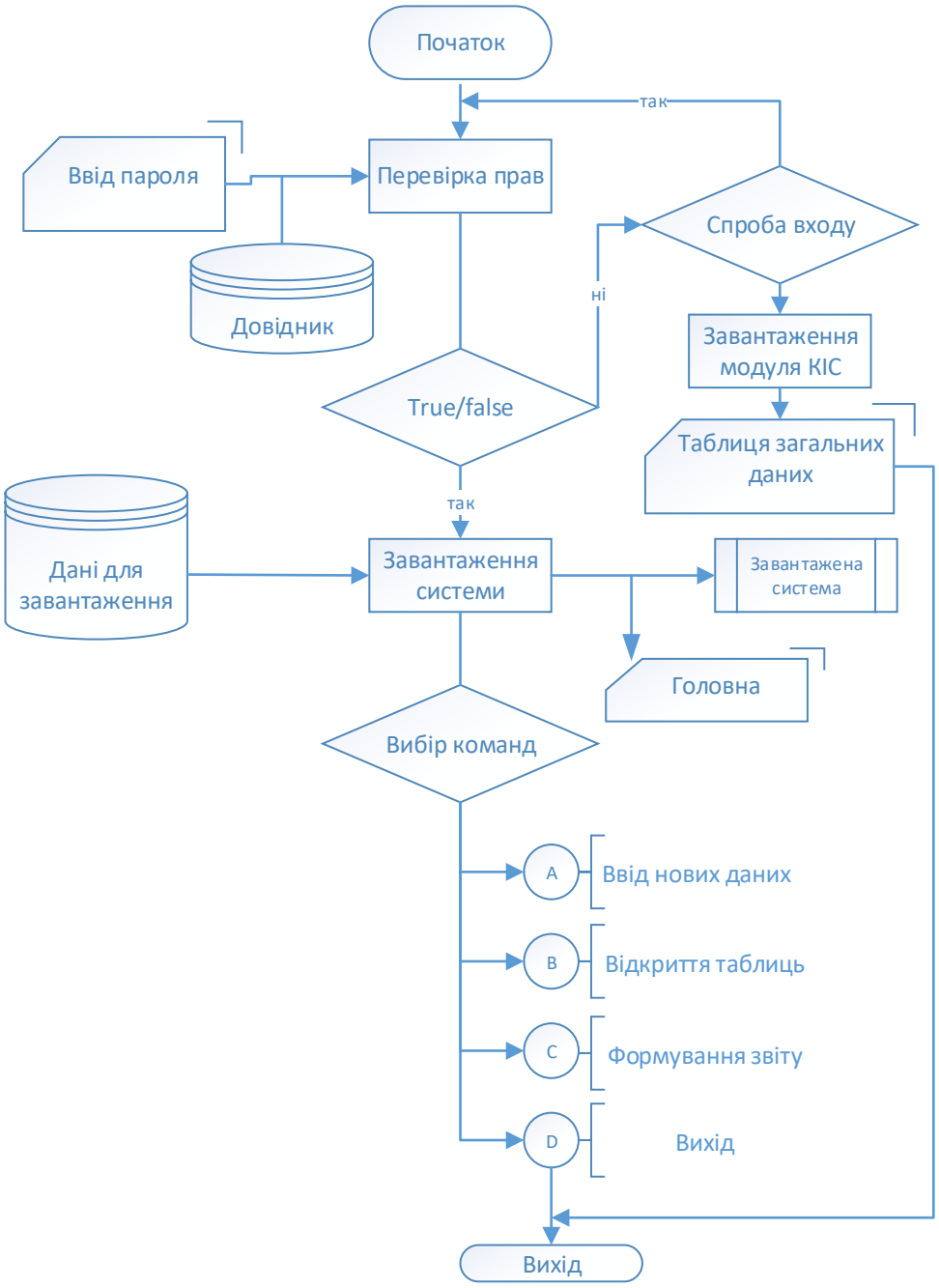


Рис. 3.14. Блок-схема алгоритму роботи користувача при імітації DDoS атаки

Перед початком ітераційного навантажувального тестування повинна бути розроблена програма та методика випробувань, що включає:

- опис цілей та завдань тестування;
- опис об'єкта тестування, включаючи опис основних бізнес-процесів системи;
- архітектурну схему об'єкту тестування;
- короткий опис підходу до тестування та способів генерації навантаження;

- опис профілів навантаження, включаючи перелік операцій, що виконуються віртуальними користувачами та зовнішніми системами через відповідні інтерфейси; інтенсивність операцій, що виконуються віртуальними користувачами та зовнішніми системами;
- список запланованих тестів;
- типові сценарії тестування;
- тестові скрипти;
- вимоги до тестових даних;
- вимоги до продуктивності;
- вимоги до моніторингу продуктивності;
- метрики продуктивності;
- обмеження тестування;
- ключові показники досягнення цілей;
- ризики проекту.

Навантажувальне тестування КІС виконується на заздалегідь підготовленій базі з наступними параметрами:

1. Кількість співробітників, що одночасно працюють, до 50.
2. Кількість об'єктів обслуговування – 10 000.
3. Кількість видів обслуговування для кожного об'єкта – 4.
4. Кількість контрольованих показників кожного об'єкта 4-5.
5. Кількість показників напрацювання для кожного об'єкта – 4-5.

Після задоволення вимог щодо технічного та програмного забезпечення запускається навантажувальний тест, та оцінюється швидкість виконання операцій та інші параметри відповідно до запропонованої методики. Далі результат порівнюється з "еталонними" значеннями. За результатами порівняння приймається рішення про необхідність оптимізації та випуск оновлень.

Передача інформації в КІС здійснюється шляхом електронної пошти, локальною мережею або у формі документів. Достовірність введення даних буде реалізована за допомогою лічильного контролю. Формування звітів є механізм, заснований на використанні шаблонів звітів. Шаблони звітів є параметризованими

звітами.

Аутентифікація користувачів у КІС базується на механізмах, вбудованих у використовувану СУБД (Microsoft SQL Server або Sybase Adaptive Server). При цьому використовується концепція «наскрізної» автентифікації та авторизації, яка передбачає використання єдиного облікового запису як на рівні системи, так і на рівні СУБД.

Для захисту інформації від несанкціонованого доступу та зміни особами, які мають адміністративні повноваження, в системі вжито спеціальних заходів щодо поділу сфер відповідальності адміністраторів та обмеження їх повноважень.

При проведенні тестів важливим питанням є отримання пропорційних результатів відповіді на аналогічних тестах продуктивності. Актуальність цього питання пов'язана з тим, що результати тесту залежать не тільки від параметрів апаратного та програмного налаштування сервера СУБД, а й від побудови самого тесту. Налаштування СУБД доцільно виконувати за результатами двох або більше продуктивних тестів. Тому, крім тестування, заснованого на марківських моделях поведінки клієнтів СУБД, доцільно проводити тестування одним або декількома синтетичними тестами, структура запитів яких найбільше збігаються зі структурою запитів КІС. Для з'ясування ступеня цієї відповідності, побудовані раніше марківські моделі, необхідно доповнити інформацією структуру запитів КІС.

Тестування продуктивності дозволяє визначити ступінь готовності системи до позаштатних ситуацій (відмова обладнання, DDoS- атаки), рівень її надійності та здатність до самовідновлення. Також навантажувальні випробування допомагають розробити комплекс адекватних заходів для підвищення продуктивності системи, її стійкості та захищеності корпоративного оточення.

3.3. Сценарій навантажувального тестування та формування критеріїв на доопрацювання

Як приклад зробимо навантаження тестування КІС під 30 користувачами одночасно, які послідовно входять в систему з інтервалом 3 секунди. Динаміка

навантаження представлена рис. 3.15.

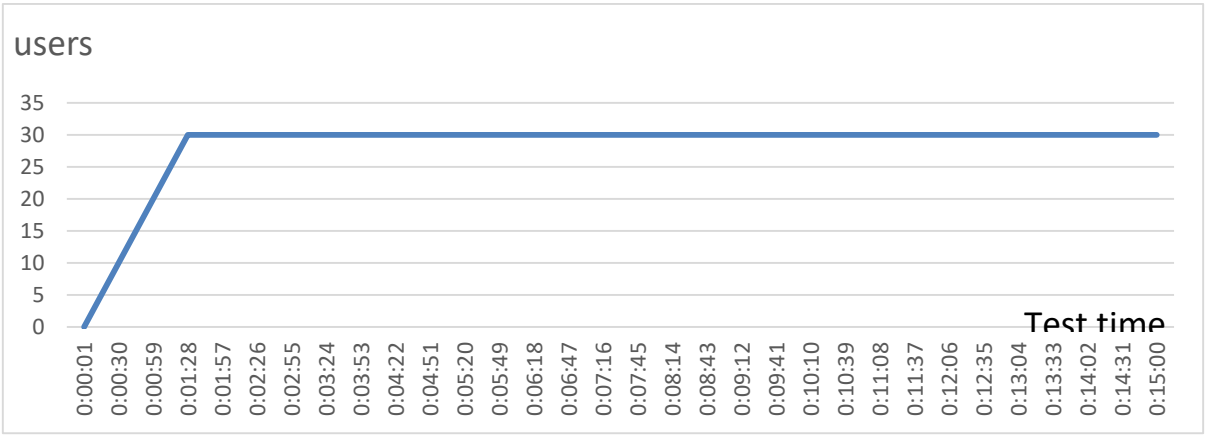


Рис. 3.15. Динаміка навантаження

Усі 30 користувачів починають працювати з системою через 90 секунд з моменту початку тесту навантаження. Розподіл часу відгуку по транзакціях щодо початку тесту навантаження наведено на рис. 3.16.

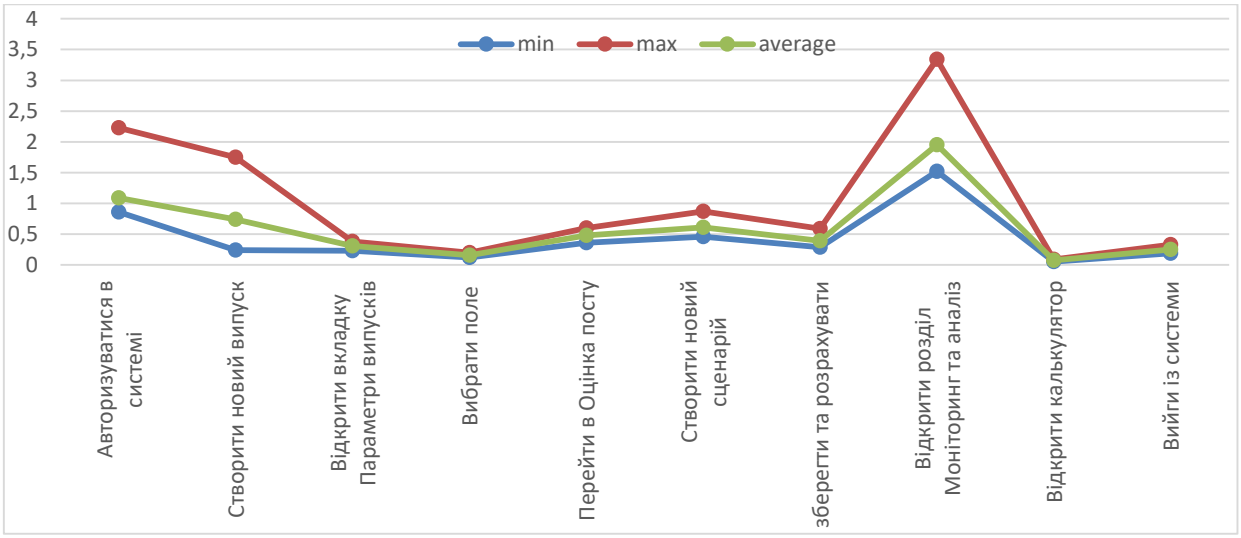


Рис. 3.16. Час відгуку системи

У процесі виконання сценарію усі транзакції було виконано успішно. У додатку Е наведено дані щодо розподілу часу відгуку.

За результатами таблиці 3.4 видно, що час відгуку системи відповідає вимогам продуктивності при навантаженні 30 користувачів. Найменш продуктивними є

транзакції: «13. Вибрати таблицю 1 та Сформувати», «15. Вибрати таблицю 2 та Сформувати», «Відкрити вкладку «Моніторинг», рекомендується звернути увагу на їхню оптимізацію.

У процесі навантажувальне тестування необхідно знімати лічильники продуктивності з сервера додатків. Результати подано на рисунках 3.17-3.18.

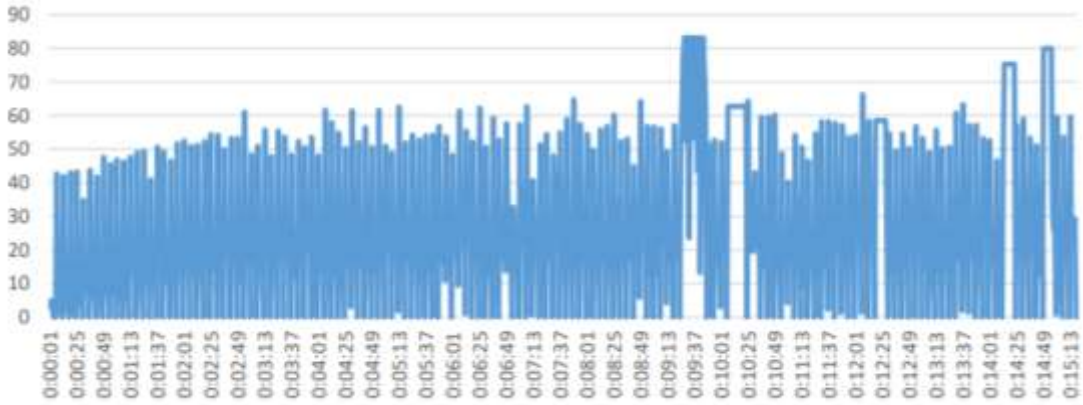


Рис. 3.17. Використання ресурсів CPU сервера додатків

Рис. 3.18. Використання пам'яті сервера програм

Також необхідно знімати лічильники продуктивності із сервера бази даних (БД). Результати подано на рисунках 3.19-3.20.

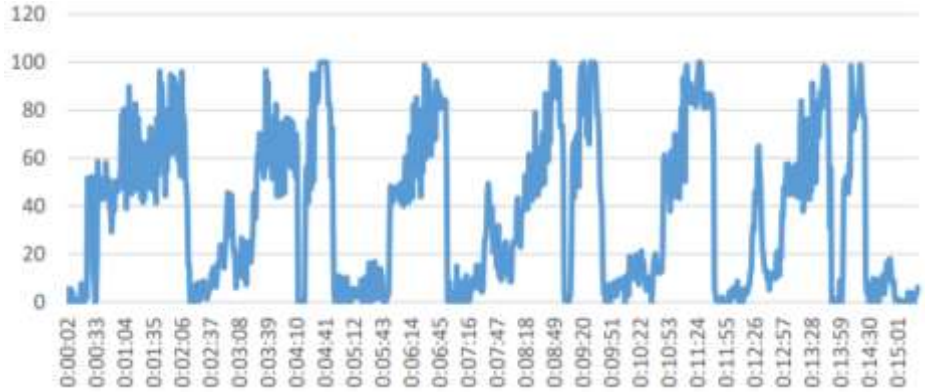


Рис. 3.19. Використання пам’яті сервера БД

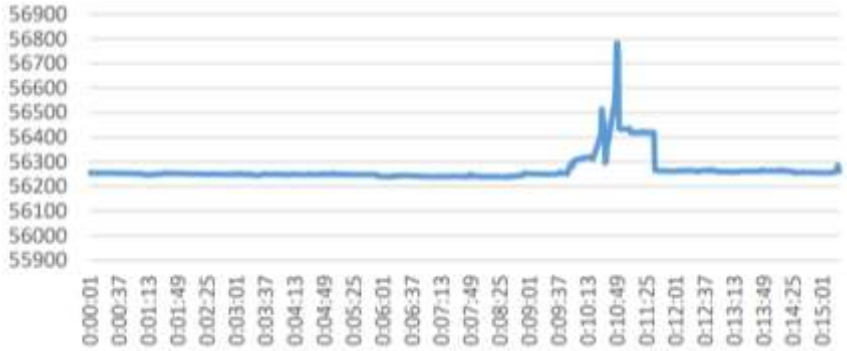


Рис. 3.20. Використання ресурсів CPU сервера БД

З графіків видно, що протягом усього виконання сценарію спостерігається середнє завантаження CPU сервера в додатку 60% з піками до 85%, середнє сервера БД 60% з піками до 100%. Використання пам'яті сервером БД і додатків перебуває в середньому рівні 10,4-11,3 гігабайт і 56,2-56,8 гігабайт.

В результаті проведеного тесту навантаження можна зробити наступні висновки:

- CPU сервера БД, ймовірно, не є вузьким місцем, що обмежує продуктивність системи;
- CPU сервера додатків, ймовірно, є вузьким місцем, що обмежує продуктивність системи;
- споживання пам'яті сервера додатків та сервера БД не є вузьким місцем, що обмежує продуктивність системи;
- на основі аналізу лічильників продуктивності робота дискової підсистеми не є вузьким місцем; аналогічно завантаження клієнтської станції.

- як рекомендація розробникам системи слід звернути увагу на оптимізацію найменш продуктивних транзакцій «13. Вибрати таблицю 1 та Сформувати», «15. Вибрати таблицю 2 та Сформувати», «Відкрити вкладку «Моніторинг».

3.4. Інтеграційне тестування комп’ютерної інформаційної системи

Проведено інтеграційне тестування за допомогою платформи Citrus Framework.

Citrus - це інтегроване середовище тестування, написане на Java, яке тестує КІС щодо відповідності середовищу клієнта. Інструмент імітує навколишні системи через різні порти (http, JMS, TCP/IP, SOAP, ...), щоб виконати автоматичне наскрізне тестування варіанта використання.

Складаючи можливі сценарії, здійснюється перевірка КІС, щоб мати можливість відправляти повідомлення за різними протоколами та виконувати всю кореляцію повідомлень, які надходять та зрештою прибувають до цільового пункту призначення.

Щоб змоделювати один із цих сценаріїв, був використаний Citrus Framework. Для перевірки його роботи була змодельована структура каталогів (рис. 3.21):

- src / citrus / java - місце розташування згенерованих тестових прикладів Testng;
- src / citrus / resoruces - розташування всієї конфігурації, необхідної для фреймворку;
- src/citrus/tests- розташування всіх певних тестових випадків;
- lib – розташування всіх бібліотек залежностей;
- log – розташування, в якому Citrus Framework зберігатиме логи (фрагменти корисного навантаження);
- build.xml - скрипт складання ant.

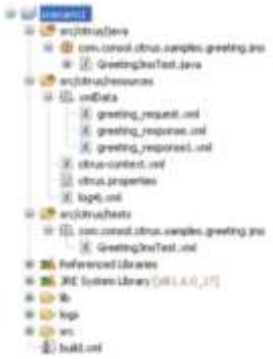


Рис. 3.21. Структура каталогів

На рис. 3.22 наведено приклад використовуваного build.xml.

```
<project name="greetings" basedir="." default="citrus.run.tests">

    <property file="src/citrus/resources/citrus.properties"/>

    <path id="citrus-classpath">
        <pathelement path="src/citrus/java"/>
        <fileset dir="lib">
            <include name="*.jar"/>
        </fileset>
    </path>

    <typedef resource="citrustasks" classpath="lib/citrus-ant-tasks-1.1-SNAPSHOT.jar"/>

    <target name="compile.tests">
        <javac srcdir="src/citrus/java" classpathref="citrus-classpath"/>
        <javac srcdir="src/citrus/tests" classpathref="citrus-classpath"/>
    </target>

    <target name="citrus.run.tests" depends="compile.tests" description="Runs all Citrus
        <citrus suitename="${testsuite.name}" package="com.consol.citrus.samples.*"/>
    </target>

    <target name="citrus.run.single.test" depends="compile.tests" description="Runs a si
        <touch file="test.history"/>

        <loadproperties srcfile="test.history"/>

        <echo message="Last test executed: ${last.test.executed}"/>

        <input message="Enter test name:" addproperty="testclass" defaultvalue="${last.t

        <propertyfile file="test.history">
            <entry key="last.test.executed" type="string" value="${testclass}"/>
        </propertyfile>

        <citrus suitename="citrus-samples" test="${testclass}"/>
    </target>

    <target name="create.test" description="Creates a new empty test case">
        <input message="Enter test name:" addproperty="test.name"/>
        <input message="Enter test description:" addproperty="test.description" defaultv
        <input message="Enter author's name:" addproperty="test.author" defaultvalue="${
        <input message="Enter package:" addproperty="test.package" defaultvalue="${defau
        <input message="Enter framework!" addproperty="test.framework" defaultvalue="tes

        <java classname="com.consol.citrus.util.TestCaseCreator">
            <classpath refid="citrus-classpath"/>
            <arg line="-name ${test.name} -author ${test.author} -description ${test.des
        </java>
    </target>

    <target name="create.html.doc" description="Creates test documentation in html">
        <mkdir dir="target"/>

        <java classname="com.consol.citrus.doc.HtmlTestDocGenerator">
            <classpath refid="citrus-classpath" />

            <arg value="src/citrus/tests"/>
            <arg value="target/CitrusTests.html"/>
            <arg value="Citrus Test Documentation"/>
            <arg value="logo.png"/>
            <arg value="Overview"/>
        </java>

        <copy todir="target" file="src/citrus/resources/logo.png"/>

        <zip destfile="target/CitrusTests.zip">
            <fileset dir="target">
                <include name="CitrusTests.html"/>
                <include name="logo.png"/>
            </fileset>
        </zip>
    </target>

    <target name="create.xls.doc" description="Creates test documentation in excel">
        <mkdir dir="target"/>

        <java classname="com.consol.citrus.doc.ExcelTestDocGenerator">
            <classpath refid="citrus-classpath" />

            <arg value="src/citrus/tests"/>
            <arg value="CitrusTests"/>
            <arg value="Citrus Test Documentation"/>
            <arg value="Citrus Testframework"/>
            <arg value="ConSol* Software GmbH"/>
        </java>
    </target>
</project>
```

Рис. 3.22. Приклад використовуваного build.xml

ПЗ КІС за сценарієм отримує нове повідомлення у черзі. Якийсь процес отримує повідомлення, виконує його перетворення / перевірку та передає його. Зрештою, нове повідомлення буде надіслано в іншу чергу.

Стандартна конфігурація описує використання:

<citrus:jms-message-sender id="getOrdersRequestSender" destination-name="testJMSServer/citrus_queue_in"/>

Потрібно налаштувати src / citrus / tests / com / consol / citrus / samples / reeting / jms / GreetingJmsTest.xml (рис. 3.23).

```
<?xml version="1.0" encoding="UTF-8"?>
<spring:beans xmlns="http://www.citrusframework.org/schema/testcase" xmlns:spring="http:.
    <testcase name="GreetingJmsTest">
        <meta-info>
            <author>Eric Elzinga</author>
            <creationdate>2010-03-25</creationdate>
            <status>FINAL</status>
            <last-updated-by>Eric Elzinga</last-updated-by>
            <last-updated-on>2010-03-28T00:00:00</last-updated-on>
        </meta-info>

        <variables>
            <variable name="correlationId" value="citrus:randomNumber(10)"></variable>
            <variable name="user" value="Eric"></variable>
        </variables>

        <actions>
            <send with="sendGreeting">
            <message>
                <resource file="classpath:xmlData/greeting_request.xml" />
            </message>
            <header>
                <element name="Operation" value="sayHello"/>
                <element name="CorrelationId" value="${correlationId}"/>
            </header>
        </send>

        <receive with="receiveGreeting">
        <selector>
            <value>CorrelationId = '${correlationId}'</value>
        </selector>
        <message>
          <resource file="classpath:xmlData/greeting_response.xml" />
          <ignore path="//tns:GreetingRequestMessage/tns:Text" />
        </message>
        <header>
            <element name="Operation" value="sayHello"/>
            <element name="CorrelationId" value="${correlationId}"/>
        </header>
    </receive>
</actions>
</testcase>
</spring:beans>
```

Рис. 3.23. Файл конфігурації

Щоб увімкнути корисне навантаження тесту, можна використовувати такі команди:

```
<message>
    <resource file="classpath:xmlData/greeting_request.xml" />
</message>
```

Рис.3.24. Увімкнення навантаження

або

```
<message>
    <data>
        <![CDATA[
            <tns:GreetingRequestMessage xmlns:tns="http://www.citrusframework.org/samp
                <tns:CorrelationId>${correlationId}</tns:CorrelationId>
                <tns:Operation>sayHello</tns:Operation>
                <tns:User>${user}</tns:User>
                <tns:Text>Hello Citrus!</tns:Text>
            </tns:GreetingRequestMessage>
        ]]>
    </data>
</message>
```

Рис.3.25. Увімкнення навантаження

Таким чином, була проведена перевірка повідомлень XML та отримуємо результат, наведений на рис. 3.26

```
[citrus] 4547   INFO  port.LoggingReporter| TEST FINISHED: GreetingJmsTest
[citrus] 4547   INFO  port.LoggingReporter| ----------------------------------------
[citrus] 4563   INFO  port.LoggingReporter| FINISH TESTSUITE citrus-samples-greeting
[citrus] 4563   INFO  port.LoggingReporter| ----------------------------------------
[citrus] 4563   INFO  port.LoggingReporter| ________________________________________
[citrus] 4563   INFO  port.LoggingReporter|
[citrus] 4563   INFO  port.LoggingReporter| CITRUS TEST RESULTS
[citrus] 4563   INFO  port.LoggingReporter|
[citrus] 4563   INFO  port.LoggingReporter|  GreetingJmsTest ..........................
[citrus] 4563   INFO  port.LoggingReporter|
[citrus] 4563   INFO  port.LoggingReporter| Total number of tests: 1
[citrus] 4563   INFO  port.LoggingReporter| Skipped: 0 (0.0%)
```

Рис. 3.26. Результат тесту

Даним прикладом була продемонстрована можливість перевірки - чи можна помістити повідомлення в чергу та отримати відповідь назад у чергу; чи перевіряє відповідь xml відповідність даному визначенню схеми (xsd).

На рис. 3.27 наведено результати тестування інтеграції за допомогою soapUI.

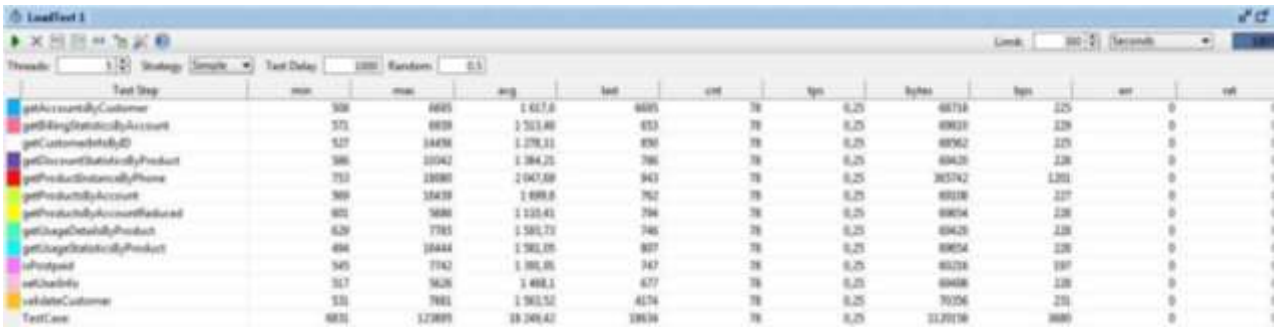


Рис. 3.27. Результати навантажувальне тестування Wizard API

Для тестування було визначено 1000 запитів з одночасним надсиланням 5 запитів на сервер. Середній час повного завантаження сторінки під час роботи одночасно 50 людина становить близько 25 секунд, що у межах допустимості. Швидкість передачі між системами визначається інтервалах: до 30 с - висока, 30 - 50 - середня, від 50 - низька.

Таким чином, на даному прикладі була продемонстрована можливість приймального тестування показати, що всі модулі інтегровані між собою та мають гарну взаємодію, при роботі не виявлено дефектів. І це підтверджується досвідченим шляхом.

3.5. Аналіз ефективності застосування комплексного тестування комп’ютерних інформаційних систем

Перед будь-яким інтеграційним процесом завжди стояло завдання покращення роботи комплексу інформаційних систем. Як критерії оцінки добре побудованого інтеграційного процесу можна виділити такі:

- швидкість передачі даних;
- ефективність обміну даними;
- оцінка користувачів інтеграційних систем

На рис. 3.28 представлена гістограма, що відображає різницю швидкості роботи між інтеграційними операціями до і після апробації сценарію тестування.

За даними гістограми можна зробити висновок, що запропонований сценарій тестування, що включає методи комплексного тестування, сприяє підвищенню

якості роботи КІС, але для чистоти експерименту, щоб оцінити достовірність результатів дослідження, необхідно скористатися критеріями ефективності.

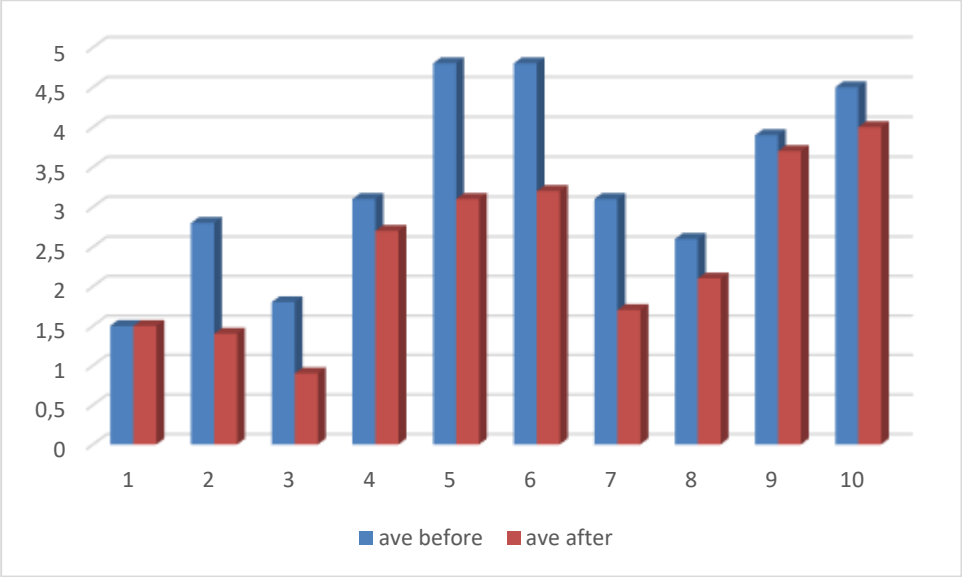


Рис. 3.28. Середній час роботи інтеграційних операцій

Висунемо основну (нульову) гіпотезу H0 і перевіримо, чи не суперечить вона наявним емпіричним даним. Конкуруючою (альтернативною) гіпотезою призначимо H1 яка суперечить нульовій.

Розглянемо використання t -критерію Стьюдента визначення наявності відмінностей між двома вибірками. При цьому вибірки можуть бути:

- незалежними, незв'язними з різним числом значень у вибірках;
- залежними, пов'язаними з рівним числом значень у вибірках .

Критерій t Стьюдента спрямований на оцінку відмінностей величин середніх x та y двох вибірок X та Y, які розподілені за нормальним законом. Однією з головних переваг критерію є широта його застосування.

Як правило, у дослідженнях найчастіше застосовуються такі параметричні критерії як: t - критерій Стьюдента, що дозволяє оцінювати відмінності середніх у двох вибірках і F – критерій Фішера, який оцінює різницю між двома дисперсіями. Вони дозволяють отримати найбільш наочне уявлення аналізованих сукупностей. Обчислення значення $t_{\text{емп}}$ здійснюється за формулою:

$$t_{\text{емп}} = \frac{d}{Sd}$$

де $di = xi - yi$- різниці між відповідними значеннями змінної X та змінної Y, d середнє цих різниць, n - обсяг вибірки, границі від i = 0 до n.

У свою чергу Sd обчислюється за такою формулою:

$$S_d = \sqrt{\frac{\sum d_i^2 - \frac{\left(\sum d_i\right)^2}{n}}{n(n-1)}}.(4.3)$$

Число ступенів свободи визначається за формулою k = n – 1.

Для застосування t -критерію Стьюдента необхідно дотримуватися таких умов:

1.Вимірювання може бути проведено у шкалі інтервалів та співвідношень.

2.Вибірки які порівнюються мають бути розподілені за нормальним законом.

Для розрахунку t -критерію Стьюдента необхідно скористатися процедурою «Парний двовибірковий t -тест для середніх» із пакета аналізу програмного продукту Microsoft Excel (табл. 3.1).

Таблиця 3.1

Результати розрахунку

	Змінна 1	Змінна 2
Mean	3,27	2,22
Variance	1,626777778	1,166222222
Observations	10	10
Кореляція Пірсона	0,817653538	
Hypothesized Mean Difference	0	
df	9	
t Stat	4,516158041	
P(T<=t) one-tail	0,000727396	
t Critical one-tail	1,833112933	
P(T<=t) two-tail	0,001454792	
t Critical two-tail	2,262157163	

Нульова гіпотеза H_0:μ_1-μ_2=δ приймається, якщо |t| < $t_{кр1}$ (інакше відкидається); гіпотеза H0 при конкуруючій гіпотезі H1:μ_1>μ_2+δ приймається, якщо t < $t_{кр2}$; при конкуруючій гіпотезі H_1: μ_1<μ_2+5 нульова гіпотеза приймається під час виконання

нерівності $t_{кр2}<t$.

За допомогою процедури «Парний двовибірковий t -тест для середніх» перевіримо на рівні значущості α = 0.05 гіпотезу Н1 : середні часові інтервали процесів обміну даними між системами із застосуванням запропонованої методики комплексного тестування суттєво зменшилися. Основна гіпотеза H_0: середній час процесів обміну даними між системами без застосування запропонованої методики та з її застосуванням суттєво не змінилося.

Аналіз результатів рішення показав, що розрахункове значення t=4,5 статистики Т знаходиться в області (2,26; +∞). Це означає, що гіпотеза про рівність часу роботи інтеграційних операцій без застосування запропонованої методики та з її застосуванням суперечить фактичним даним спостереження і, отже, її треба відхилити (на рівні значущості α=0.05) та прийняти альтернативну гіпотезу H_1 що передбачає, що середній час процесів обміну даними між модулями КІС без застосування запропонованої методики тестування більше часу роботи після застосування.

Таким чином, початкове припущення підтвердилося, дійсно середній час роботи інтеграційних операцій, після впровадження сценарію комплексного тестування КІС дає кращі результати в порівнянні з попередніми.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Усі дослідження методів тестування комп'ютерної інформаційної системи проводились з дотриманням правил та норм охорони праці і вимог техніки безпеки.

Сервери які використовуються в сучасних комп'ютерних системах використовують різного роду системи охолодження, а також робота інших системних компонентів є потенційними джерелами цілого ряду звуків, що містять як коливання, які можна почути, так і коливання ультразвукового діапазону. Цей шум справляє негативний вплив на функціональний стан користувачів. Вимірювання шуму на робочих місцях і санітарні норми рівня шуму визначені в ДСТУ 2867-94.

Так як не завжди є можливість виділити для розміщення серверів окреме приміщення їх встановлюють в спеціальних комутаційних шафах. Однак даний конструктив незважаючи на всі свої позитивні особливості не обмежує повністю шумове навантаження від роботи системи охолодження і інших механічних пристроїв серверів.

Висококваліфікована розумова робота, що вимагає зосередженості, може проводитись у приміщеннях, де рівень шуму не перевищує 55 дБ. Сумарний вплив численних джерел шуму у приміщенні у результаті багаторазового відбиття звукових хвиль може значно перевищити енергію прямого звука від тих же джерел. Шум від окремих приладів не повинен перевищувати фоновий більше ніж на 5 дБ.

Для боротьби з шумом застосовують так звані «тихі» системи охолодження зокрема і водяне охолодження, або спеціальні великогабаритні вентилятори. Крім того комутаційні шафи частково поглинають шуми і вібрації. Тому в приміщенні де виконувалась наукова робота рівень шуму був в районі 40-45 дБ, тобто по рівню шумового навантаження приміщення відповідає вимогам охорони праці.

Також при роботі за ЕОМ необхідно особливу увагу звертати на правильне освітлення. Неправильне освітлення (пряма та відбита від екранів близькість, вуалюючі відбиття, несприятливий розподіл яскравості в полі зору, невірна орієнтація робочого місця відносно світлових отворів) призводить до негативних фізіологічних впливів на користувачів ЕОМ. Погана якість символів, що представлені на екрані, також може викликати зоровий дискомфорт, бути стресовим фактором та ін.

Освітлення повинно відповідати нормальним рівням за ДБН В.2.5-28:2018 Природне і штучне освітлення.

Вимоги до освітлення для візуального сприймання користувачами інформації з двох різних носіїв (з екрана ЕОМ та паперового носія) різні. Надто низький рівень освітленості погіршує сприймання інформації при читанні документів, а надто високий призводить до зменшення контрасту зображення знаків на екрані. Відношення яскравості екрана ЕОМ до яскравості оточуючих його поверхонь не перевищує у робочій зоні 3:1.

Наближено можна вважати, що при 10%-ному зменшенні освітленості працездатність знижується на 1%. Коли за характером роботи вимагається комбінація цих двох носіїв інформації, освітленість можна варіювати від 300 до 700 лк, причому чим рідшою є зміна полів зору в процесі роботи (з екрана на документ та навпаки), тим вищим може бути рівень освітленості. 300-500 лк — оптимальна освітленість робочих приміщень для роботи з ЕОМ. Стрибки яскравості при зміні полів зору мають бути мінімальними, тобто інтенсивність освітлення поверхні, де знаходяться рукописи та документи, не повинна перевищувати яскравості екрана дисплея.

Приміщення в якому виконувалась кваліфікаційна робота забезпечене природнім і штучним освітленням. При роботі за ЕОМ обрано місце, щоб в поле зору не потрапляли вікна або освітлювальні прилади. Крім того шкідливо коли вікна знаходяться за спиною оскільки на моніторі з’являється відбиття світла. Завдяки наявним жалюзі можна регулювати світловий потік і захистити робоче місце від

попадання прямих сонячних променів. Адже вікна приміщення орієнтовані на південний схід.

Штучне освітлення у приміщені реалізовано у вигляді комбінованої системи освітлення з використанням люмінесцентних джерел світла у світильниках загального освітлення, які слід розташовані над робочими поверхнями у рівномірно-прямокутному порядку. Для запобігання засвітленню екранів ЕОМ прямими світловими потоками лінії світильників розташовані з достатнім бічним зміщенням відносно робочих місць, а також паралельно до вікон.

На робочому місці забезпечена рівномірна освітленість за допомогою переважно відбитого або розсіяного світлорозподілу. Світлових відблисків з клавіатури, екрана та від інших частин ЕОМ у напрямку очей користувача немає. Дискомфорт від відбиття світла знижується при збільшенні яскравості екрана та зниженні рівня навколишнього освітлення.

Щоб уникнути пульсацій освітленості використовують якісні LED лампи.

Інформація, яку одержує користувач, генерується на екрані, а комфортність її сприймання залежить від чіткості символів. На робочих місцях користувачів використовуються сучасні ноутбуки та ПК обладнані рідкокристалічними моніторами. Завдяки використанню IPS матриць з частотою оновлення не менше 60 Гц очі практично не втомлюються. Матове покриття екрану виключає відблиск сторонніх джерел світла. Роздільна здатність моніторів що використовуються на робочих місцях 1920×1080 точок що забезпечує високу чіткість зображення. Ще одною важливою перевагою даних моніторів відсутнє опромінення користувача.

Отже в даному підрозділі розглянуто вплив середовища на працездатність та здоров’я користувачів комп’ютерів. Як висновок можна сказати, що робоче місце яке використовувалось для написання даного наукового дослідження відповідає вимогам з охорони праці.

Однак необхідно не забувати що надмірна робота з ПК може привезти до порушення роботи організму користувача. Тому необхідно дотримуватись вимог щодо планування робочого часу за ЕОМ і робити перерви. Це дозволить підвищити продуктивність роботи і зменшить втомленість організму.

4.2. Підвищення стійкості роботи об'єктів господарської діяльності у воєнний час

Під стійкістю об’єкту роботи об'єктів господарської діяльності країни в цілому розуміється його здатність в умовах воєнного часу забезпечувати всі галузі необхідною продукцією. Здатність об’єкта народного господарства випускати продукцію залежить від захисту і нормального функціонування чотирьох основних елементів сучасного виробництва, якими є:

– виробничий персонал (робітники та службовці);

– будинки і споруди з технологічним устаткуванням;

– система постачання енергією, водою, паливом, устаткуванням і ремонтною базою;

– система виробничих і кооперативних зв’язків з іншими об’єктами.

Тому стійкість роботи об’єктів і галузі народного господарства в цілому в умовах воєнного стану визначається наступними факторами:

– надійністю захисту робітників та службовців від усіх вражаючих факторів зброї масового ураження;

– здатністю інженерно-технічного комплексу (ІТК) об’єкта протистояти вражаючим факторам ядерного вибуху;

– надійністю системи постачання об’єкта всім необхідним для виробництва продукції (сировиною, паливом, що комплектують виробами, електроенергією, водою, газом тощо.);

– захищеності об’єкта від вторинних вражаючих факторів (пожеж, вибухів, затоплень, зараження місцевості отруйними і сильнодіючими отруйними речовинами);

– стійкістю і безперервністю керування виробництвом і цивільною обороною;

– підготовленість об’єкта до проведення рятувальних та інших невідкладних робіт і робіт з відновленням порушеного виробництва.

Перераховані фактори визначають собою й основні, загальні для всіх об’єктів господарювання, шляхи підвищення стійкості роботи в умовах військового стану, а саме:

– забезпечення надійного захисту робітників та службовців від вражаючих факторів зброї масового ураження;

– захист основних виробничих фондів від вражаючих факторів, у тому числі й від вторинних;

– підвищення надійності й оперативності керування виробництвом;

– забезпечення стійкості постачання всім необхідним для випуску запланованої на час надзвичайних ситуацій продукцією;

– підготовка до відновлення порушеного виробництва.

Захист робітників та службовців в умовах НС воєнного часу. Це найголовніша задача по підвищенню стійкості роботи об’єкта господарювання. Робітники й службовці – головна продуктивна сила і тому стійкість економіки визначається, насамперед, здатністю захистити і зберегти цю силу.

Військові конфлікти супроводжуються руйнуванням будинків, споруджень і знищенням основної продуктивної сили – працюючого населення. Тому серед усіх задач по підвищенню стійкості роботи об’єктів народного господарства основною є задача завчасного вживання заходів по забезпеченню захисту робітників та службовців і членів їхніх родин. Захист робітників та службовців від зброї масової поразки в сучасних умовах здійснюється трьома основними способами:

– укриття людей у захисних спорудженнях (сховищах, протирадіаційних укриттях);

– проведення евакуації робітників, службовців і членів їхніх родин;

– використання засобів індивідуального захисту, а також проведенням заходів щодо протирадіаційного, протихімічного і протибактеріологічного захисту з урахуванням конкретних обставин.

Варто також підкреслити, що найважливішою умовою успішного вирішення задачі захисту людей є навчання їх правилам дії по сигналах оповіщення цивільного

захисту, застосуванню способів і засобів захисту, наданню самодопомоги і взаємодопомоги, діям у складі формувань ЦЗ.

Захист засобів виробництва. полягає в підвищенні фізичної опірності будинків, споруджень і конструкцій об’єкта до впливу вражаючих факторів ядерного вибуху, захисту технологічного і верстатного устаткування, засобів зв’язку й інших засобів, що складають матеріальну основу виробничого процесу.

Основу діяльності керівника виробництва – начальника ЦЗ, а також його штабу складає якісне та професійне керування підлеглими йому структурами в організації їхньої дії і напрямку зусиль на своєчасне й успішне виконання виробничих завдань. Тому, забезпечення надійності й оперативності керування є важливою ланкою в підвищенні стійкості роботи об’єкта, в умовах швидко мінливої обстановки воєнного часу.

Забезпечення стійкого постачання підприємств. Для виробництва продукції необхідні: електроенергія, вода, паливо, сировина, матеріали й інші матеріально-технічні засоби. Забезпечення підприємств цими ресурсами багато в чому визначає можливість нормального їхнього функціонування в умовах воєнного часу. Це досягається проведенням таких заходів, що сприяють підвищенню не ураженості комунально-енергетичних мереж, транспортних комунікацій і джерел постачання, надійному захисту необхідних запасів палива, сировини, напівфабрикатів, що комплектують, виробів тощо.

Можливості вражаючою дії сучасних видів зброї такі, що забезпечити абсолютний захист від нього об’єктів і споруд практично неможливо. Вони можуть одержати той чи інший ступінь руйнування. У цих умовах задача зводиться до того, щоб у випадку слабких і середніх руйнувань на об’єкті відбудувати об’єкт і відновити випуск необхідної продукції в мінімальний термін.

Таким чином в даному розділі були розглянуті особливості підвищення стійкості об’єктів господарської діяльності в умовах воєнного стану.

ВИСНОВКИ

У ході проведеного дослідження було розглянуто найбільш популярні методи тестування, які застосовуються для проведення комплексного тестування комп’ютерних інформаційних систем. Показано, що комплексне тестування КІС відіграє важливу роль у процесі життєвого циклу ІС для визначення її працездатності та надійності.

Під час проведення досліджень у роботі отримано такі теоретичні та прикладні результати.

1. У ході аналізу теоретичних положень життєвого циклу тестування КІС було отримано концепції, необхідні визначення поняття комплексного тестування та виділення його основних видів.

2. Для розробки тестів докладно розглянуто методи функціональних діаграм та попарного тестування. Показано переваги проведення аналізу предметної галузі на основі онтологічної моделі.

3. На основі порівняльного аналізу методів тестування зроблено висновки, що для якіснішого проведення комплектного тестування КІС необхідне використання сукупності розглянутих методів. Для запропонованих методів визначено основні концепції та етапи проведення тестування.

4. В рамках навантажувального тестування розглянуто підхід, заснований на моделях. Показано його ефективне застосування для тестування прикладного програмного забезпечення в ІТ- середовищі функціонування, що включає СУБД, бази даних, системне ПЗ, розгорнуті на обладнанні комплексу технічних засобів ІС.

5. Докладно досліджено технологію навантажувального тестування з використанням моделей, сформовано вимоги до проведення навантажувального експерименту та властивості об'єкта тестування, що охоплюють усі аспекти планування, проведення навантажувального експерименту та аналізу його результатів. Показано, що використання моделей суттєво знижує трудомісткість навантажувального тестування та забезпечує можливість повторного використання підготовленого експерименту.

6. Розглянуто низку інструментальних засобів, що надають можливості для реалізації автоматизованого тестування. Кожен із розглянутих інструментів було оцінено з погляду його впровадження у існуючий процес розробки, і навіть з погляду задоволення вимог КІС. Наведено результати аналізу щодо основних із запропонованих інструментів. Показано, що автоматизація процесу тестування допомагає компаніям скорочувати час, що витрачається на тестування, а також спрощувати весь процес, оскільки застосовуються спеціальні програмні інструменти для створення та запуску тестів та перевірки результатів їх виконання.

7.Таким чином, запропоновану методику (сценарій) комплексного тестування КІС можна використовувати для оцінки надійності та ефективності ІС, що функціонує у бізнес-середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elfriede Dustin Automated Software Testing: Introduction, Management, and Performance: Introduction, Management, and Performance. Addison-Wesley Professional, 2008. 608 p.

2. What is automated testing? URL: https://www.functionize.com/automated-testing (дата звернення: 05.11.2022).

3. Cem Kaner Testing Computer Software / Wiley; 2nd edition, 2001. 480 p.

4. Слюз І., Жаровський Р. Принципи та основні етапи комплексного тестування комп'ютерної інформаційної системи. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. с. 93.

5. Слюз І., Жаровський Р. Критерії ефективності тестування комп'ютерної інформаційної системи. Матеріали XI Міжнародна науково-технічна конференція молодих учених та студентів «Актуальні задачі сучасних технологій» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. с. 174.

6. Challenges of Large-Scale Software Testing and the Role of Quality Characteristics. URL: https://www.diva-portal.org/smash/get/diva2:1421638/FULLTEXT01.pdf (дата звернення: 10.12.2022).

7. Сучасні підходи до інтеграційного та навантажувального тестування на базі Spring. URL: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/18914/Nikitchenko_Bakalavrska_robota.pdf?sequence=1 (дата звернення: 10.12.2022).

8. Золотухіна О.А., Негоденко О.В., Резник С.Ю., Разіна С.Я. Якість та тестування інформаційних систем. URL: https://dut.edu.ua/uploads/l_2177_57414302.pdf (дата звернення: 14.12.2022).

9. Тестування програмного забезпечення. URL: http://eprints.cdu.edu.ua/1482/1/testyvan.pdf (дата звернення: 15.12.2022).

10. Рівні тестування. URL: https://qalearning.com.ua/theory/lectures/material/testing-levels/ (дата звернення: 10.12.2022).

11. Andrews A., Offut J., Alexander R. Testing Web Applications by Modeling with FSMs.: National Science Foundation, 2005. 28 p.

12. Automated Software Testing Isn't Automatic: Introduction to the Basics. URL: https://www.smartsheet.com/automation-testing-software (дата звернення: 15.12.2022).

13. What is server testing, and why should you do it. URL: https://www.websitepulse.com/blog/what-is-server-testing (дата звернення: 10.12.2022).

14. Changyou Xing, Guomin Zhang, Ming Chen. Research on Universal network performance testing model/ International Symposium on Communications and Information Technologies, 2007. p. 780-784.

15. Heiskanen H., Maunumaa M., Katara, M. Test Process Improvement for Automated Test Generation. Tampere: Tampere University of Technology, Department of Software Systems, 2010.

16. Jie M., Honlin Zh., Wenbo X., Jin L., Reliability Testing Methods for Critical Information System URL: http://www.ipcsit.com/vol16/6- ICICM2011M009.pdf (дата звернення: 10.12.2022).

17. Khatko E, Fillipov V. Mobile applications testing processes metrics and optimization criteria. Software Engineering. 5, 2012, T. 2.

18. Kim, G.-B., A Method of Generating Massive Virtual Clients and Model-Based Performance Test, Proc. of the Fifth Int. Conf. on Quality Software, 2005, pp. 250–254.

19. Makinen M. Model Based Approach to Software. Helsinki: Helsinki University of Technology, 2007. 69 p.

20. Robinson H. Graph Theory Techniques в Model-Based Testing. 1999. 10 p.

21. Xijiang L., Pomeranz I., Sudhakar M. Techniques for Improving Efficiency of Sequential Circuit Test Generation. Iowa: University of Iowa. 2015. 5 p.

Додаток А.

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.416.2
І. Слюз, Р. Жаровський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРИНЦИПИ ТА ОСНОВНІ ЕТАПИ КОМПЛЕКСНОГО ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

UDC 004.416.2
I. Slyuz, R. Zharovskyi

PRINCIPLES AND MAIN STAGES OF COMPLEX TESTING OF A COMPUTER INFORMATION SYSTEM

Тестування комп'ютерної інформаційної системи охоплює основні стадії життєвого циклу, аналогічний до послідовності процесів розробки програмного забезпечення: постановка задачі для тесту, проектування, написання тестів, тестування тестів, виконання тестів та вивчення результатів тестування

Якщо врахувати, що комп'ютерна інформаційна система - це не тільки програмні компоненти, що використовуються в її складі, але й апаратне та організаційне забезпечення, то і в результатах її випробувань повинні бути відображені показники роботи серверів, робочих станцій, мережевого обладнання (їх надійність та продуктивність), а також ефективність розробленого регламенту експлуатації системи. У зв'язку з цим виникає необхідність у проведенні комплексного тестування на відповідність усім вимогам, що висуваються.

Комплексне тестування комп'ютерної інформаційної системи складається з наступних етапів:

1. Детальне вивчення проекту системи та її експлуатаційних документів.
2. Створення та впровадження автоматизованої системи відстеження помилок.
3. Безпосереднє тестування роботи інформаційної системи, яке включає:
- тестування роботи всіх модулів інформаційної системи,
- перевірка внутрішньосистемних зв'язків,
- тестування обладнання, у тому числі на наявність необхідних ліцензій,
- перевірка програмного забезпечення, у тому числі перевірка коду,
- тестування продуктивності та максимальних навантажень інформаційної системи,
- тестування на відмови: несподівані програмні збої, вихід з ладу модулів системи, людський фактор та ін.
- тестування на захищеність інформаційної системи – включає комплекс досліджень з виявлення способів злому системи і витоків інформації,
- загальна перевірка роботи системи.
4. Тестування роботи системи управління ІТ-структурою.
5. Тестування персоналу.
6. Аналіз отриманих даних та вироблення стратегії з виправлення виявлених помилок.

Комплексне тестування не призначене для тестування всіх функцій повністю зібраної системи. Воно спрямовано на пошуку невідповідності системи її вихідним цілям [1]. Таким чином, у комплексному тестуванні бере участь комп'ютерна інформаційна система, опис її цілей та вся документація, яка поставлятиметься разом із системою.

Література

1. Changyou Xing, Guomin Zhang, Ming Chen. Research on Universal network performance testing model. International Symposium on Communications and Information Technologies, 2007. P. 780–784.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XI Міжнародної науково-практичної конференції молодих учених та студентів
7-8 грудня 2022 року

УКРАЇНА
ТЕРНОПІЛЬ – 2022

УДК 004.416.2

І. Слюз, Р. Жаровський, к.т.н.
Тернопільський національний технічний університет імені Івана Пулюя

КРИТЕРІЇ ЕФЕКТИВНОСТІ ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

I. Slyuz, R. Zharovskyi, Ph.D.

COMPUTER INFORMATION SYSTEM TESTING EFFICIENCY CRITERIA

Під тестуванням розуміють процес виконання функцій інформаційної системи на кінцевій множині вхідних даних X, отримання відгуку Y та його порівняння з еталонною множиною вихідних значень Y_{em} з метою виявлення помилок і дефектів в ІС.

Пара (x, y_{em}), $x \in X, y_{em} \in Y_{em}$ називається тестовим випадком (тест-кейс), а всі тестові випадки, згруповані за певною ознакою, іменуються тестовим комплектом. Рішення про наявність помилки у функціонуванні КІС приймається або при розбіжності результатів на одному з тестових випадків, або якщо відрізняються закони розподілу вихідних даних.

В обох випадках вважається, що тестування пройшло успішно, оскільки виявлено як мінімум одну помилку.

Було помічено, що у міру виявлення найбільш серйозних помилок і недоліків, ефективність низьковитратних методів падає разом з кількістю помилок, що виявляються. Тому всі методи тестування в межах своїх задач мають більшу ефективність в порівнянні з іншими. Можна виділити вимоги до ідеального критерію тестування: достатній, повний, надійний, легко перевіряється.

При цьому існують такі класи критеріїв:

1. Структурні критерії, що використовують модель програми у вигляді «білої скриньки», що передбачає знання вихідного тексту програми чи специфікації програми як потокового графа управління. До них відносяться: тестування команд (критерій C0); тестування гілок (критерій C1); тестування шляхів (критерій C2).
2. Функціональні критерії, що використовують модель «чорної скриньки», що передбачає формулювання в описі вимог до програмного виробу та забезпечення контролю ступеня виконання вимог замовника у програмному продукті. До них відносяться: тестування пунктів специфікації; тестування класів вхідних даних; тестування правил; тестування класів вихідних даних; тестування функцій; комбіновані критерії для програм та специфікацій.
3. Критерії стохастичного тестування, які формулюються в термінах перевірки наявності заданих властивостей у програми, що тестується, коли набір детермінованих тестів (X, Y) має величезну потужність. Критерії стохастичного тестування: статистичні методи закінчення тестування - стохастичні методи прийняття рішень про збіг гіпотез про розподіл випадкових величин; Метод оцінки швидкості виявлення помилок - заснований на моделі швидкості виявлення помилок, згідно з якою тестування припиняється, якщо оцінений інтервал часу між поточною помилкою та наступною занадто великий для фази тестування програми.
4. Мутаційні критерії, орієнтовані на перевірку властивостей програмного виробу на основі підходу Монте-Карло, що дозволяє на основі дрібних помилок оцінити загальну кількість помилок, що залишилися у програмі.

Для розробки тестів слід вибирати методи, які охоплюють різні критерії і дозволяють забезпечити повноту результатів тестування компонентів комп'ютерної інформаційної системи

Додаток Б.

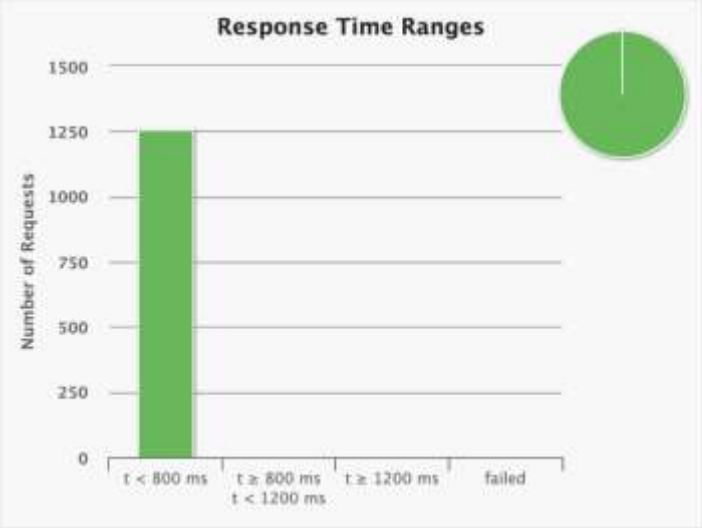
Порівняльний аналіз методів тестування продуктивності

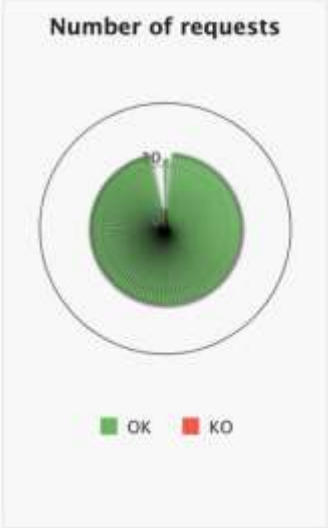
	Тестування продуктивності	Навантажувальне тестування	Стрес-тестування
Домен	Суперсет навантажувального та стрес-тестування	Підмножина тестування продуктивність.	Підмножина тестування продуктивність.
Об’єм	Дуже широка сфера застосування. Включає навантаження тестування, стрес-тестування, тестування ємності, об'ємне тестування, тестування на витривалість, тестування піків, тестування на масштабованість та тестування надійності та ін.	Вужче охоплення в порівнянні з тестування продуктивності. Включає об'ємні випробування та випробування на витривалість.	Вужче охоплення в порівнянні з тестування продуктивності. Включає тестування на вбирання та тестування шпильок.
Основна мета	Щоб встановити орієнтир та стандарти для програми.	Щоб визначити верхня межа системи, встановіть SLA додатки та подивіться, як система обробляє томи з великою навантаженням.	Визначити, як система веде себе при інтенсивних навантаженнях та як відновлюється після збою. В основному, щоб підготувати ваш додаток до несподіваному стрибку трафіку.
Межа наванта ження	Обидва - нижчі і вищі за поріг перерви.	До порога розриву	Вище порога розриву
Атрибути вивчення	Використання ресурсів, надійність, масштабованість, використання ресурсів, час відгуку, пропускна здатність, швидкість і т.д.	Пікова продуктивність, пропускна здатність сервера, час відгуку при різних рівнях навантаження (нижче порога перерви), адекватність робочої середовища, кількість користувацьких додатків, вимоги до балансування навантаження і т.д.	Стабільність за межами пропускної спроможності, час відгуку **(** вище порога розриву) тощо.

Проблеми, виявлені за допомогою цього типу тестування	Усі помилки продуктивності, включаючи роздування під час виконання, можливості оптимізації, проблеми, пов'язані зі швидкістю, затримкою, пропускною спроможністю тощо.	Проблеми з балансуванням навантаження, проблеми з пропускною спроможністю, проблеми з пропускною спроможністю системи, поганий час відгуку, проблеми з пропускною спроможністю і т.д.	Прогалини в безпеці з перевантаження, проблеми з пошкодженням даних у ситуації перевантаження, повільність, витоку пам'яті і т.д.

Додаток В.

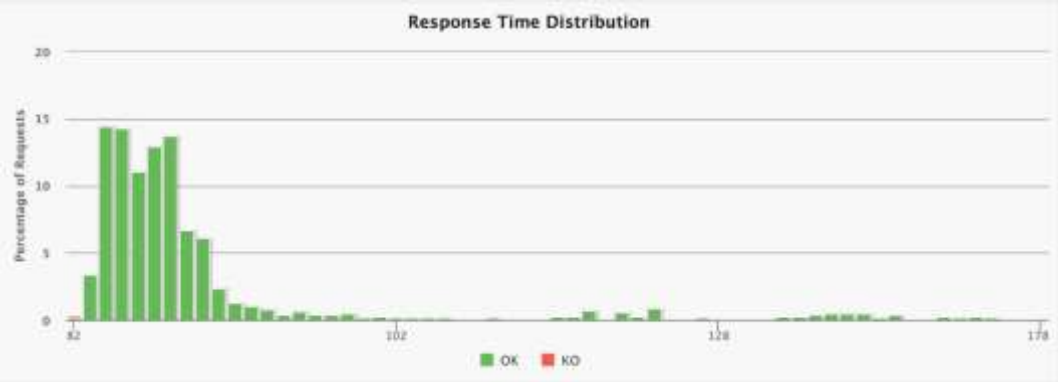
Звіт навантажувального тестування з Gatling

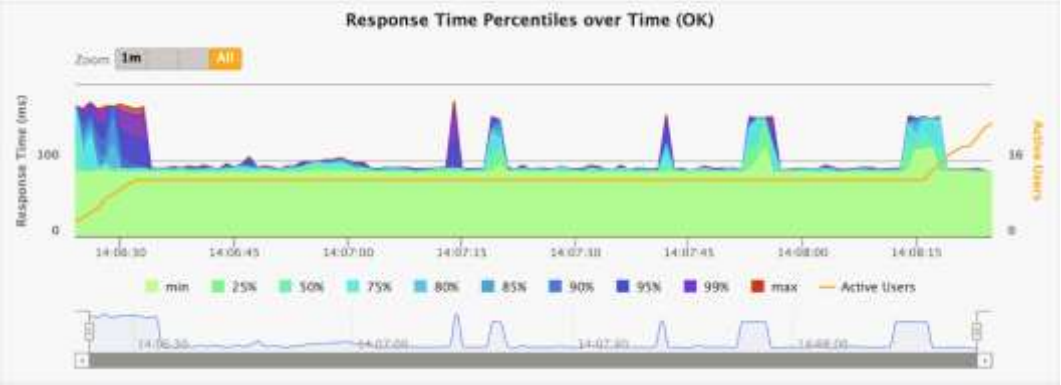


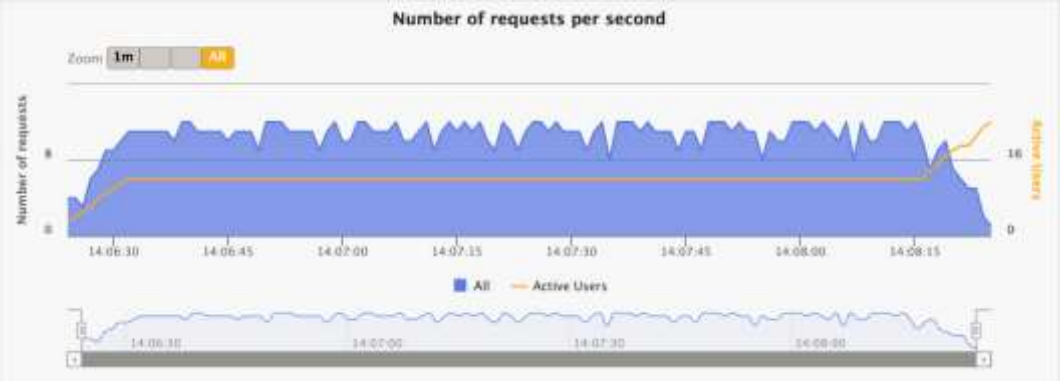


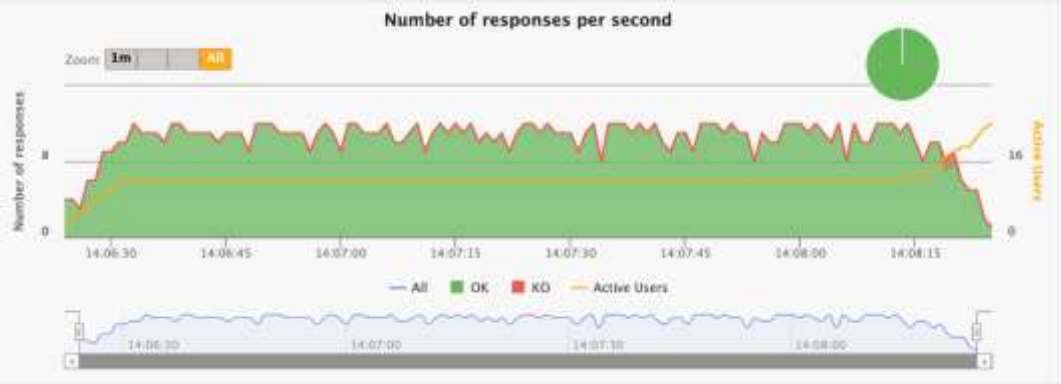
Stats Fixed height Full size Expand all groups Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	1257	1255	2	0%	10.303	82	87	89	120	164	178	91	16
Search	12	12	0	0%	0.098	422	435	437	439	441	441	433	5
SubGroup	12	12	0	0%	0.098	252	259	261	266	267	267	259	4
Search	12	12	0	0%	0.098	84	86	88	91	92	92	87	2
Select	12	12	0	0%	0.098	83	88	89	90	91	91	87	2
Browse	12	12	0	0%	0.098	8935	9059	9128	9204	9228	9234	9067	91
Edit	2	1	1	50%	0.016	253	420	503	569	583	586	420	167




Response Time Percentiles over Time (OK)
Zoom
1m
All
Response Time (ms)
100
0
16
0
Active Users
14:06:30
14:06:45
14:07:00
14:07:15
14:07:30
14:07:45
14:08:00
14:08:15
min
25%
50%
75%
80%
85%
90%
95%
99%
max
Active Users


Number of requests per second
Zoom
1m
All
Number of requests
16
0
Active Users
14:06:30
14:06:45
14:07:00
14:07:15
14:07:30
14:07:45
14:08:00
14:08:15
All
Active Users


Number of responses per second
Zoom
1m
All
Number of responses
16
0
Active Users
14:06:30
14:06:45
14:07:00
14:07:15
14:07:30
14:07:45
14:08:00
14:08:15
All
OK
KO
Active Users

Додаток Г.

Порівняльний аналіз інструментальних засобів навантажувального тестування

Особливість	The Grinder	Gatling	Tsung	JMeter	Locust
Операційні системи	будь-яка	будь-яка	Linux / Unix	будь-яка	будь-яка
Графічний інтерфейс користувача	Тільки консоль	Тільки рекордер	ні	Повний	ні
Тест рекордер	TCP (включаючи HTTP)	HTTP	HTTP, Postgres	HTTP	ні
Тест Мова	Python, Clojure	Scala	XML	XML	Python
Мова Розширення	Python, Clojure	Scala	Erlang	Java, Beanshell, Javascript, Jexl	Python
Звіти	Console	HTML	HTML	CSV, XML, вбудовані таблиці, графіки, плагіни	HTML
Протоколи	HTTP SOAP JDBC POP3 SMTP LDAP JMS	HTTP JDBC JMS	HTTP WebDAV Postgres MySQL XMPP WebSocket AMQP MQTT LDAP	HTTP FTP JDBC SOAP LDAP TCP JMS SMTP POP3 IMAP	HTTP
Моніторинг хоста	ні	ні	так	Так, з плагіном PerfMon	ні
Обмеження	Знання Python потрібне для розробки та редагування тестів. Звіти дуже прості та короткі.	Обмежена підтримка протоколів. Потрібне знання мови DSL на основі Scala. Не масштабується.	Протестована і підтримується тільки в системах Linux.	Пов'язані звіти не легко інтерпретувати.	Знання Python потрібне для розробки та редагування тестів.

Додаток Д.

Технічне забезпечення КІС

Позиція	Позначення	Найменування та технічна характеристика
1	Маршрутизатор	Asus RT-AC68U
2	Сервер	Xeon E5-2630, 12 x 2.60 ГГц, процесорів - 2, оперативна пам'ять – 8 ГБ, пам'ять – 2 x 500 ГБ
3	Зовнішнє сховище даних	BLOB у SharePoint Foundation
4	Джерело безперебійного живлення	Smart-UPS RT 3000VA RM 230V
5	Робочі станції	Dell Precision 7510 (4K IGZO) (Precision 7000 Серія) Процесор: Intel Xeon E3-1535M v5 2.9 GHz Графічний адаптер: NVIDIA Quadro M2000M, Ядро: 1038 - 1197 MHz МГц, 10.18.13.5445 - nVIDIA ForceWare 354.45, yes Оперативна пам'ять: 32768 Мбайт , DDR4-2133 Дисплей: 15,6 дюйм. 16:9, 3840x2160 пікс. 282 точок/дюйм, Sharp LQ156D1, IGZO IPS, глянсове покриття: немає Материнська плата: Intel Sunrise Point CM236 Зберігання даних: Samsung SSD SM951a 512GB M.2 PCIe 3.0 x4 NVMe (MZVKV512), 512 Гбайт
6	Роутер	TP-LINK TL-WA901ND
7	Лазерні багатофункціональні пристрої	Canon i-SENSYS MF3010
8	Сервер додатків та БД	Intel Xeon 3,2Mhz; 8Гб ОЗУ; 500Гб HDD; 512Мб відео

Додаток Е.

Статистика виконання сценарію

Транзакція	**Мінімум (сек)**	**Середнє (сек)**	**Максимум (сек)**
01. Ввести невірний логін та пароль та перевірити аут	0,012	0,013	0,017
02. Увійти до системи під паролем	0,70	4,01	63,0
03. Відкрити розділ «Введення даних»	0,10	2,89	32,1
04. Відкрити Паспорт 1	0,12	0,17	1,43
05. Відкрити Паспорт 2.	0,11	0,15	0,71
06. Перейти до таблиці 1	0,054	0,37	25,4
07. Ввести 50 значень у таблицю та Зберегти	0,080	1,19	32,4
08. Перейти до рівня ЗМ	0,087	0,54	8,94
09. Перейти до таблиці 4	0,12	0,36	19,6
10. Ввести 50 значень у таблицю та Зберегти	0,12	0,43	27,9
11. Перейти до розділу Друковані форми	0,15	0,43	23,9
12. Вибрати програму та версію	0,16	0,21	0,28
13. Вибрати таблицю 1 та Сформувати	5,84	6,33	7,29
14. Вибрати таблицю 2 та Сформувати	1,05	1,28	1,72
15. Вибрати таблицю 3 та Сформувати	9,67	14,2	43,9
16. Вийти із системи	0,028	0,057	0,13
17. Увійти до системи під admin	1,07	3,6	33,7
18. Відкрити вкладку «Моніторинг»	0,15	14,1	32,4
19. Відкрити розділ «Звед. Звіт»	0,098	7,13	30,7
20. Вибрати 3 квартал 2022 року	2,05	3,81	20,7
21. Сформувати пояснювальну записку	0,12	0,14	0,22
22. Відкрити розділ «Зведений звіт»	2,07	2,28	2,61
23. Вийти із системи	0,033	0,043	0,070