МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ І.С. Скарга-Бандурова
«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Методи автоматизації аналізу тестування веб-додатків

Освітній рівень"бакалавр"
Спеціальність 122 "Комп'ютерн і науки"

Науковий керівник роботи:	_____ (підпис)	О.І.Рязанцев (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	О.Є.Волчанський (ініціали, прізвище)
Група:		КН-16бз

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній рівень бакалавр
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри _____
_____ С.О. Сафонова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Волчанського Олександра Євгеновича
(прізвище, ім'я, по батькові)

1. Тема роботи Методи автоматизації аналізу тестування веб-додатків

керівник проекту (роботи) Рязанцев Олександр Іванович, д.т.н., проф.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «30» 04 2020 р. № 77/15.15
2. Строк подання студентом роботи 10.06.2020
3. Вихідні дані до роботи Матеріали переддипломної практики, теорія пошуку досяжності на графі, найбільш популярні сайти

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз видів тестування, математична постановка завдання, методи та алгоритми, програмна реалізація, аналіз результатів, охорона праці, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 30.04.2020

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка технічного завдання	01.05.2020-07.05.2020	
2	Аналіз літературних джерел	08.05.2020-13.05.2020	
3	Розробка частини проекту "Охорона праці"	13.05.2020-15.05.2020	
4	Проведення навантажувального тестування	15.05.2020-01.06.2020	
5	Оформлення пояснювальної записки та презентації	2.06.2020-09.06.2020	

Здобувач вищої освіти _____ О.Є.Волчанський
(підпис) (прізвище та ініціали)

Науковий керівник _____ О.І.Рязанцев
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 79с., 15 рис., 8 табл., 1 додаток, 31 бібліографічне джерело посилань.

Дана робота присвячена аналізу тестування веб-додатків. Під час роботи над проектом був використаний алгоритм оцінки досяжності на графі. Саме за допомогою нього була проаналізована продуктивність сайту під час роботи з ним великої кількості користувачів. А також були математично вираховувані значення, що характеризували різні критерії якості сайта.

Проведені дослідження, які показали, що методи, використані під час аналізу даної роботи є досить точними для того, щоб у повній мірі мати змогу отримати відповідні дані при роботі з веб-застосунками.

Ключові слова: тестування, продуктивність, навантаження, алгоритм оцінки досяжності на графі, веб-застосунок, інструменти тестування

ЗМІСТ

ВСТУП

Існує кілька визначень для такого терміна, як якість програмного забезпечення (ПЗ). Перше звучить, як ступінь, в якій програмне забезпечення володіє необхідною комбінацією властивостей. Друге ж - як сукупність характеристик програмного забезпечення, що відносяться до його здатності задовольняти встановлені і передбачувані потреби.

Забезпеченням даної якості називається сукупність заходів, що охоплюють всі технологічні етапи розробки, випуску та експлуатації ПЗ інформаційних систем, основаних на різних стадіях життєвого циклу ПЗ, для забезпечення якості продукту, що випускається.

Так само має місце таке поняття, як контроль якості, який являє собою сукупність дій проведених над об'єктом тестування в процесі розробки для отримання інформації про актуальний стан об'єкта тестування в розрізах: "готовність Продукту до випуску", "Відповідність зафіксованим вимогам", "Відповідність заявленому рівню якості продукту ".

1 АНАЛІЗ ВИДІВ ТЕСТУВАННЯ

Тестування - це будь-яка діяльність, спрямована на виявлення помилок у програмному продукті. Тестування проводиться для того, щоб знайти помилки в програмі і тим самим підвищити її надійність, а отже, цінність.

Всі види тестування програмного забезпечення, залежно від переслідуваних цілей, можна умовно розділити на наступні групи:

1) функціональні;

2) нефункціональні;

3) пов'язані зі змінами.

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і можуть бути представлені на всіх рівнях тестування: компонентному або модульному (Component / Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing). Функціональні види тестування розглядають зовнішню поведінку системи. Далі перераховані одні з найпоширеніших видів функціональних тестів:

Функціональне тестування розглядає заздалегідь вказане поведінку і грунтується на аналізі специфікацій функціональності компонента або системи в цілому.

Функціональні тести грунтуються на функціях, виконуваних системою, і можуть проводитися на всіх рівнях тестування (компонентному, інтеграційному, системному). Як правило, ці функції описуються у вимогах, функціональних специфікаціях або у вигляді випадків використання системи (use cases).

Тестування функціональності може проводитися у двох аспектах:

– вимоги;

– бізнес-процеси.

Тестування в перспективі «вимоги» використовує специфікацію

функціональних вимог до системи як основу для дизайну тестових випадків (Test Cases). У цьому випадку необхідно зробити список того, що буде тестуватися, а що ні, пріоритезувати вимоги на основі ризиків (якщо це не зроблено в документі з вимогами), а на основі цього ставити в пріоритет тестові сценарії (test cases). Це дозволить сфокусуватися і не упустити при тестуванні найбільш важливий функціонал.

Тестування в перспективі «бізнес-процеси» використовує знання цих самих бізнес-процесів, які описують сценарії щоденного використання системи. У цій перспективі тестові сценарії (test scripts), як правило, ґрунтуються на випадках використання системи (use cases).

Переваги функціонального тестування:

– імітує фактичне використання системи;

Недоліки функціонального тестування:

– можливість упущення логічних помилок у програмному забезпеченні;

– ймовірність надмірного тестування.

Тестування безпеки - це стратегія тестування, використовувана для перевірки безпеки системи, а також для аналізу ризиків, пов'язаних із забезпеченням цілісного підходу до захисту додатки, атак хакерів, вірусів, несанкціонованого доступу до конфіденційних даних.

Загальна стратегія безпеки програмного забезпечення ґрунтується на трьох основних принципах:

1) конфіденційність;

2) цілісність;

3) доступність.

Конфіденційність - це приховування певних ресурсів або інформації. Під конфіденційністю можна розуміти обмеження доступу до ресурсу деякої категорії користувачів, або іншими словами, за яких умов користувач авторизований отримати доступ до даного ресурсу.

Існує два основних критерії при визначенні поняття цілісності:

1. Довіра. Очікується, що ресурс буде змінений тільки відповідним способом певною групою користувачів;

2. Пошкодження і відновлення. У разі коли дані пошкоджуються або неправильно змінюються авторизованим або авторизованим користувачем, ви повинні визначити на скільки важливою є процедура відновлення даних.

Доступність являє собою вимоги про те, що ресурси повинні бути доступні авторизованому користувачеві, внутрішньому об'єкту або пристрою. Як правило, чим більш критичний ресурс тим вище рівень доступності повинен бути.

В даний час найбільш поширеними видами уразливості в безпеці програмного забезпечення є:

– XSS (Cross-Site Scripting) - це вид уразливості програмного забезпечення (Web додатків), при якій, на генерованої сервером сторінці, виконуються шкідливі скрипти, з метою атаки клієнта;

– XSRF / CSRF (Request Forgery) - це вид уразливості, що дозволяє використовувати недоліки HTTP протоколу, при цьому зловмисники працюють за такою схемою: посилання на шкідливий сайт установливается на сторінці, що користується довірою у користувача, при переході по шкідливої посиланням виконується скрипт, який зберігає особисті дані користувача (паролі, платіжні дані і т.д.), або відправляє СПАМ повідомлення від особи користувача, або змінює доступ до облікового запису користувача, для отримання повного контролю над нею.

– Code injections (SQL, PHP, ASP і т.д.) - це вид уразливості, при якому стає можливо здійснити запуск виконуваного коду з метою отримання доступу до системних ресурсів, несанкціонованого доступу до даних або виведення системи з ладу;

– Server-Side Includes (SSI) Injection - це вид уразливості, що використовує вставку серверних команд в HTML код або запуск їх безпосередньо з сервера;

– Authorization Bypass - це вид уразливості, при якому можливе дістати несанкціонований доступ до облікового запису або документам іншого користувача.

Тестування на відмову і відновлення (Failover and Recovery Testing) перевіряє тестований продукт з точки зору здатності протистояти і успішно відновлюватися після можливих збоїв, що виникли у зв'язку з помилками програмного забезпечення, відмовами обладнання або проблемами зв'язку (наприклад, відмова мережі). Метою даного виду тестування є перевірка систем відновлення (або дублюючих основний функціонал систем), які, у разі виникнення збоїв, забезпечать збереження і цілісність даних тестованого продукту.

Тестування на відмову і відновлення дуже важливо для систем, що працюють за принципом "24x7". Якщо Ви створюєте продукт, який буде працювати, наприклад в інтернеті, то без проведення даного виду тестування Вам просто не обійтися. Так як. кожна хвилина простою або втрата даних у разі відмови обладнання, може коштувати вам грошей, втрати клієнтів і репутації на ринку.

Методика подібного тестування полягає в симулюванні різних умов збою і наступному вивченні та оцінці реакції захисних систем. У процесі подібних перевірок з'ясовується, чи була досягнута необхідна ступінь відновлення системи після виникнення збою.

Для наочності розглянемо деякі варіанти подібного тестування і загальні методи їх проведення.

Об'єктом тестування в більшості випадків є вельми вірогідні експлуатаційні проблеми, такі як:

– відмова електрики на комп'ютері-сервері;

– відмова електрики на комп'ютері-клієнті;

– незавершені цикли обробки даних (переривання роботи фільтрів даних, переривання синхронізації);

– оголошення або внесення в масиви даних неможливих або

помилкових елементів;

– відмова носіїв даних.

Дані ситуації можуть бути відтворені, як тільки досягнута деяка точка в розробці, коли всі системи відновлення або дублювання готові виконувати свої функції. Технічно реалізувати тести можна наступними шляхами:

– симулювати раптову відмову електрики на комп'ютері (знеструмити комп'ютер);

– симулювати втрату зв'язку з мережею (вимкнути мережевий кабель, знеструмити мережевий пристрій);

– симулювати відмова носіїв (знеструмити зовнішній носій даних);

– симулювати ситуацію наявності в системі невірних даних (спеціальний тестовий набір або база даних).

При досягненні відповідних умов збою і за результатами роботи систем відновлення, можна оцінити продукт з точки зору тестування на відмову. У всіх перерахованих вище випадках, по завершенні процедур відновлення, має бути досягнуто певний необхідний стан даних продукту:

– втрата або псування даних в допустимих межах;

– звіт або система звітів із зазначенням процесів або транзакцій, які не були завершені в результаті збою;

– нефункціональні види тестування.

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, які можуть бути виміряні різними величинами. В цілому, це тестування того, "Як" система працює. Далі перераховані основні види нефункціональних тестів:

Тестування навантаження або тестування продуктивності - це автоматизоване тестування, що імітує роботу певної кількості бізнес користувачів на якому-небудь загальному (розділяємому ними) ресурсі.

Розглянемо основні види навантажувального тестування, також завдання стоять перед ними.

Завданням тестування продуктивності є визначення масштабованості додатки під навантаженням, при цьому відбувається:

– вимірювання часу виконання обраних операцій при певних интенсивностях виконання цих операцій;

– визначення кількості користувачів, що одночасно працюють з додатком;

– визначення меж прийнятної продуктивності при збільшенні навантаження (при збільшенні інтенсивності виконання цих операцій);

– дослідження продуктивності на високих, граничних, стресових навантаженнях.

Стрес-тестування - один із видів тестування програмного забезпечення, яке оцінює надійність і стійкість системи в умовах перевищення меж нормального функціонування. Стрес-тестування особливо необхідно для «критично важливого» ПЗ, проте також використовується і для решти ПЗ. Зазвичай стрес-тестування краще виявляє стійкість, доступність і обробку винятків системою під великим навантаженням, ніж те, що вважається коректною поведінкою в нормальних умовах.

У загальному випадку методологія стрес-тестування заснована на знятті і аналізі показників продуктивності додатка при навантаженнях, які значно перевищують очікувані на стадії супроводу і несе в собі мету визначити витривалість або стійкість додатки на випадок сплеску активності щодо його використання.

Необхідність стрес-тестування диктується наступними чинниками:

– велика частина всіх систем розробляються з допущенням про функціонування в нормальному режимі і навіть у випадку, коли допускається можливість збільшення навантаження, реальні обсяги її збільшення не беруться до уваги;

– у разі SLA-контракту (угоди про рівень послуг) вартість відмови системи в екстремальних умовах може бути дуже велика;

– виявлення деяких помилок або дефектів у функціонуванні системи не завжди можливо з використанням інших типів тестування;

– тестування, проведеного розробником, може бути недостатньо для емуляції умов при яких відбувається відмова системи;

– переважно бути готовим до обробки екстремальних умов системи, ніж очікувати її відмови.

Основні напрямки застосування стрес-тестування:

1) загальне дослідження поведінки системи при пікових навантаженнях;

2) дослідження обробки помилок і виняткових ситуацій системою при пікових навантаженнях;

3) дослідження вузьких місць системи або окремих компонент при диспропорційних навантаженнях;

4) тестування ємності системи.

Стрес-тестування, як і тестування навантаження також може бути використано для регулярної оцінки змін продуктивності з метою отримання для подальшого аналізу динаміки зміни поведінки системи за тривалий період.

Стрес-тестування може застосовуватися як для відокремлених додатків, так і для розподілених систем з клієнт-серверною архітектурою. Найпростішим прикладом стрес-тестування відокремленого програми може бути відкриття файлу розміром в 50 Мб програмою Notepad, яка входить в комплект ОС Windows. Умови стрес-тестування програми зазвичай формуються виходячи з критичних бізнес-процесів його функціональності, визначеними на стадії розробки вимог та аналізу ризиків групою, відповідальною за продуктивність.

У загальному випадку в якості умов для стрес-тестування може використовуватися лінійно збільшена очікувана навантаження.

У разі тестування багатоланкових розподілених систем необхідно враховувати вже не тільки фактичний обсяг навантаження, що складається з

множини елементів, але і їх пропорції в загальному обсязі.

Завданням тестування стабільності (надійності) є перевірка працездатності програми при тривалому (багатогодинному) тестуванні з середнім рівнем навантаження. Час виконання операцій може грати в даному вигляді тестування другорядну роль. При цьому на перше місце виходить відсутність витоків пам'яті, перезапусків серверів під навантаженням та інші аспекти впливають саме на стабільність роботи.

Завданням об'ємного тестування є отримання оцінки продуктивності при збільшенні обсягів даних в базі даних програми, при цьому відбувається:

– вимірювання часу виконання обраних операцій при певних інтенсивностях виконання цих операцій;

– може проводитися визначення кількості користувачів, що одночасно працюють з додатком;

Тестування установки направлено на перевірку успішної інсталяції і налаштування, а також оновлення або видалення програмного забезпечення.

На даний момент найбільш поширена установка ПЗ за допомогою інсталяторів (спеціальних програм, які самі по собі так само потребують належного тестування, опис якого розглянуто в розділі "Особливості тестування інсталяторів.").

У реальних умовах інсталяторів може не бути. У цьому випадку доведеться самостійно виконувати установку програмного забезпечення, використовуючи документацію у вигляді інструкцій або readme файлів, крок за кроком описують всі необхідні дії та перевірки.

У розподілених системах, де додаток розгортається на вже працюючому оточенні, простого набору інструкцій може бути мало. Для цього, найчастіше, пишеться план установки (Deployment Plan), що включає не тільки кроки по інсталяції програми, але і кроки відкату (roll-back) до попередньої версії, у разі невдачі. Сам по собі план установки також повинен пройти процедуру тестування для уникнення проблем при видачі в реальну експлуатацію. Особливо це актуально, якщо установка виконується на

системи, де кожна хвилина простою - це втрата репутації та великої кількості коштів, наприклад: банки, фінансові компанії або навіть банерні мережі. Тому тестування установки можна назвати одним з найважливіших завдань щодо забезпечення якості програмного забезпечення.

Саме такий комплексний підхід з написанням планів, покрокової перевіркою установки і відкату інсталяції, повноправно можна назвати тестуванням установки або Installation Testing.

Іноді ми стикаємося з незрозумілими, нелогічними додатками, багато функцій і способи використання яких часто не очевидні. Після такої роботи рідко виникає бажання використовувати додаток знову, і ми шукаємо більш зручні аналоги. Для того щоб додаток був популярним, йому мало бути функціональним - воно має бути ще й зручним. Якщо задуматися, інтуїтивно зрозумілі додатки економлять нерви користувачам і витрати роботодавця на навчання. А значить вони більш конкурентоспроможні! Тому тестування зручності використання, про який піде мова далі є невід'ємною частиною тестування будь-яких масових продуктів.

Тестування зручності користування - це метод тестування, спрямований на встановлення ступеня зручності використання, навченості, зрозумілості та привабливості для користувачів розроблювального продукту в контексті заданих умов.

Тестування зручності користування дає оцінку рівня зручності використання програми за наступними пунктами:

– продуктивність, ефективність (efficiency) - скільки часу і кроків знадобиться користувачеві для завершення основних завдань програми, наприклад, розміщення новини, реєстрації, покупка і т.д. (Менше - краще);

– правильність (accuracy) - скільки помилок зробив користувач під час роботи з додатком? (Менше - краще);

– активізація в пам'яті (recall) - як багато користувач пам'ятає про роботу програми після припинення роботи з ним на тривалий період часу? (Повторне виконання операцій після перерви має проходити швидше ніж у

нового користувача);

– емоційна реакція (emotional response) - як користувач себе почуває після завершення завдання - розгублений, випробував стрес? Порекомендує користувач систему своїм друзям? (Позитивна реакція - краще).

Перевірка зручності використання може проводитися як по відношенню до готового продукту, за допомогою тестування чорного ящика (black box testing), так і до інтерфейсів додатки (API), використовуваним при розробці - тестування білого ящика (white box testing). У цьому випадку перевіряється зручність використання внутрішніх об'єктів, класів, методів і змінних, а також розглядається зручність зміни, розширення системи та інтеграції її з іншими модулями або системами. Використання зручних інтерфейсів (API) може поліпшити якість, збільшити швидкість написання і підтримки коду, що розробляється, і як наслідок поліпшити якість продукту в цілому.

Звідси стає очевидно, що тестування зручності користування може проводитися на різних рівнях розробки програмного забезпечення: модульному, інтеграційному, системному і приймальному. При цьому воно цілком і повністю буде залежить від того, хто буде використовувати додаток на виділеному конкретному рівні - розробника, бізнес користувач системи і т.д.

Статичне тестування проводиться без запуску програмного коду продукту. Тестування здійснюється шляхом аналізу програмного коду (code review) або скомпільованого коду. Аналіз може проводитися як вручну, так і за допомогою спеціальних інструментальних засобів. Метою аналізу є раннє виявлення помилок і потенційних проблем у продукті.

За допомогою code review на ранньому етапі можуть бути виявлені помилки в коді продукту. Як правило code review виробляється самими розробниками.

Прикладами помилок, які потенційно можна виявити за допомогою автоматичного статичного тестування, можуть бути:

– виток ресурсів (витоку пам'яті, файлові дескриптори, що не звільняються і т.д.);

– можливість переповнення буфера (buffer overflows);

– ситуації часткової (неповної) обробки помилок.

Як правило, результатом автоматичного аналізу коду є список рекомендацій для ручного review деяких ділянок коду,де потенційно міститися помилки.

Цей вид тестування ПЗ застосовується на завершальних фазах розробки ПЗ. Під час динамічного тестування тестувальник ставить себе на місце користувача і виявляє дефекти в роботі програми. Тестувальники вводять різні вхідні дані і досліджують відмінності вихідних даних, спостерігають за продуктивністю комп'ютера. На підставі отриманих результатів роблять висновок про готовність програмного забезпечення.

Під час тестування на сумісність одні й ті ж тести проводять на різних комп'ютерних платформах, на серії конфігурацій.

Статичний аналіз коду - аналіз програмного забезпечення, вироблений (на відміну від динамічного аналізу) без реального виконання досліджуваних програм. У більшості випадків аналіз проводиться над якою-небудь версією вихідного коду, хоча іноді аналізу піддається якій-небудь вид об'єктного коду, наприклад Р-код або код на MSIL. Термін зазвичай застосовують до аналізу, виробленому спеціальним програмним забезпеченням (ПЗ), тоді як річний аналіз називають «program understanding», «program comprehension» (розумінням або осягненням програми).

Залежно від використовуваного інструменту глибина аналізу може варіюватися від визначення поведінки окремих операторів до аналізу, що включає весь наявний вихідний код. Способи використання отриманої в ході аналізу інформації також різні - від виявлення місць, можливо з помилками (утиліти типу Lint), до формальних методів, що дозволяють математично довести будь-які властивості програми (наприклад, відповідність поведінки специфікації).

При динамічному тестуванні потрібно враховувати такі фактори як справність апаратури, налагодженість і правильна настройка операційної системи, компіляторів та багато інших.Динамічне тестування дозволяє перевірити продукт на відповідність вимогам з точки зору функціональності, ефективності роботи і т.д.

Конфігураційне тестування (Configuration Testing) - спеціальний вид тестування, спрямований на перевірку роботи програмного забезпечення при різних конфігураціях системи.

Залежно від типу проекту конфігураційне тестування може мати різну мету:

1) проект по профілізації роботи системи;

2) проект з міграції системи з однієї платформи на іншу.

Для клієнт-серверних додатків конфігураційне тестування можна умовно розділити на два рівні (для деяких типів додатків може бути актуальний тільки один):

1) серверний;

2) клієнтський.

На першому (серверному) рівні, тестується взаємодія випускається програмного забезпечення з оточенням, в яке воно буде встановлено:

1) апаратні засоби (тип і кількість процесорів, об'єм пам'яті, характеристики мережі / мережевих адаптерів);

2) програмні засоби (ОС, драйвера і бібліотеки, стороннє ПЗ, що впливає на роботу програми).

Основний упор тут робиться саме на тестування з метою визначення оптимальної конфігурації обладнання, що повністю задовольняє необхідним характеристикам якості (ефективність, портативність, зручність супроводу, надійність).

На наступному (клієнтському) рівні, програмне забезпечення тестується з позиції його кінцевого користувача і конфігурації його робочої станції. На цьому етапі будуть протестовані наступні характеристики:

зручність використання, функціональність. Для цього необхідно буде провести ряд тестів з різними конфігураціями робочих станцій:

1) тип, версія і бітность операційної системи (подібний вид тестування має назву кросс-платформенне тестування);

2) тип і версія Web барузер, у разі якщо тестується Web додаток (подібний вид тестування називається крос-браузерні тестування);

3) тип і модель відео адаптера (при тестуванні ігор це дуже важливо);

4) робота програми при різних дозволах екрану;

5) версії драйверів, бібліотек і т.д. (Для JAVA додатків версія JAVA машини дуже важлива, теж можна сказати і для .NET додатків щодо версії .NET бібліотеки) і т.д.

Перед початком проведення конфігураційного тестування рекомендується:

– створювати матрицю покриття (матриця покриття - це таблиця, вяку заносять усі можливі конфігурації);

– проводити пріоритезацію конфігурацій (на практиці, швидше за все, всі бажані конфігурації перевірити не вийде);

– крок за кроком, відповідно до розставленими пріоритетами, перевіряють кожну конфігурацію.

Вже на початковому етапі стає очевидно, що чим більше вимог до роботи програми при різних конфігураціях робочих станцій, тим більше тестів нам необхідно буде провести. У зв'язку з цим, рекомендуємо, по можливості, автоматизувати цей процес, оскільки саме при конфігураційному тестуванні автоматизація реально допомагає заощадити час і ресурси. Звичайно ж автоматизоване тестування не є панацеєю, але в даному випадку воно виявиться дуже ефективним помічником.

Після проведення необхідних змін, таких як виправлення бага / дефекту, програмне забезпечення має бути перетестовано для підтвердження того факту, що проблема була дійсно вирішена. Нижче перераховані види тестування, які необхідно проводити після установки програмного

забезпечення, для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

Поняття димове тестування пішло з інженерної середовища:"При введенні в експлуатацію нового обладнання ("заліза") вважалося, що тестування пройшло вдало, якщо з установки не пішов дим."

В області ж програмного забезпечення, димове тестування розглядається як короткий цикл тестів, що виконується для підтвердження того, що після складання коду (нового або виправленого) встановлюване додаток, стартує і виконує основні функції.

Висновок про працездатність основних функцій робиться на підставі результатів поверхневого тестування найбільш важливих модулів програми на предмет можливості виконання необхідних завдань та наявності критичних і блокуючих дефектів,що швидко знаходяться. У разі відсутності таких дефектів димове тестування оголошується пройденим, і додаток передається для проведення повного циклу тестування, в іншому випадку, димове тестування оголошується проваленим, і додаток йде на доопрацювання.

Аналогами димового тестування є Build Verification Testing і Acceptance Testing, що виконуються на функціональному рівні командою тестування, за результатами яких робиться висновок про те, приймається чи ні встановлена версія програмного забезпечення в тестування, експлуатацію або на поставку замовнику.

Регресійне тестування - це вид тестування спрямований на перевірку змін, зроблених в додатку або навколишньому середовищу (лагодження дефекту, злиття коду, міграція на іншу операційну систему, базу даних, веб-сервер або сервер додатки), для підтвердження того факту, що існуюча раніше функціональність працює як і колись (див. також Санітарне тестування або перевірка узгодженості / справності). Регресійний можуть бути як функціональні, так і нефункціональні тести.

Як правило, для регресійного тестування використовуються тест кейси,

написані на ранніх стадіях розробки і тестування. Це дає гарантію того, що зміни в новій версії програми не пошкодили вже існуючу функціональність. Рекомендується робити автоматизацію регресійних тестів, для прискорення подальшого процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення.

Сам по собі термін "Регресійне тестування", залежно від контексту використання може мати різний зміст. Сем Канер, наприклад, описав 3 основних типи регресійного тестування:

– регресія багів (Bug regression) - спроба довести, що виправлена помилка насправді не виправлена;

– регресія старих багів (Old bugs regression) - спроба довести, що нещодавня зміна коду або даних зламало виправлення старих помилок, тобто старі баги стали знову відтворюватися;

– регресія побічного ефекту (Side effect regression) - спроба довести, що нещодавня зміна коду або даних зламало інші частини розробляється;

Тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. За своїми цілями є аналогом димових Тестування, спрямованого на приймання нової версії в подальше тестування або експлуатацію. Вглиб воно може проникати далі, залежно від вимог до якості випущеної версії.

Санітарне тестування - це вузьконаправлене тестування достатню для доказу того, що конкретна функція працює згідно заявленим в специфікації вимогам. Є підмножиною регресійного тестування. Використовується для визначення працездатності певної частини програми після змін вироблених в ній або навколишньому середовищу. Зазвичай виконується вручну.

Відмінність санітарного тестування від димового (Sanity vs Smoke testing).У деяких джерелах помилково вважають, що санітарне та димове тестування - це одне і теж. Ми ж вважаємо, що ці види тестування мають "вектора руху", напрямки в різні боки. На відміну від димового (Smoke testing), санітарний тестування (Sanity testing) направлено вглиб

перевіряється функції, в той час як димове направлене в ширину, для покриття тестами якомога більшої функціоналу в найкоротші терміни.

1.1 Огляд методів тестування

Існує кілька методів тестування:

- тестування програм методом "чорного ящика" (Black box testing);
- тестування софта методом "білого ящика" (White box);
- тестування ПЗ методом "сірого ящика" (Grey box).

У термінології професіоналів тестування (програмного і деякого апаратного забезпечення) фрази "тестування білого ящика" і "тестування чорного ящика" ставляться до того, чи має розробник тестів і тестувальник доступ до вихідного коду тестованого ПЗ, або ж тестування виконується через інтерфейс користувача або прикладний програмний інтерфейс, наданий тестованим модулем.

При тестуванні білого ящика (англ. White-box testing, також кажуть - прозорого ящика), розробник тесту має доступ до вихідного коду і може писати код, який пов'язаний з бібліотеками тестованого ПЗ. Це типово для юніт-тестування (англ. Unit testing), при якому тестуються лише окремі частини системи. Воно забезпечує те, що компоненти конструкції - працездатні і стійкі, до певної міри.

При тестуванні чорного ящика (англ. Black-box testing), тестувальник має доступ до ПЗ тільки через ті ж інтерфейси, що і замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншого комп'ютера або іншому процесу підключитися до системи для тестування. Наприклад, тестуючий модуль може віртуально натискати клавіші або кнопки миші в тестованій програмі за допомогою механізму взаємодії процесів, з упевненістю в тому, чи все йде правильно, що ці події викликають

той же відгук, що й реальні натискання клавіш і кнопок миші. Як правило, тестування чорного ящика ведеться з використанням специфікацій або інших документів, що описують вимоги до системи.

Бета-тестування в цілому обмежена технікою чорного ящика (хоча постійна частина тестувальників зазвичай продовжує тестування білого ящика паралельно бета-тестування). Таким чином, термін «бета-тестування» може вказувати на стан програми (ближче до випуску ніж «альфа»), або може вказувати на деяку групу тестувальників і процес, що виконується цією групою. Отже, тестувальник може продовжувати роботу з тестування білого ящика, хоча ПЗ вже «в беті» (стадія), але в цьому випадку він не є частиною «бета-тестування» (групи / процесу).

Існують спеціальні методи для тестування аспектів програм, які не є функціональними, тобто не відносяться до працездатності самих програм. Це тестування:

– тестування продуктивності програмного забезпечення - подивитися працездатність, якщо програма управляє великою кількістю даних або має велике число користувачів. Це безпосередньо відноситься до поняття масштабованості додатків;

– тестування "Юзабіліті" - тестування інтерфейсу користувача, його зручності, практичності і легкості для освоєння звичайним користувачем;

– тестування безпеки програм важливо для програм, що мають справу з конфіденційними даними для запобігання використання вразливостей хакерами;

– тестування якості інтернаціоналізації та локалізації програмного забезпечення.

Користуватися цими методами можна і потрібно, щоб програма була якісною.

1.2 Огляд існуючих програмних продуктів для навантажувального тестування

На сьогоднішній день існує безліч засобів для автоматизованого тестування, що дозволяє зробити процес тестування ПЗ / web-додатків / мобільних пристроїв більш легким, швидким і доступним.

За допомогою використання різних драйверів та інструментів, можна створювати скрипти і виконувати безліч команд для найбільш точної перевірки відповідності будь-якого продукту стандартам і поставленим вимогам.

Для порівняння характеристик існуючих інструментів, за визначенням переваг і недоліків кожного з них, буде доцільним розглянути деякі з них, такі як: WAPT, Cucumber, Selenium, Silk Test, WinRunner, TestComplete і Rational Functional Tester.

WAPT є засобом навантажувального і стресового тестування, дає можливості для детального та ефективного тестування веб-сайтів та Інтернет додатків з веб-інтерфейсом. Використовуючи WAPT можна тестувати і аналізувати характеристики продуктивності та вузькі місця веб-сайтів за різних умов навантаження.

Опис графіків і звітів дозволить вам аналізувати експлуатаційні характеристики компонентів системи при різних умовах навантаження, виявляти й усувати будь-які вузькі місця і оптимізувати програмне забезпечення та апаратну конфігурацію.

Cucumber - це програмний інструмент, який виконує автоматизовані приймальні випробування, написані на підставі стилю поведінки (BDD). Cucumber, написаний на мові програмування Ruby, дозволяє реалізувати характерну документацію, написану в діловій манері.

Selenium - це Java-додаток, який може аналізувати файли певної структури для того, щоб знаходити в них команди для маніпуляції браузером

і команди для виконання певних дій і перевірок.

Selenium підтримується Microsoft Internet Explorer, Google Chrome, Mozilla Suite і Mozilla Firefox для Microsoft Windows, Linux і Apple Macintosh.

Інструмент Selenium IDE, являє собою версію досить популярної бібліотеки Selenium в GUI-оболонці. Реалізовано це у вигляді розширення до браузера Firefox, розміром близько 240 кб, включаючи сам Selenium. Цей інструмент дозволяє записувати і відтворювати скрипти, що представляють собою звичайні HTML-сторінки з однією таблицею, що містить команди.

Данний продукт підтримує широкий спектр додатків таких як десктопне Web, SAP, і .NET

Існує два різних підходи до створення тестів: Visual test і .NET

Перший спосіб вкрай простий і зрозумілий - це запис всіх дій користувача з захопленням об'єктів, екранів та інших параметрів. Цей спосіб доступний навіть тим, хто ніколи не займався тестуванням. Інший спосіб передбачає написання скрипта .NET.

Обидва способи можуть бути використані один з одним, тому тести можуть вкладені один в інший, тим самим створювати залежність з ланцюжка тестів.

Також є можливість роботи з VB.NET, C #, Java. Т.к. Silk Test здатний інтегруватися з Visual Studio і Eclipse

SilkTest є інструментом для автоматизованого функціонального і регресивного тестування корпоративних додатків.

SilkTest може бути запропонований різним клієнтам:

SilkTest Класичний використовує домен специфічний для конкретної мови 4Test для автоматизації сценаріїв. Це об'єктно-орієнтована мова схожий на C ++. Він використовує поняття класів, об'єктів та успадкування.

Silk4J дозволяє автоматизувати В Eclipse використання Java як мови сценаріїв

Silk4Net дозволяє те ж саме в Visual Studio за допомогою VB або C #

SilkTest Workbench дозволяє автоматизувати тестування на візуальному

рівні (за аналогією з колишнім TestPartner), а також за допомогою VB.Net в якості мови сценаріїв

SilkTest Host: містить всі вихідні файли зі скриптами.

SilkTest Agent: переводить сценарій команди в GUI (дії користувача). Ці команди можуть бути виконані на тій же машині, як і хост або на віддаленому комп'ютері.

SilkTest визначає всі вікна і елементи управління тестованого програми в якості об'єктів і визначає всі властивості й атрибути кожного вікна. Таким чином, підтримує об'єктно-орієнтовану реалізацію (частково).

SilkTest може бути запущений, щоб визначити рух миші разом з клавішами (корисно для користувацького об'єкта) . Можно використовувати, як запис, так і відтворення або описові методи програмування, щоб захопити діалоги.

Розширення, підтримувані SilkTest: .NET, Java (Swing, SWT), DOM, IE, Firefox, SAP Windows GUI.

SilkTest використовує Silk Bitmap Tool (bitview.exe), щоб захопити і порівняти вікна та області.

Програмне забезпечення HP WinRunner це автоматизований функціональний інструмент тестування GUI, який дозволяє поставити користувачеві записувати і відтворювати користувальницький інтерфейс (UI) взаємодії як тестові сценарії.

В якості функціонального набору тестів, він працює з HP QuickTest Professional і підтримує гарантію якості підприємства. Він захоплює, перевіряє і автоматично відтворює взаємодії з користувачем, з метою виявлення недоліків і визначення, чи працюють бізнес-процеси як задумано.

Програмне забезпечення реалізовано патентованим мовою Test Script Language (TSL), що дозволяє настройку та параметризацію користувача введення.

TestComplete використовується для створення та автоматизації багатьох різних типів тестування програмного забезпечення. Запис і

відтворення створення тестів записує дії тестера і виконується за допомогою ручного тестування ,що дозволяє їм бути відтвореними і зберігатися знову і знову, як автоматизований тест. Записані тести можуть бути змінені пізніше шляхом тестерів, щоб створити нові тести або розширити існуючі тести з більш варіантів використання.

Основні характеристики:

– Keyword Testing: TestComplete має вбудований клавіатурний редактор тестів, який містить клавіатурні операції, які відповідають автоматизованим дії тестування.

– Scripted Testing: TestComplete має вбудований редактор коду, який допомагає тестерам писати сценарії вручну. Він також включає в себе набір спеціальних допоміжних плагінів.

– Test Record and Playback: TestComplete записує основні дії, необхідні для відтворення тесту і відкидає всі непотрібні дії.

– Distributed Testing: TestComplete можете запустити кілька автоматизованих тестів між різними робочими станціями або віртуальними машинами.

– Access to Methods and Properties of Internal Objects: TestComplete зчитує імена видимих елементів і багатьох внутрішніх елементів Delphi, C ++ Builder, .NET, WPF, Java і додатків Visual Basic і дозволяє тестовим скриптам отримувати доступ до цих значення для перевірки або використовувати в тестах 4;

– Bug Tracking Integration: TestComplete включає в себе шаблони відстеження проблем, які можуть бути використані для створення або зміни елементів, що зберігаються в системах відстеження проблем. TestComplete в даний час підтримує Microsoft Visual Studio 2005, 2008, 2010 Team System, Bugzilla, Jira і AutomatedQA AQdevTeam;

– Data-driven testing: тестування на основі даних, що надходять з TestComplete означає використання одного тесту, щоб перевірити різні тест-

кейси допомогою запуску тесту з вхідними і очікуваними значення від зовнішнього джерела даних, замість використання складно-кодованих значень кожного разу при запуску тесту;

– COM-based, Open Architecture: двигун TestComplete заснований на відкритому API, інтерфейсі COM. Це джерело мовою незалежний, і може читати інформацію відладчика і використовувати його під час виконання через TestComplete Debug Info Agent.

Тест Visualizer - TestComplete автоматично захоплює скріншоти під час випробування запису і відтворення. Це дозволяє швидко порівнювати очікуваний і фактичний екрани під час тесту.

Extensions and SDK - Все візуалізовано в TestComplete - панелі, елементи проекту, конкретні об'єкти сценаріїв, а інші - реалізовані у вигляді плагінів. Ці плагіни включені в продукт і встановлені на вашому комп'ютері разом з іншими модулями TestComplete. Ви можете створювати свої власні плагіни, які розширять TestComplete і забезпечать специфічні можливості для власних потреб. Наприклад, ви можете створити плагіни або використовувати сторонні плагіни для:

1) підтримки користувачем елементів управління;
2) користувальницькі тестові операції з клавіатурою;
3) нові об'єкти сценаріїв;
4) користувальницькі контрольно-пропускні пункти;
5) команди для обробки результатів тестування;
6) панелі;
7) елементи проекту;
8) меню і панелей інструментів.

IBM Rational Functional Tester - це інструмент автоматичного функціонального тестування та регресійного тестування. Це програмне забезпечення надає функції автоматичного тестування для функціонального, регресійного тестування, тестування графічних користувацьких інтерфейсів і тестування, орієнтованого на дані. Rational Function Tester підтримує ряд

додатків, таких як веб-додатки, додатки для .Net, Java, Siebel, SAP, додатки на основі емулятора термінала, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, документи Adobe PDF, zSeries, iSeries і pSeries.

Тестування розкадровки: спрощує візуалізацію тестування і зміна за допомогою природної мови і виведених знімків екрану.

Автоматичне тестування: дозволяє випробувачам автоматизувати тести, стійкі до частих змін з боку користувача інтерфейсу програми, за допомогою технології Script Assure.

Тестування, орієнтоване на дані: дозволяє виконувати однакові послідовності дій тестування над змінюються набором даних тестування.

Сценарії тестування: об'єднує реєстратор дій користувача з декількома опціями настройки і функціями інтелектуального обслуговування сценарію.

Інтеграції: інтегрується з IBM Rational Team Concert і IBM Rational Quality Manager, забезпечуючи доступ до завдань і логічним і складовим ресурсам тестування SCM.

1.3 Постановка завдання

У даній роботі слід провести аналіз даних, отриманих при тестуванні деякої кількості сайтів. При тестуванні були використані тестові сценарії, розроблені на підставі підбору дій користувача, які найчастіше мають місце при знаходженні на сайті. Були проведені такі види тестування, як тестування навантаження, тобто була проведена імітація одночасного перебування на сайті великої кількості користувачів, які здійснювали аналогічну послідовність дій і тестування продуктивності, а саме - досліджувався середня кількість часу, потрібного для вчинення таких дій, як, наприклад, процес завантаження сторінки, процес відкриття пункту меню і т.д.

Результатом завершення процесу тестування та аналізу даних є збір статистики, заснованої на отриманій інформації про взаємодію користувачів з сайтами. Для проведення відповідних розрахунків передбачається використання такого програмного засобу, як Matematica. За отриманими цифрам будуть побудовані графіки для більш наочної ілюстрації наявної статистики.

2 МАТЕМАТИЧНА ПОСТАНОВКА ЗАВДАННЯ

Нехай сайт являє собою деякий граф G = <E, L>, де E-множина вершин-сторінок сайту, L- множина всіх дуг-переходи між сторінками.

Для кожної сторінки EI є E = (e1, e2, ...) відомий наступний набір показників:

$$e_i=(c_i , t_i^{min}(n), t_i^{avg}(n), t_i^{max}(n)),$$

де c_i – контент представлений на сайті / на сторінці e_i // текст, відео, картинки та інше.

T_{imin} (n) – залежність мінімального часу завантаження сторінки від кількості n одночасних користувачів на e_i сторінці. Відповідно $t_i$avg (n), $t_i$max(n) –це функції залежності середнього і максимального часу.

Проведемо експеримент: створимо одночасну навантаження на сторінку e_i у 10 користувачів і побудуємо гістограму часу завантаження сторінки для кожного користувача:

Наприклад, графік для користувачів буде виглядати так (рис.2.1):

Рисунок 2.1 – Гістограма розподілу часу для 10 користувачей

В якості критерію оцінки стійкості сайту виберемо наступне:

$$\Delta t^{avg}=1\backslash N\Sigma \mid t^{avg}(n_{01})- t^{avg}(n_{02})\mid$$
$$N=(10,100,700)\ ,\ |N|\ =3.$$

k-коефіцієнт кореляції між числом користувачів і часом завантаження сторінки.

Число користувачів, для яких час завантаження сторінки не перевищує необхідний:

$$n_0= \{n \in N :t^{avg}(n_0) <t^{треб.} \}.$$

Узагальнена оцінка сайту будується як:

$$z= \frac{1}{n}\sum_{i=0}^{n} \Delta\tilde{t}iavg\ /w,i=1,\ldots,n\},$$

де w – досяжність сторінки з головної сторінки сайту.

3 МЕТОДИ ТА АЛГОРИТМИ

3.1 Алгоритм оцінки досяжності на графі

Один з перших питань, що виникають при вивченні графів, це питання про існування шляхів між заданими або всіма парами вершин. Відповіддю на це вопрос - відношення досяжності на вершинах графа G = (V, E): вершина w досяжна з вершини v, якщо v = w або в G є шлях з v в w. Інакше кажучи, ставлення досяжності є рефлексивним і транзитивним замиканням відношення E. Для неорієнтованих графів це відношення також симетрично і, отже, є відношенням еквівалентності на множині вершин V. В неорієнтованому графі класи еквівалентності по відношенню досяжності називаються зв'язковими компонентами. Для орієнтованих графів досяжність, взагалі кажучи, не повинна бути симетричним відношенням. Симетричною є взаємна досяжність.

Вершини v і w орієнтованого графа G = (V, E) називаються взаємно досяжними, якщо в G є шлях із v в w і шлях з w в v.

Зрозуміло, що ставлення взаємної досяжності є рефлексивним, симетричним і транзитивним і, отже, еквівалентністю на множині вершин графа. Класи еквівалентності по відношенню взаємної досяжності називаються компонентами сильної зв'язності або двузв’язних компонентів графа.

Розглянемо спочатку питання про побудову відносини досяжності. Визначимо для кожного графа його граф досяжності (званий іноді також графом транзитивного замикання), ребра якого відповідають шляхах вихідного графа.

Нехай G = (V, E) - орієнтований граф. Граф досяжності G * = (V, E *) для G має те ж множина вершин V і наступне множина ребер E * = {(u, v) | у графі G вершина v досяжна з вершини u}.

Розглянемо граф G(рис 3.1)

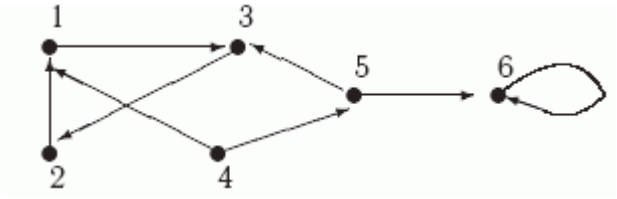


Рисунок3.1 - Граф G

Тоді можна перевірити, що граф досяжності G * для G виглядає так:(рис.3.2)

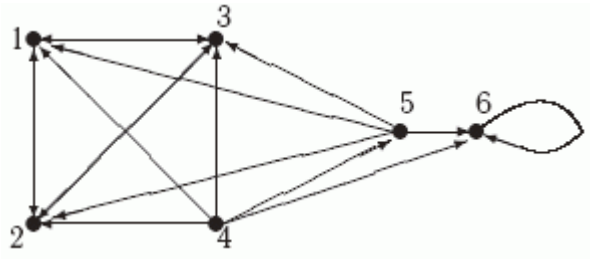


Рисунок 3.2 -Граф G*

Яким чином по графу G можна побудувати граф G *? Один спосіб полягає в тому, щоб для кожної вершини графа G визначити множину досяжних з неї вершин, послідовно додаючи в нього вершини, досяжні з неї шляхах і довжини 0, 1, 2.

Ми розглянемо інший спосіб, заснований на використанні матриці суміжності AG графа G і булевих операцій. Пустьмножество вершин V = {v1, ..., vn}. Тоді матриця AG - це булева матриця розміру nx n.

Нижче для збереження подібності зі звичайними операціями над матрицями ми будемо використовувати «арифметичні» позначення для булевих операцій: через + будемо позначати диз'юнкцію а через x - кон'юнкцію.

Позначимо через En одиничну матрицю розміру nx n.

Покладемо

$$\tilde{A} = A_G + E_n,$$

Нехай $\tilde{A}$=E_n, $\tilde{A}_1$= $\tilde{A}$,…,$\tilde{A}_{k+1}$=$\tilde{A}_k \cdot \tilde{A}$.Наша процедура побудови G * заснована на наступному твердженні.

Лемма 3.1. Нехай $\tilde{A}_k$=($a^{(k)}_{ij}$). Тоді

$$\begin{cases} a(k)ij = 1, \text{в } G \text{ з } vi \text{у } vj \text{маємо шлях довжини } \leq k, \\ 0, \text{у іншому випадку,} \end{cases}$$

Доказ проведемо індукцією по k.

Базис. При k = 0 і k = 1 твердження справедливо по визначенню $\tilde{A}^0$

Індукційний крок. Нехай лема справедлива для k. Покажемо, що вона залишається справедливою і для k + 1. за визначенням$\tilde{A}^{k+1}$ маємо:

$$a^{(k+1)}_{ij} = a^{(k)}_{i1}\, a^{(1)}_{1j} + \ldots + a^{(k)}_{ir}\, a^{(1)}_{rj} + \ldots a^{(k)}_{in} a^{(1)}_{nj}\ .$$

Припустимо, що в графі G з vi в vj є шлях довжини <= k + 1. Розглянемо найкоротший з таких шляхів. Якщо його довжина <= k, то за припущенням індукції a_ {ij} ^ {(k)} = 1. Крім того, ajj (1) = 1. Тому aij (k) ajj (1) = 1 і aij (k + 1) = 1. Якщо довжина найкоротшого шляху з з vi в vj дорівнює k + 1, то нехай vr - його передостання вершина. Тоді з vi в vr мається шлях довжини k і за припущенням індукції air (k) = 1. так як (v_r,v_j)є E, тоді a_{rj}^{(1)}=1. Тому air(k) arj(1)=1и aij(k+1)=1.

Та навпаки, якщо aij (k + 1) = 1, то хоча б для одного r доданок air (k) arj (1) в сумі дорівнює 1. Якщо це r = j, то aij (k) = 1 і по індуктивному припущенням в G мається шлях з vi в vj довжини <= k. якщо ж r≠j, тоді a_{ir}(k)=1 и a_{rj}(1)=1. Це значить, що у G має місце шлях з v_i в vr довжини <= k й ребро (v_r,v_j) є E. Об'єднав їх, отримуємо шлях з v_i у v_j довжини <= k+1.

Нехай G = (V, E) - орієнтований граф з n вершин і, а G * - його граф досяжності. Тоді A_ {G *} =$\tilde{A}_{n-1}$. Доказ. З леми випливає, що, якщо в G є шлях з u вv≠ u, то в ньому є і простий шлях з u BV довжини <= n-1. А по лемі

всі такі шляхи представлені в матриці $\tilde{A}^{n-1}$.

Таким чином процедура побудови матриці суміжності AG * графа досяжності для G зводиться до зведення матриці$\tilde{A}$ у ступінь n-1. Зробимо декілька зауважень, що дозволяють спростити цю процедуру.

Для зведення матриці $\tilde{A}$ в довільну ступінь n досить виконати

$[\log_2 n]$ возведений в квадрат:

$$\tilde{A} => \tilde{A}^2 => (A^2)^2 => \ldots => (\tilde{A}^2)^k,$$

де k - це найменше число таке, що 2k> = n.

Так як на діагоналі у матриці $\tilde{A}$ стоять одиниці, то для будь-яких i <j всі одиниці матриці $\tilde{A}$i зберігаютьсяуматриці $\tilde{A}$j, зокрема, й у матриці $(\tilde{A}i)^{2.}$

Якщо при обчисленні елемента aij(2) матриці $(\tilde{A}k)2$ згідно з стандартною формулою:

$$a^{(2)}_{ij} = a_{i1}a_{1j} + a_{i2}a_{2j} + \ldots + a_{ir}a_{rj} + \ldots + a_{in}a_{nj},$$

виявляється таке r, що $a_{ir} = 1$ й $a_{rj} = 1$, тоді й вся сума $a_{ij}(2) = 1$. Тому інші складові можна не розглядати.

На етапі контролю якості реалізованої функціональності використовується тестова документація, в якій записані стандартні і альтернативні сценарії роботи з додатком, використовувані при тестуванні чергової версії програми.

Тестова документація складається з тестових сценаріїв, тобто спеціальним чином розробленого опису послідовності дій в системі і очікуваного поведінки,що у даній роботі виконуються у Selenium IDE.(рис.3.3)

Тестові сценарії використовуються для проведення різних видів

ручного тестування:

– функціонального тестування;
– приймального тестування;
– навантажувального або стрес-тестування;
– дослідного тестування;
– smoke-тестування та ін.

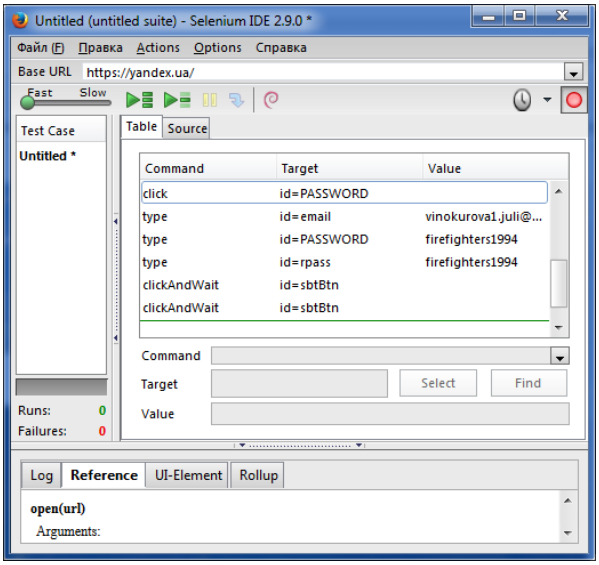


Рисунок 3.3– Створення тестового сценарію у Selenium

3.2 Метод побудови нечіткого виводу

3.2.1 Основні правила виведення в нечіткої логіки

Припустимо, що присутні в правилах modus ponens і modus tollens судження характеризуються деякими нечіткими множинами. Таким чином ми отримуємо узагальнене правило висновку modus ponens і узагальнене правило висновку modus tollens, таблиці 3.1 - 3.4. У подальшому викладі залежності типу "Якщо A, то B" будемо записувати у символіці класичної мови програмування ALGOL у вигляді IF A THEN B. З цієї причини в деяких російськомовних формулах будуть зустрічатися англійські терміни.

Таблиця 3.1 - Правило виводу modus ponens

Умова імплікації	A $A \rightarrow B$
Висновок	B

Таблиця 3.2 - Правило виводу modus tollens

Умова імплікації	B $A \rightarrow B$
Висновок	A

Таблиця 3.3 - Узагальнене (нечітке) правило modus ponens

Умова імплікації	x це A` IF x це A THEN y це B
Висновок	y це B`

де A, A` $\subseteq$ X и B, B` $\subseteq$ X - нечіткі множини, тоді як x та y - так звані лінгвістичні змінні.

Лінгвістичними називаються змінні, значення яких представляють собою слова або судження на природній мові. В якості прикладів можна навести вирази типу "мала швидкість", "помірна температура" або "молода людина". Подібні висловлювання можна формалізувати приписуванням їм деяких нечітких множин. Слід підкреслити, що лінгвістичні змінні крім словесних значень можуть мати і чисельні значення - так само, як звичайні математичні змінні.

Таблиця 3.4 - Узагальнене (нечітке) modus tollens

Умова імплікації	*y* це *B*` IF *x*це*A* THEN *y*це*B*
Висновок	*x* це *A*`

де *A*, *A*`$\subseteq$ X и *B*, *B*`$\subseteq$ X - нечіткі множини в той час як x та y - так звані лінгвістичні змінні.

3.2.2 Правила нечіткої імплікації

Нехай A та B - це нечіткі множини, A $\subseteq$ X и B $\subseteq$ Y. Нечіткої імплікацією A $\rightarrow$ B називається відношення R, визначене на X $\times$ Y та задовольняє наступним правилам:

1) Правило типу minimum, також зване «правило Мамдані»

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=\mu_A(x)\wedge\mu_B(y)=\min[\mu_A(x),\mu_B(y)].$$

2) Правило типу "відношення", відоме як «правило Ларсена»

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=\mu_A(x)\cdot\mu_B(y).$$

3) Правило Лукашевича

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=1\wedge[1-\mu_A(x)+\mu_B(y)]=\min[1,1-\mu_A(x)+\mu_B(y)].$$

4) Бінарне правило

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=[1-\mu_A(x)]\vee\mu_B(y)=\max[1-\mu_A(x),\mu_B(y)].$$

5) Правило Гогуен

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=1\vee\frac{\mu_B(x)}{\mu_A(y)}=\min[1,\frac{\mu_B(x)}{\mu_A(y)}].$$

6) Правило Шарпа

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=\begin{cases}1, & \textit{если} \quad \mu_A(x)\le\mu_B(y),\\ 0, & \textit{если} \quad \mu_A(x)>\mu_B(y).\end{cases}$$

7) Правило Геделя

$$\mu_{A\to B}(x,y)=\mu_R(x,y)=\begin{cases}1, & \textit{если} \quad \mu_A(x)\le\mu_B(y),\\ \mu_B(y), & \textit{если} \quad \mu_A(x)>\mu_B(y).\end{cases}$$

Набір цих правил не вичерпує всі відомі з літератури визначення нечіткої імплікації.

3.2.3 Нечітке управління

Для багатьох додатків, пов'язаних з управлінням технологічними процесами, необхідна побудова моделі розглянутого процесу. Володіння моделі дозволяє підібрати відповідний регулятор (модуль управління). Однак часто побудова коректної моделі являє собою важку проблему, що вимагає іноді введення різних спрощень. Застосування теорії нечітких множин для управління технологічними процесами не припускає знання моделей цих

процесів. Слід тільки сформулювати правила поведінки у формі нечітких умовних суджень типу IF ... THEN.

Слід підкреслити, що додатки нечітких множин охоплюють в даний час широкий спектр завдань - від простих пристроїв побутового призначення до більш серйозних систем.

3.2.4 Класичний модуль нечіткого управління

На рисунку 3.4 представлена типова структура модуля нечіткого управління. Він складається з таких компонентів: бази правил, блоку фазифікації, блоку вироблення рішення, блоку дефаззифікації.

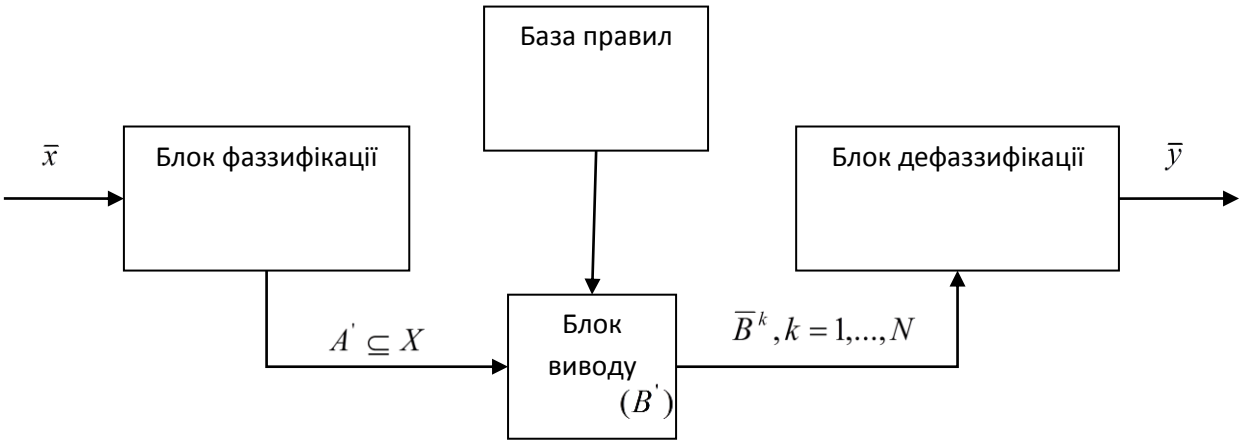


Рисунок 3.4 - Модуль нечіткого управління

3.2.5 База правил

База правил, іноді називається лінгвістичною моделлю, являє собою множину нечітких правил $R^{(k)}, k = 1,..., N$, виду

$$R^{(k)}: \text{IF } (x_1 \text{ це } A_1^k \text{ AND } x_2 \text{ це } A_2^k \text{ ... AND } x_n \text{ це } A_n^k)$$
$$\text{THEN } (y_1 \text{ це } B_1^k \text{ AND } y_2 \text{ це } B_2^k \text{ ... AND } y_m \text{ це } B_m^k),$$

де:

N - кількість нечітких правил,

A_i^k – нечіткі множини

$$A_i^k \subseteq X_i \subset R, i = 1,..., n,$$

B_i^k – нечіткі множини

$$B_j^k \subseteq Y_j \subset R, j = 1,..., m,$$

$x_1, x_2,..., x_n$ - вхідні змінні лінгвістичної моделі,

причому

$$(x_1, x_2,..., x_n)^T = x \in X_1 \times X_2 \times ... \times X_n,$$

$y_1, y_2,..., y_m$ - вихідні змінні лінгвістичної моделі, причому

$$(y_1, y_2,..., y_m)^T = y \in Y_1 \times Y_2 \times ... \times Y_m.$$

Символами $X_i, i=1,...,n$ та $Y_j, j=1,...,m$ позначаються відповідно простору вхідних і вихідних змінних.

Для подальших міркувань приймемо, що конкретні правила $R^{(k)}, k=1,...,N$ пов'язані між собою логічним оператором "АБО". Крім того, припустимо, що виходи $y_1, y_2,..., y_m$ взаємно незалежні. Тому без втрати спільності будемо використовувати нечіткі правила зі скалярним виходом у формі

$$R^{(k)}: \text{IF } (x_1 \text{ это } A_1^k \text{ AND } x_2 \text{ это } A_2^k ... \text{AND } x_n \text{ это } A_n^k)$$
$$\text{THEN } (y \text{ это } B^k), \qquad (3.1)$$

де $B_i^k \subseteq Y_j \subset R$ та $k=1,...,N$.

Зауважимо, що кожне правило виду (3.1) складається з частини IF, званою посилкою, і частини THEN, званою слідством. Посилка правила містить набір умов, тоді як слідство містить висновок. Змінні $x=(x_1, x_2,..., x_n)^T$ та у можуть приймати як лінгвістичні, так і числові значення.

Якщо ввести позначення

$$X = X_1 \times X_2 \times ... \times X_n,$$
$$A^k = A_1^k \times A_2^k \times ... \times A_n^k,$$

то правило (3.1) можна представити у вигляді нечіткої імплікації

$$R^{(k)}: A^k \rightarrow B^k, \; k=1,...,N.$$

Звернемо увагу, що правило $R^{(k)}$ також можна інтерпретувати як нечітке відношення, визначене на множині X×Y, тобто $R^{(k)} \subseteq$X×Y - це не чітке множина з функцією приналежності

$$\mu_{R^{(k)}}(x,y) = \mu_{A^k \to B^k}(x,y).$$

При проектуванні модулів нечіткого управління слід оцінювати достатність кількості нечітких правил, їх несуперечливість та наявність кореляції між окремими правилами.

3.2.6 Блок фазифікації

Система управління з нечіткою логікою оперує нечіткими множинами. Тому конкретне значення $\bar{x} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)^T \in$X вхідного сигналу модуля нечіткого управління підлягає операції фаззифікації, в результаті якої йому буде зіставлена нечітка множина $A' \subseteq X = X_1 \times X_2 \times ... \times X_n$. У задачах управління найчастіше застосовується операція фаззифікації типу синглетон (singleton):

$$\mu_{A'}(x) = \delta(x, \bar{x}) = \begin{cases} 1, & \textit{если} \quad x = \bar{x}, \\ 0, & \textit{если} \quad x \neq \bar{x}. \end{cases}$$

Нечітка множина A` подається на вхід вироблення рішення. Якщо вхідний сигнал надходить зашумлений, то нечітку множину A` можна визначити функцією приналежності

$$\mu_{A'} = \exp[-\frac{(x-\bar{x})^T(x-\bar{x})}{\sigma^2}],$$

де σ> 0. У цьому випадку операція фаззифікації має тип non-singleton.

3.2.7 Блок вироблення рішення

Припустимо, що на вхід блоку вироблення рішення подано нечітка множина $A' \subseteq X = X_1 \times X_2 \times ... \times X_n$. На виході цього блоку також з'явиться відповідна нечітка множина. Розглянемо два випадки, яким будуть відповідати різні методи дефаззифікації.

У першому випадку, на виході блоку вироблення рішення відповідно з узагальненим нечітким правилом modus ponens отримуємо N нечітких множин $\bar{B}^k \subset$ Y, таблиця 3.5.

Нечітка множина $\bar{B}^k$ визначається комбінацією нечіткої множини A' та відношення $R^{(k)}$,

тобто

$$\bar{B}^k = A' \circ (A^k \to B^k),\ k = 1,..., N.$$

Таблиця 3.5 - Блок вироблення рішення (N нечітких множин)

Умова	$x = (x_1, x_2, \ldots, x_n)^T$ це A' $A' = A_1' \times A_2' \times \ldots \times A_n'$
Імплікація	$R^{(k)}: A^k \rightarrow B^k,$ $k = 1, \ldots, N$ $A^k = A_1^k \times A_2^k \times \ldots \times A_n^k$
Висновок	y це $\overline{B}^k$, $k = 1, \ldots, N$

Функцію приналежності нечіткої множини $\overline{B}^k$ можна задати у вигляді

$$\mu_{\overline{B}^k}(y) = \sup_{x \in X}[\mu_{A'}(x)^T * \mu_{A^k \rightarrow B^k}(x, y)].$$

Конкретна форма функції $\mu_{\overline{B}^k}(y)$ залежить від застосованої Т-норми, визначення нечіткої імплікації $R^{(k)}$ і від способу визначення декартового добутку нечітких множин. Слід зазначити, що якщо виконується операція фаззифікації типу синглетон, то попередній вираз приймає вигляд

$$\mu_{\overline{B}^k}(y) = \mu_{A^k \rightarrow B^k}(\overline{x}, y).$$

У другому випадку, на виході блоку вироблення рішення отримуємо одну нечітку множину $\overline{B}^k \subseteq Y$ по узагальненому нечіткому правилу modes ponens, яке приймає вигляд, таблиця 3.6.

При використанні комбінованого правила виведення отримуємо

$$B' = A' \circ \bigcup_{k=1}^{N} R^{(k)} = A' \circ R$$

де $\Re = \bigcup_{k=1}^{N} R^{(k)}$. Функція приналежності має вигляд:

$$\mu_{B'}(y) = \sup_{x \in X}[\mu_{A'}(x)^T * \max_{1 \le k \le N} \mu_{R^{(k)}}(x, y)].$$

Таблиця 3.6 - Блок вироблення рішення (одна нечітка множина)

Умова	$X = (x_1, x_2, ..., x_n)^T$ це A' $A' = A'_1 \times A'_2 \times ... \times A'_n$
Імплікація	$\bigcup_{k=1}^{N} R^{(k)}$, $R^{(k)}: A^k \to B^k$ $A^k = A_1^k \times A_2^k \times ... \times A_n^k$
Висновок	y це B'

Зауважимо, що виведення за цією схемою - це результат комбінування посилки A' і глобального правила (відносини) $\Re$, яке представляє собою узагальнення окремих правил $R^{(k)}$, $k = 1,..., N$. Будемо називати такий прийом глобальним підходом до проблеми синтезу блоку вироблення рішення.

3.2.8 Блок дефаззіфікації

На виході блоку вироблення рішення формується або N нечітких множин $\overline{B}^k$ (випадок 1) з функціями приналежності $\mu_{B^k}(y)$, $k = 1, 2, ...N$, або одна нечітка множина B' (випадок 2) з функцією приналежності $\mu_{B'}(y)$. Встає

завдання відображення нечітких множин $\overline{B}^k$ (або нечіткої множини $B^{'}$) у єдине значення $\bar{y} \in Y$, яке представляє собою керуючий вплив, що подається на вхід об'єкта. Таке відображення називається дефаззифікації, і реалізується воно в однойменному блоці.

Якщо на виході блоку вироблення рішення формується N нечітких множин B^k, то значення $\bar{y} \in Y$ можна розрахувати за допомогою різних методів:

Метод дефаззифікації по середньому центру. Значення $\bar{y}$ розраховується за формулою

$$\bar{y} = \frac{\sum_{k=1}^{N} \mu_{\overline{B}^k}(\bar{y}^k)\bar{y}^k}{\sum_{k=1}^{N} \mu_{\overline{B}^k}(\bar{y}^k)},$$

де $\bar{y}^k$ - це точка, в якій функція $\mu_{B^k}(y)$ приймає максимальне значення, тобто

$$\mu_{\overline{B}^k}(\bar{y}^k) = \max_y \mu_{B^k}(y).$$

Точка $\bar{y}^k$ називається центром нечіткої множини B^k.

Метод дефаззифікації за сумою центрів. Значення розраховується наступним чином:

$$\bar{y} = \frac{\int_Y y \sum_{k=1}^{N} \mu_{\overline{B}^k}(y)dy}{\int_Y \sum_{k=1}^{N} \mu_{\overline{B}^k}(y)dy}.$$

Якщо вихідне значення блоку вироблення рішення являє собою єдину нечітку множину $B^{'}$, то значення $\bar{y}$ можна визначити з застосування

наступних двох методів.

Метод центру ваги. Значення $\bar{y}$ розраховується як центр ваги функції приналежності $\mu_{B'}(y)$, тобто

$$\bar{y} = \frac{\int\limits_{Y} y\mu_{B'}(y)dy}{\int\limits_{Y} \mu_{B'}(y)dy} = \frac{\int\limits_{Y} y \max\limits_{k} \mu_{\bar{B}^k}(y)}{\int\limits_{Y} \max\limits_{k} \mu_{\bar{B}^k}(y)},$$

за умови, що обидва інтеграла в наведеному вираженні існують.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Тестування у Selenium IDE

Як інструмент тестування веб-додатків був використаний веб-драйвер IDE -Selenium. За допомогою нього був проведений покроковий набір дій, виконуваних на обраному сайті з метою його спрощеного тестування.

Була протестована така фунціональна частина веб-додатків, як заповнення користувачем форми реєстрації на сайті.

Для цього, після відкриття Selenium в браузері Firefox, були виконані такі дії:

1) У полі Base URL було введене посилання на відповідний сайт;

2) У полі Table, при натисканні правої клавіші миші була обрана опція «Insert new command»;

3) В полі «Command» була обрана команда open, яка завантажила даний сайт для роботи з ним драйвера;

4) Для запису всіх скоєних дій була натиснута кнопка «Start Recording»;

У полі «Value» були введені рядки voidSendKeys (рядок тексту), якими були заповнені відповідні поля;

Для відправки даних на сервер була використана команда void Submit().

5) Для припинення запису була натиснута кнопка «Stop Recording».

Після завершення попередніх дій у вікні драйвера була натиснута кнопка «Play current test case», яка дозволила пройти даний тестовий сценарій.

Інтерфейс веб – драйвера показан на рисунку 4.1

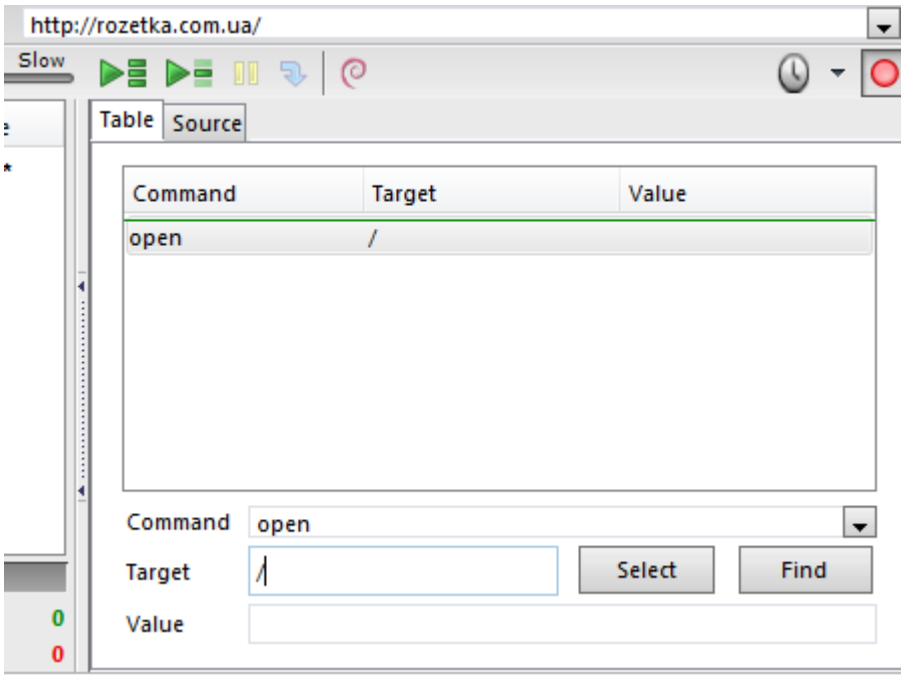


Рисунок 4.1 – Діалогове вікно Selenium IDE

4.2 Навантажувальне тестування у WAPT

4.2.1 Створення сценарію тесту

Слід виконати наступну послідовність дій:

1) Натисніть кнопку "Новий" на панелі інструментів, щоб створити новий сценарій;

2) Виберіть опцію "Тест Продуктивності" на першій сторінці Майстра;

3) Виберіть тип навантаження;

4) Задайте тривалість тесту;

5) Зазначте кількість етапів тесту.

4.2.2 Тривалість тесту і параметри звіту

1.Задати тривалість тесту;

2.Укажіте кількість етапів тесту;

3.Перегляд підказок щодо тесту.

Інтерфейс WAPT зображений на рисунку 4.2 та 4.3.

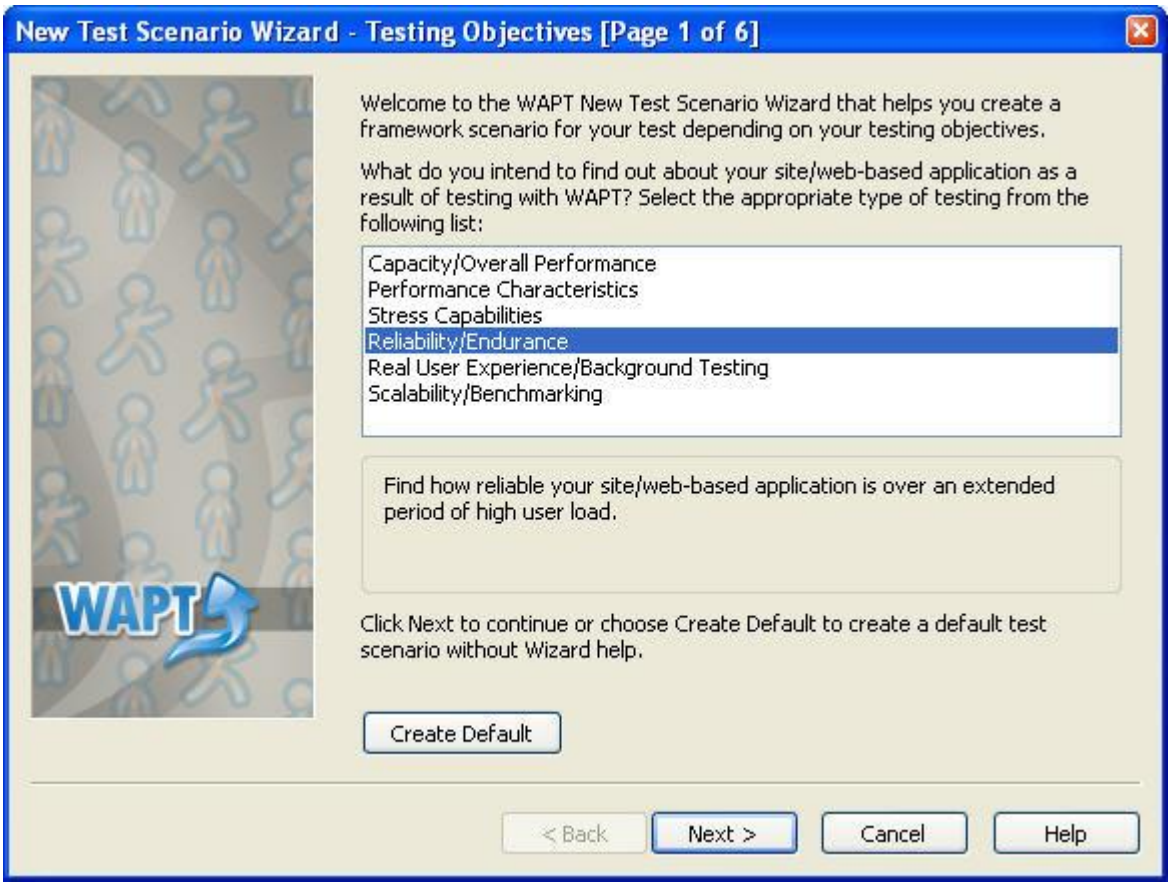


Рисунок 4.2 – Діалогове вікно створення нового сценарія у WAPT

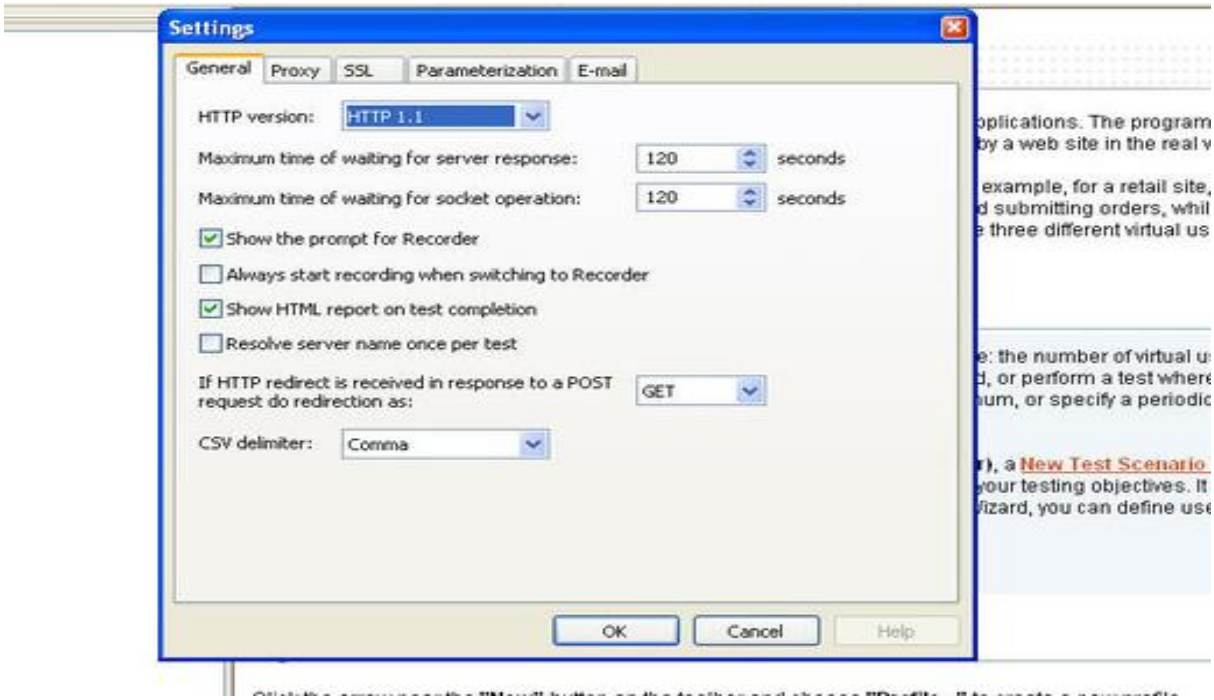


Рисунок 4.3 – Діалогове вікно завдання парметрів у WAPT

5 АНАЛІЗ РЕЗУЛЬТАТІВ

У результаті проведення навантажувального тестування веб-сервісу були отримані дані, що характеризували поведінку сайту під час того, як на нього заходили користувачі.

Нижче на графіку наочно показана поведінка сайта при навантаженні його користувачами на першій та дев’ятій сторінках (рис. 5.1 та рис. 5.2). Також на графіку показана швидкість зростання часу з збільшенням кількості користувачів, що заходять на сайт. Кількість користувачів,що зображена на обох графіках, є однаковою.

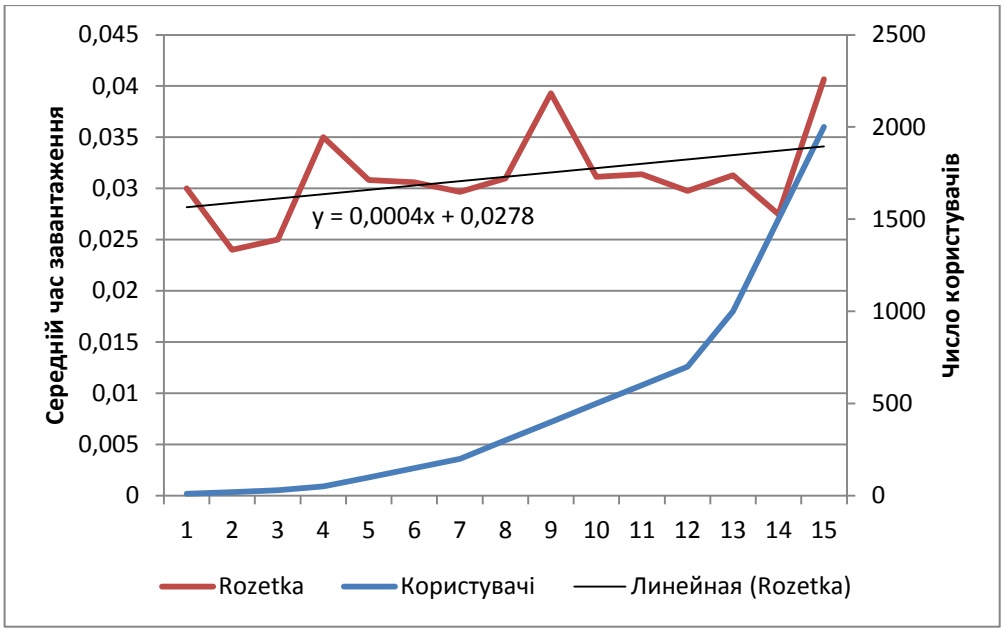


Рисунок 5.1 –Навантаження користувачами 1-ї сторінки сайту

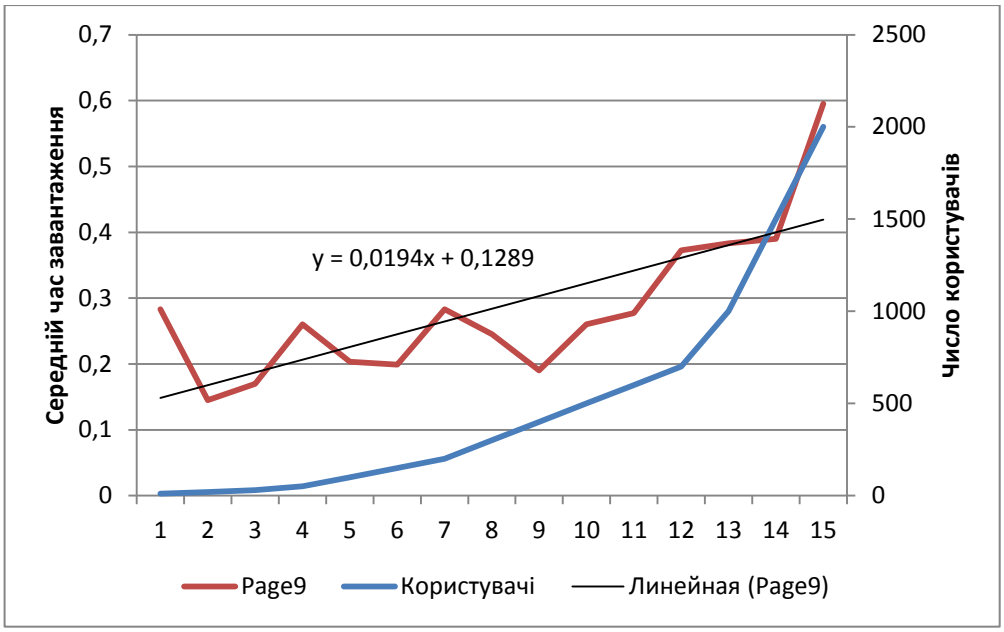


Рисунок 5.2 –Навантаження користувачами 9-ї сторінки сайту

Отримавши показники сторінок сайта, ми вирахували мінімальний, середній та максимальний час швидкості завантаження кожної з сторінок сайта по мірі надходження користувачів.

Таблиця 5.1 –Результати проведення навантажувального тестування сайта Rozetka

Користувачі	Rozetka	Page2	Page3	Page4	Page5	Page6	Page7	Page8	Page9	Page10	Page11
10	0,030	0,028	0,018	0,013	0,050	0,083	0,018	0,013	0,283	0,359	0,366
20	0,024	0,045	0,015	0,015	0,024	0,045	0,015	0,045	0,145	0,238	0,246
30	0,025	0,050	0,020	0,015	0,050	0,070	0,017	0,015	0,170	0,179	0,188
50	0,035	0,056	0,016	0,016	0,035	0,056	0,016	0,026	0,260	0,286	0,295
100	0,031	0,029	0,020	0,013	0,053	0,089	0,019	0,013	0,204	0,229	0,235
150	0,031	0,049	0,016	0,015	0,029	0,046	0,015	0,045	0,199	0,230	0,233
200	0,030	0,054	0,021	0,019	0,050	0,077	0,018	0,029	0,283	0,329	0,338
300	0,031	0,057	0,016	0,016	0,056	0,076	0,016	0,015	0,245	0,266	0,274
400	0,039	0,056	0,020	0,019	0,059	0,081	0,019	0,030	0,190	0,260	0,268
500	0,031	0,033	0,017	0,013	0,051	0,088	0,016	0,014	0,260	0,313	0,320
600	0,031	0,063	0,017	0,019	0,059	0,077	0,017	0,030	0,277	0,289	0,293
700	0,030	0,058	0,020	0,019	0,054	0,078	0,020	0,030	0,373	0,438	0,444
1000	0,031	0,061	0,017	0,019	0,051	0,079	0,017	0,030	0,383	0,481	0,487
1500	0,027	0,050	0,020	0,016	0,059	0,075	0,017	0,015	0,390	0,488	0,497
2000	0,041	0,064	0,018	0,015	0,062	0,086	0,018	0,016	0,595	0,604	0,607
Среднее время	**0,031**	**0,050**	**0,018**	**0,016**	**0,049**	**0,074**	**0,017**	**0,024**	**0,284**	**0,333**	**0,339**
Минимальное	**0,024**	**0,028**	**0,015**	**0,013**	**0,024**	**0,045**	**0,015**	**0,013**	**0,145**	**0,179**	**0,188**
Максимальное	**0,041**	**0,064**	**0,021**	**0,019**	**0,062**	**0,089**	**0,020**	**0,045**	**0,595**	**0,604**	**0,607**
Корреляция	**0,422**	**0,470**	**0,164**	**0,229**	**0,560**	**0,376**	**0,221**	**-0,230**	**0,900**	**0,891**	**0,889**
Скорость роста	**0,000**	**0,002**	**0,000**	**0,000**	**0,002**	**0,001**	**0,000**	**0,000**	**0,019**	**0,020**	**0,020**

Виходячи з даних, якими ми володіємо, були сформовані нечіткі правила, логіка яких була побудована на основі дерев рішень.

Таблиця 5.2–Побудова нечітких правил

№	Условие	Результат
1	Среднее время < 0,20312 **И**Среднее время < 0,061908 **И**Корреляция < -0,033019	1
2	Среднее время < 0,20312 **И**Среднее время < 0,061908 **И**Корреляция >= -0,033019 **И**Среднее время < 0,017711 **И**Среднее время < 0,016642	2
3	Среднее время < 0,20312 **И**Среднее время < 0,061908 **И**Кордареляция >= -0,033019 **И**Среднее время < 0,017711 **И**Среднее время >= 0,016642	1
4	Среднее время < 0,20312 **И**Среднее время < 0,061908 **И**Корреляция >= -0,033019 **И**Среднее время >= 0,017711	2
5	Среднее время < 0,20312 **И**Среднее время >= 0,061908	1
6	Среднее время >= 0,20312	0

У результаті аналізу отриманих нечітких правил, були зроблені висновки щодо двох вибраних сторінок сайта, а саме щодо першої та третьої сторінки.

Швидкість роботи сайта на першій сторінці була оцінена як достатньо висока і тому їй був назначений пріоритет 2. (рис.5.3)

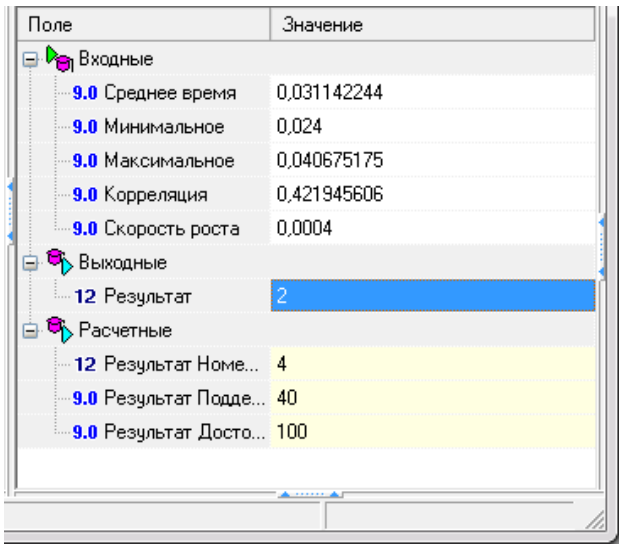


Рисунок 5.3 – Висновки щодо сайту по нечітким правилам (перша сторінка)

Швидкість роботи сайта на дев’ятій сторінці була оцінена як дуже

погана і тому ій був назначений пріоритет 0. (рис. 5.4)

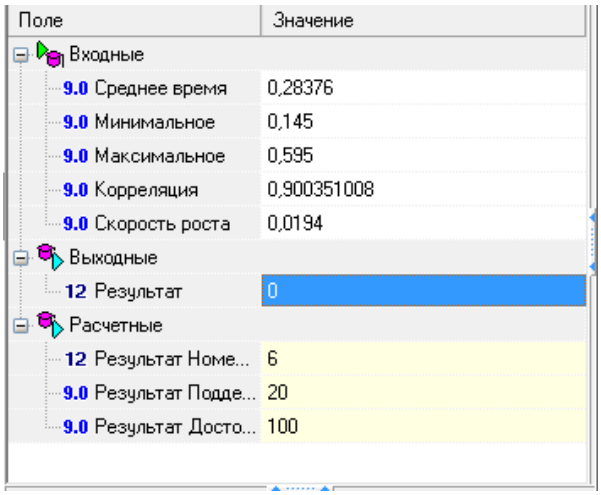


Рисунок 5.4 – Висновки щодо сайта по нечітким правилам (дев’ята сторінка)

Таблиця 5.2 – Результати проведення навантажувального тестування сайта Rozetka

Удаленность	1	2	3	4	5	6	7	8	9	10	11	Интегральная оценка суммарный результат	**Интегральная оценка средний результат**	Интегральная оценка максимальный результат
ср.время/удале	0,031	0,025	0,006	0,004	0,010	0,012	0,002	0,003	0,032	0,033	0,031	0,190	**0,017**	0,033
мин.время/удал	0,024	0,014	0,005	0,003	0,005	0,008	0,002	0,002	0,016	0,018	0,017	0,114	**0,010**	0,024
мак.время/удал	0,041	0,032	0,007	0,005	0,012	0,015	0,003	0,006	0,066	0,060	0,055	0,302	**0,027**	0,066
корр/удаленнос	0,422	0,235	0,055	0,057	0,112	0,063	0,032	-0,029	0,100	0,089	0,081	1,216	**0,111**	0,422
скор. роста/удал	0,000	0,001	0,000	0,000	0,000	0,000	0,000	0,000	0,002	0,002	0,002	0,008	**0,001**	0,002

При вирахуванні інтегральної оцінки сайта, були отримані результати, які засвідчили, що у середньому сайт справляється з навантаження на середньому рівні, саме тому був назначений пріоритет 1.(рис. 5.5)

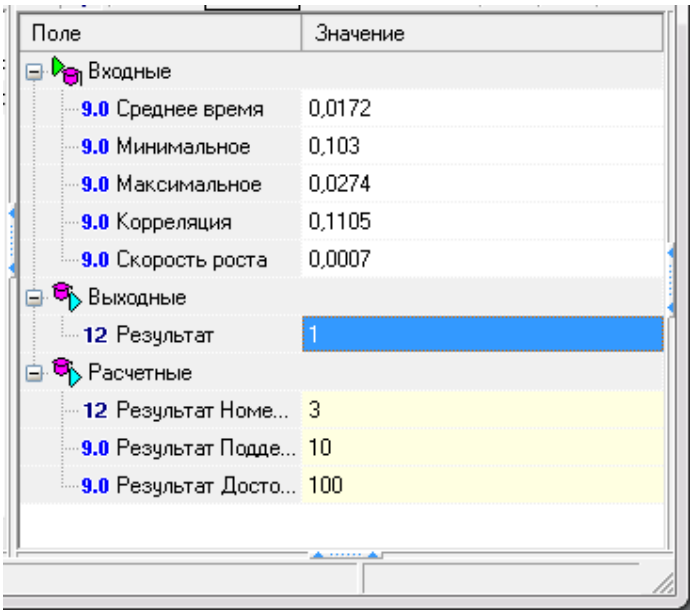


Рисунок 5.5 – Висновки по інтегральній оцінці

6 ОХОРОНА ПРАЦІ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

6.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання ДСанПіН 3.3.2.007-98 [21], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне елект-рообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є наступні:

- робоча напруга U = +220В ± 5%;
- робочий струм I = 2А;
- споживана потужність P = 350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з ві-зуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [21].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мі-крокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 6.1).

Таблиця 6.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[22]
- підвищений рівень шуму на робочому місці	-//-	2	[23]
- підвищена або знижена рухливість повітря	-//-	1	[22]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[25]
- підвищена напруженість електричного поля	-//-	2	[25]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[26]

Продовження таблиці 6.1

- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[26]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[21] [30]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безпервна робота)	2	[21] [30]

6.2 Гігієнічні вимоги до параметрів виробничого середовища

6.2.1 Мікроклімат

Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають ДСН 3.3.6.042-99 [22] і наведені в табл. 6.2:

Таблиця 6.2 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату:

температури, відносної вологості й рухливості повітря у відповідності до ДСН 3.3.6.042-99 [22]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви в роботі співробітників, з метою його провітрювання. Існують спеціальні системи кондиціонування, які забезпечують підтримання в приміщенні балансу оптимальних параметрів мікроклімату. Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

6.2.2 Освітлення

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В. 2.5-28:2018 [26]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b = (1/8 \div 1/10) \cdot S_n \qquad (6.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$S_n = a \cdot b = 5 \cdot 5 = 25 \text{ м}^2$$

$$S_{вік} = 1/8 \cdot 25 = 3{,}125 \text{ м}^2$$

Приймаємо 2 вікна площею $S = 1,6\ м^2$ кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників п виробляється по формулі (6.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \qquad (6.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (6.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2.$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

6.2.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [23]. Шум такої інтенсивності на тлі високого ступеня напруженості праці негативно впливає на функціональний стан користувачів. Тому на практиці рекомендують знижувати фактичний рівень шуму у приміщеннях, де створюють комп'ютерні програми, виконують теоретичні та творчі роботи, проводять навчання до 40 дБА, а в приміщеннях, де виконують роботу, що потребує зосередженості, — до 55 дБА. У залах опрацювання інформації та комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Шум часто є причиною зниження рівня працездатності, підвищення рівня загальної та професійної захворюваності, частоти виробничих травм. Шум є загальнобіологічним подразником, який негативно впливає на всі органи і системи організму. У разі тривалого систематичного впливу шуму може виникнути патологія з переважним ураженням слуху, центральної нервової і серцево- судинної систем.

Для зниження шуму на шляху його поширення передбачається розміщення в приміщенні штучних поглиначів. Для зниження рівня шуму

стелю або стіни вище 1.5 - 1.7 метра від підлоги повинні облицьовуватися звукопоглинальним матеріалом з максимальним коефіцієнтом звукопоглинання в області частот 63-8000 Гц. Додатковим звукопоглинанням в КВТ можуть бути фіранки, підвішені в складку на відстані 15-20 см. Від огорожі, виконані з щільної, важкої тканини. У приміщенні з ЕОМ коректований рівень звукової потужності не перевищує 45 дБА. Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами, заходи для зниження шуму не проводяться.

Віброізоляцію можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [23]. Допустимий рівень вібрацій на робочому місці: для 1 ступеня шкідливості до 3 дБ; для 2-3 - 1-6 дБ; для 3 - більше 6 дБ.

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1) застосування нових плазмових моніторів, LG W2271TC,

2) віддалення робочого місця не менше, ніж на 0,4-0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля,

3) встановлення раціональних режимів роботи персоналу (обмеження часу перебування),

4) раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

6.2.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки в віконному отворі автономного кондиціонера БК-2000. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП (30 м3 на годину на одного працюючого).

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

6.3 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п}, \qquad (6.3)$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

Rш - опір опори, яке з'єднує заземлювачі, Ом;

п - кількість заземлювачів;

ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; ηз = 0,7

ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; ηш = 0,5;

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right), \quad (6.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м;

t = 2м.

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3} \left(\ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \ln \frac{4 \cdot 2 + 3}{4 \cdot 2 - 3} \right) = 110 \text{, Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_ш = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1}, \quad (6.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: Пбуд. = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t1 - глибина заземлення від рівня землі, м; t1 = 0,5 м.

$$R_п = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot \ln \frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99 \text{, Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \tag{6.6}$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: Rззп < 4 Ом.

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони

повинні бути в робочому стані і перевірятися згідно з нормами. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

ВИСНОВКИ

У даній дипломній роботі була протестована деяка кількість веб-застосунків за допомогою імітації навантажувального тестування. Також був проведений аналіз даних,отриманих після тестування, які були представлені у вигляді графіків, що наочно продемонструвало усі етапи роботи веб-застосунків під час навантаження їх користувачами. В результаті була оцінена якість сайтів за їх основними характеристиками, що дозволило зробити висновки щодо їх працездатності під час критичних ситуацій.

Надалі буде введена можливість автоматизації запуску сценарія навантажувального тестування «у глибину», тобто для усього сайта, висновків про сайт, а також рекомендацій щодо поліпшення його функціонування.

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Було наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Тестування навантаження [Електронний ресурс]/ПроТестинг.ru. – Режим доступа: www/URL: http://www.protesting.ru/automation/performance.html – 03.05.2020г. –Загол. з екрану.

2) Що таке Selenium? [Електронний ресурс]/Selenium+WebDriver.Автоматизація додатків через браузер.– Режим доступа: www/ URL :http://selenium2.ru/– 04.05.2020г.– Загол. з екрану.

3) Автоматизоване тестування. [Електронний ресурс]/Википедия.Свободная энциклопедия.– Режим доступа: www/URL: https://ru.wikipedia.org/wiki/Автоматизированное_тестирование–04.05.2020г. – Загол. з екрану.

4) Виды тестирования [Електронний ресурс] / ПроТестинг.ru.– Режим доступа: www/URL: http://www.protesting.ru/testing/testtypes.html–07.05.2020г. – Загол. з екрану.

5) Орлик, С. Програмна інженерія. Тестування програмного забезпечення [Текст] / С.Орлик – М. : SWEBOK, 2004. - 36 с.

6) Канер, С. Тестування програмного забезпечення. Фундаментальні концепції менеджменту бізнес -додатків [Текст] / С. Канер, Д. Фолк, Енг. - К.: Видавництво «Диасофт», 2001. - 544 с.

7) Тамре, Л. Введення в тестування програмного забезпечення [Текст] / Л. Тамре. – М. : Видавничий будинок «Вільямс», 2003. -356 с.

8) Бейзер Б. Тестування чорного ящика. Технології функціонального тестування програмного забезпечення й систем [Текст] / Б.Бейзер. - М.: IT-Stars, 1998. - 387 с.

9) ANSI/IEEE Std 610.12-1990. IEEE Standart Glossary of Software Engineering Terminology [Текст]. - IEEE Computer Society Press, 1990.

10) ANSI/IEEE Std 829-1998. IEEE Standart Glossary for Software Test

Documentation [Текст]. - IEEE Computer Society Press, 1998.

11) Леоненков, Ф. Нечітке моделювання у середі MATLAB й fuzzyTECH [Текст] / Ф.Леоненков – СПб.:БХВ-Петербург, 2005. – 736 с.

12) А.В. Леоненков. Нечітке моделювання у середі MATHLAB й fuzzyTECH. – СПб.: БХВ-Петербург, 2003.

13) G. Chen, J. Vanthienen, G. Wets. Fuzzy decision tables: extending the classical formalism to enhance intelligent decision making // Proc. of the Fourth IEEE International Conference on Fuzzy Systems, 1995, vol. 2, pp. 599 - 606.

14) F. Witlox, T. Arentze, H. Timmermans. Constructing and consulting fuzzy decision tables // Decision support systems in urban planning, 1997, pp. 156-174.

15) Агапов А. С., Зенин С. В., Михайловский Н. Э., Мкртумян А. А. Оцінка й атестація зрілості процесів створення і супроводження програмних засобів і інформаційних систем(ISO/IEC TR 15504-CMM), Пер. с англ. Москва, "Книга и бізнес", 2001.

16) Лямец В.И., Тевяшев А.Д. Системний аналіз. Вводний курс.: Учбовий посібник. – Харків: ХТУРЕ, 1998.– 252с.

17) Побудова нечітких правил. [Електронний ресурс]/Наукова бібліотека.−Режим доступа: www/URL: http://sernam.ru/book_gen.php?id=26 – 08.05.2020г. – Загол. з екрану.

18) Системи нечіткого виводу. [Електронний ресурс]/БСТУ.− Режим доступа: www/URL: http://nrsu.bstu.ru/chap27.html –08.05.2020г. – Загол. з екрану.

19) Каскиаро, М. Шаблоны проектирования Node.js [Текст] / М. Каскиаро, Л. Маммино ; пер. с англ. А.Н. Киселева – М.: ДМК Пресс, 2017. - 396 с.

20) Weston, J. Multiclass support vector machines [Text] / J. Weston, M. Verleysen, C. Watkins - Proceedings of ESANN99, D. Facto Press, Brussels. 1999 – pp. 650.

21) Державні санітарни норми і правила. ДСанПіН 3.3.2.007-98

«Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

22) Державні санітарни норми України. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

23) Державні санітарни норми України. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va037282-99

24) Державні санітарни норми України. ДСН 3.3.6.039-99 «Санітарні норми виробничої загальної та локальної вібрації» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va039282-99

25) Державний стандарт України. ГОСТ 13109-97 «Электрическая энергия. Совместимость технических средств электромагнитных. Нормы качества электроэнергоснабжения общего назначения» Режим доступу: WWW. URL: http://odz.gov.ua/lean_pro/standardization/files/elektromagnitnaja_sovmestimost_2014_03_11_1.pdf

26) Державні будівельні норми України. ДБН В.2.5-28:2018 «Природнє і штучне освітлення» Режим доступу: WWW. URL: https://okna.ua/img_all/oknaua/dbn-V-2-5-28-2018-ed.pdf

27) Нормативно-правові акти з охорони праці. НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/laws/show/z0093-98

28) Державні будівельні норми України. ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» Режим доступу: WWW. URL: https://dnaop.com/html/32609/doc-%D0%94%D0%91%D0%9D_%D0%92.2.5-67_2013

29) Державний стандарт України. ГОСТ 12.1.044-89 «ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и

методы их определения» Режим доступу: WWW. URL: http://online.budstandart.com/ru/catalog/doc-page?id_doc=51048

30) НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями Міністерство доходів і зборів України Наказ від 05.09.2013 р. № 443 «Про затвердження Примірної інструкції з охорони праці під час експлуатації електронно-обчислювальних машин» Режим доступу: WWW. URL: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php

31) Нормативно-правові акти з охорони праці. НПАОП 0.00-4.15-98 «Про розробку інструкцій з охорони праці» Режим доступу: WWW. URL: http://sop.zp.ua/norm_npaop_0_00-4_15-98_01_ru.php

Додаток А.

Комп’ютерна презентація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

ДИПЛОМНА РОБОТА БАКАЛАВРА

Методи автоматизації аналізу тестування веб-додатків

Науковий керівник:
проф. Рязанцев О.І.

Виконав:
студент групи КН-16бз Волчанський О. Є.

Сєвєродонецьк, 2020

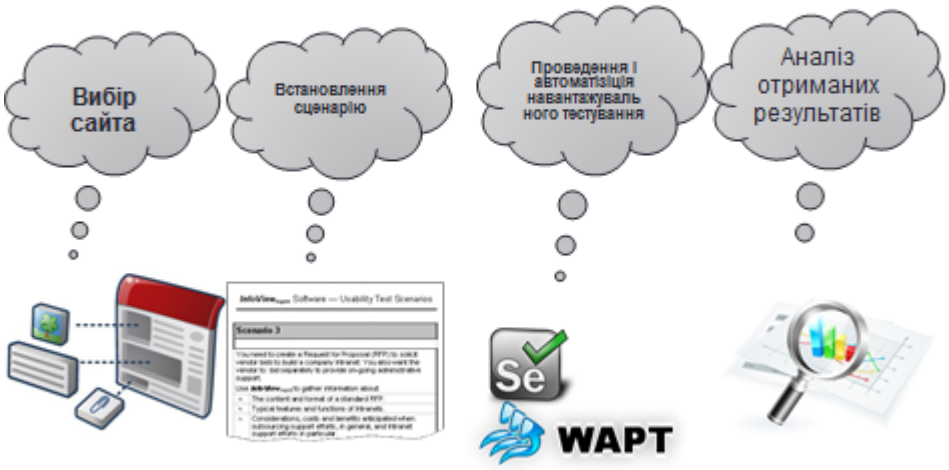
Актуальність

Работоспособность сайта стала критически важной для большинства современных компаний, а цена просчета в оценке его реальной производительности значительно увеличилась. Все это делает актуальным нагрузочное тестирование.

Главное задание нагрузочного тестирования – обеспечить надлежащую стрессоустойчивость веб-приложения во время чрезмерной нагрузки его пользователями.

Аналіз даних, отриманих на основі тестування, проведеного за допомогою сервісу, є найбільш зручним і швидким способом для отримання відомостей про якість веб додатки, що дозволяє в подальшому виправити помилки допущені при розробці сайту і поліпшити його функціонування при непередбачуваних ситуаціях.

Постановка задачі



Формальна постановка задачі

- Критерій оцінки стійкості сайту:

$$\Delta t^{avg}=1\backslash N \mid t^{avg}(n_1)-t^{avg}(n_2)\mid$$

$N=(10,100,700)$, $|N|=3$.

k-коефіцієнт кореляції між числом користувачів і часом завантаження сторінки;

t^{avg} -функція залежності середнього часу;

N - кількість користувачів.

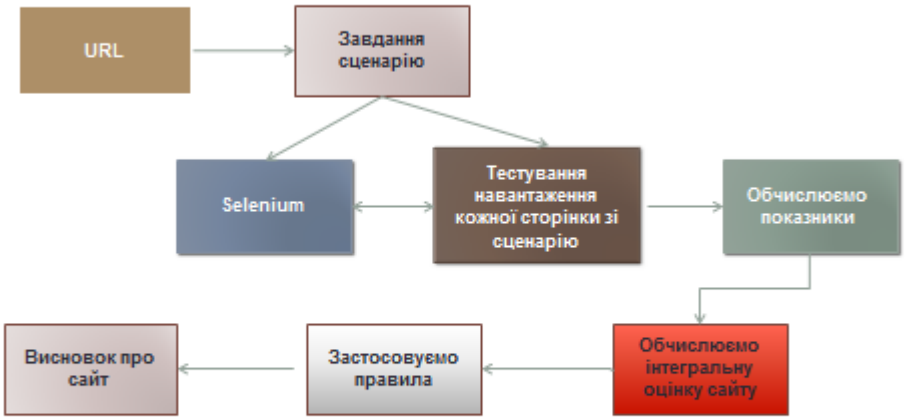
Число користувачів, для для яких час завантаження сторінки не перевищує необхідне:

$$n_0=\{n \in N: \ t^{avg}(n_0)<t^{треб.}\}.$$

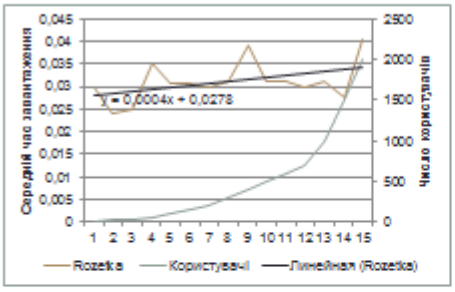
- Узагальнена оцінка сайту : $z=\frac{1}{n}\sum_{i=0}^{n}\Delta\tilde{t}_i^{avg}/w, i=1,\ldots,n\},$

де w – досяжність сторінки з головної сторінки сайту.

Метод вирішення

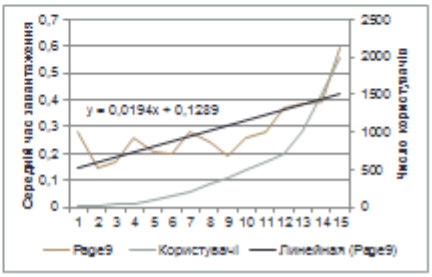


Аналіз результатів

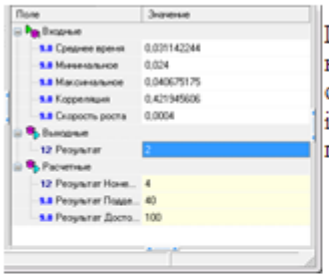


Даний графік відображає зміни середнього часу завантаження 1-ої сторінки зі збільшенням кількості користувачів. Можна зробити висновок, що сторінка є досить стійкою до навантаження, оскільки при відвідуванні все більшої кількості користувачів час практично не зростає.

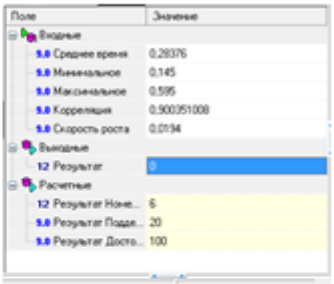
Графік показує зміну середнього часу завантаження 9-ої сторінки зі збільшенням кількості користувачів. Дана сторінка є нестійкою до навантаження, виходячи з того, що при збільшенні кількості користувачів зростає і середній час завантаження сторінки.



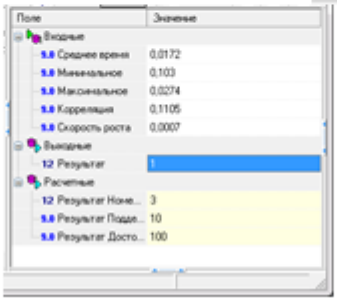
Аналіз результатів



Швидкість роботи сайту на першій сторінці була оцінена як досить висока і тому їй був призначений пріоритет 2.



При обчисленні інтегральної оцінки сайту, були отримані результати, які показали, що в середньому сайт справляється з навантаженням на середньому рівні, тому був призначений пріоритет 1.



Швидкість роботи сайту на дев'ятій сторінці була оцінена як дуже погана і тому їй був призначений пріоритет 0.

ВИСНОВКИ

У даній дипломній роботі було протестовано кілька веб-додатків за допомогою імітації навантажувального тестування. Також був проведений аналіз даних, отриманих після тестування, які були представлені у вигляді графіків, що наочно продемонструвало всі етапи роботи веб-додатків при навантаженні їх користувачами. В результаті було оцінено якість сайтів по їх основних характеристик, що дозволило зробити висновки щодо їх працездатності під час критичних ситуацій.

Надалі буде введена можливість автоматизації висновків про сайт, а також рекомендацій щодо поліпшення його функціонування.

Під час розробки питань, що стосуються охорони праці була прорахована можливість спрацьовування пристрою максимального захисту від струму, яке забезпечувало б надійне відключення споживачів електроенергії для запобігання виникнення короткого замикання в мережі.

ДЯКУЮ ЗА УВАГУ!