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
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Аналіз навантажувального тестування інформаційної системи

Освітній ступінь "Магістр"
Спеціальність 122 "Комп'ютерні науки "

Науковий керівник роботи: _____ (підпис) Д.О.Недзельський (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О.Критська (ініціали, прізвище)

Студент: _____ (підпис) О.М.Торопов (ініціали, прізвище)

Група: КН-17зм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній ступінь магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Торопову Олексію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз навантажувального тестування інформаційної системи

керівник проекту (роботи) Недзельський Дмитро Олександрович, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «18» 10 2018 р. № 221/48
2. Строк подання студентом роботи 10.01.2018
3. Вихідні дані до роботи Матеріали науково-дослідної практики, тестові зображення результатів тестування Web-додатків, теоретичні відомості про види навантажувального тестування, середа розробки MATLAB

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Опис предметної області, автоматизація навантажувального тестування програмного забезпечення, методи аналізу результатів навантажувального тестування програмного забезпечення, загальна схема запропонованого методу аналізу web-застосунку, програмна реалізація автоматичної обробки результатів навантажувального тестування web-додатків, охорона праці та безпека в надзвичайних ситуаціях, 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 та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 18.10.2018

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	10.09.2018-15.09.2018	
2	Аналіз завдання, робота з літературою	16.09.2018-22.09.2018	
3	Проведення аналітичних досліджень та розрахунків	26.09.2018-06.10.2018	
4	Робота над текстом пояснювальної записки	07.10.2018-25.11.2018	
5	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	26.11.2018-1.12.2018	
6	Оформлення пояснювальної записки, автореферату та презентації	2.12.2018-09.01.2019	
7			

Студент _____ Торопов О.М.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Недзельський Д.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Торопов О.М. Аналіз навантажувального тестування інформаційної системи.

У магістерській роботі проведено аналіз навантажувального тестування, з метою оцінки ефективності інформаційної системи. Досліджено основні методи автоматизації обробки та аналізу результатів навантажувального тестування, а так само запропоновано аналітичні методи оцінки характеристик продуктивності сайту. Представлена база правил для автоматизованного аналізу, на основі яких можна ставити оцінки для системі. Зроблені висновки, на основі нечіткої логіки, про результативність розглянутих систем.

Ключові слова: результати, тестування програмного забезпечення, помилка, аналіз, якість, навантажувальне тестування, правила, обробка.

ABSTRACT

Toropov O.M. Analysis of load testing of the information system.

In the master's work the analysis of load testing was conducted, in order to evaluate the effectiveness of the information system. The basic methods of automation of processing and analysis of the results of load testing are investigated, as well as analytical methods for assessing the performance of a site are proposed. The rule base for automated analysis is presented, on the basis of which it is possible to set estimates for the system. The conclusions are drawn, based on fuzzy logic, on the effectiveness of the systems under consideration.

Keywords: results, software testing, error, analysis, quality, load testing, rules, processing.

АННОТАЦИЯ

Торопов А.Н. Анализ нагрузочного тестирования информационной системы.

В магистерской работе рассмотрена задача проведения нагрузочного тестирования, с целью оценки эффективности информационной системы. Изучены основные графические методы анализа результатов нагрузочного тестирования, а так же предложены аналитические методы оценки характеристик производительности сайта. Представлена база правил для автоматического анализа, на основе которых можно оценивать систему. Сделаны выводы, на основе нечеткой логики, о результативности рассмотренных систем.

Ключевые слова: результаты, тестирование программного обеспечения, ошибка, анализ, качество, нагрузочное тестирование, правила, обработка.

ЗМІСТ

ВСТУП

На сьогоднішній день, створення високоякісного програмного забезпечення, є одним з найважливіших завдань розвитку науки та виробництва. На питання про якість програмного забезпечення (ПЗ) відповісти може тільки його експлуатація.

Дуже часто, замовник розраховує на високу якість, але після великої кількості невдалих спроб отримати замовлене ПЗ високої якості викликає у замовників законні питання: а чи можливо в принципі створювати якісне ПЗ? як цього домогтися? що повинен вимагати замовник від підрядника, щоб отримати якісне ПЗ? як повинен працювати сам замовник?

В зв'язку з тим, що тільки деякі суттєві властивості програмного забезпечення можуть бути виміряні безпосередньо і оцінені кількісними показниками, якість комп'ютерних технологій набуває першочергового значення.

Для оцінки якості ПЗ завжди застосовується цілий комплекс заходів, серед яких тестування ПЗ на предмет виявлення помилок - один з найважливіших етапів.

Спільний погляд на тестування програмного забезпечення останні роки активно еволюціонував, стаючи усе більш конструктивним, прагматичним та наближеним до реалій сучасних проектів розробки програмних систем. Тестування більше не розглядається як діяльність, що починається тільки після завершення фази конструювання. Сьогодні тестування розглядається як діяльність, яку необхідно здійснювати протягом усього процесу розробки й супроводу і являється важливою частиною конструювання програмних продуктів.

Відповідно до формулювання ISO 8402, під якістю розуміється сукупність характеристик програмного продукту, що стосуються його здатності задовольнити встановлені й передбачувані потреби клієнтів. Основними параметрами якості вважаються: функціональна повнота, відповідність вимогам законодавства, безпека інформації, простота експлуатації, не потребуючих спеціальних знань в області інформаційних технологій, ергономічність користувальницького інтерфейсу, мінімізація витрат на експлуатацію, розвиток і модернізацію.

Під надійністю звичайно розуміється здатність системи виконувати задані функції, зберігаючи основні характеристики за певних умов експлуатації. Стосовно до програмного забезпечення - це насамперед безвідмовна робота, відсутність помилок, що перешкоджають нормальному функціонуванню підприємства [1].

Якість і надійність у комплексі забезпечують високі споживчі властивості ПЗ. У процесі створення програмного продукту вони одночасно й безупинно контролюються й удосконалюються. Однак наскільки реально забезпечити якість і надійність складної багатофункціональної системи при обмежених термінах розробки? Для ілюстрації можна привести результати опитування більше тисячі великих компаній, проведених міністерством торгівлі й промисловості Великобританії. Виявилося, що середня частота відмов інформаційних систем склала: 1 відмову в рік - 40% компаній, 1 відмова на місяць - 29%, 1 відмова в тиждень - 15% компаній, 1 відмова в день - 7% й 5% компаній спостерігали в себе більше однієї відмови в день. При цьому доля відмов і збоїв програмного забезпечення в загальному списку причин непрацездатності (простоїв) інформаційних систем становила 24%. Один із ключових елементів забезпечення якості - це тестування [2].

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Технології створення програмного забезпечення, за останні роки, стали базою для різних розділів комп'ютерних наук, як засіб подолання складностей при розробці якісного продукту. Тестування ПЗ найбільш дорогий та важливий етап життєвого циклу, на нього відводиться більше половини загальних витрат.

Сучасне суспільство навряд чи можна уявити без інформаційних технологій. Перспективи їх розвитку на сьогодні дуже стрімкі.

1.1 Огляд основних видів тестування

Види та методи тестування ПЗ залежать від стадії проектування, на якій починається перевірка правильності функціонування результатів проектування.

1) Тестування специфікації.

Специфікація є основним документом, виходячи з якого формуються вимоги для тестування. Специфікація відповідає на питання коли і як повинен працювати сам додаток, як система або її модуль повинен реагувати на не коректні дані або невірну поведінку користувача, що повинне бути в результаті правильного відпрацьовування, при яких умовах і вхідних даних правильне відпрацьовування повинно мати місце, що повинне бути в результаті не правильного відпрацьовування додатка, що тестується, при яких умовах вона повинна мати місце. На всі ці питання повинні бути відповіді в документації. Інакше документація вважається неповною, що рівняється помилці в документації. Тестування специфікації це повноцінний вид тестування, основним показником успішності завершення якого є досягнення наступних критеріїв якості вимог:

- коректність;
- недвозначність;
- повнота;
- несуперечність;
- упорядкованість по важливості й стабільності;
- перевіряємість (веріфікованість або тестопригодність);
- модифікованість;
- трасованість;

– зрозумілість.

Тестувальник, що працює з документацією, відповідає за технічну точність кожного її слова. Він зобов'язаний зробити саму ретельну перевірку її відповідності реальній структурі й поводженню програми. Необхідно звертати увагу на складні й заплутані місця тексту. Вони можуть відображати невдало спроектовані елементи самої програми.

2) Модульне тестування.

Цей рівень тестування дозволяє перевірити функціонування окремо взятого елемента системи. Що вважати елементом - модулем системи визначається контекстом.

Юніт-тести дозволяють перевірити, чи дійсно окрема частина коду виконує саме ті дії, яких очікує розроблювач, і не робить нічого іншого, для чого тестуються дуже маленькі, ізольовані частини коду. При цьому залишається відкритим питання, чи відповідає результат виконання коду очікуванням клієнтів або кінцевих користувачів.

3) Інтеграційне тестування.

Даний рівень тестування є процесом перевірки взаємодії між програмними компонентами/модулями. Класичні стратегії інтеграційного тестування - “зверху” й “знизу” - використовуються для традиційних, ієрархічно структурованих систем й їх складно застосовувати, наприклад, до тестування слабозв'язаних систем, побудованих у сервісно-орієнтованих архітектурі (SOA). Сучасні стратегії більшою мірою залежать від архітектури тестової системи й будуються на основі ідентифікації функціональних “потоків” (наприклад, потоків операцій і даних). Інтеграційне тестування - постійно проведена діяльність, що припускає роботу на досить високому рівні абстракції. Найбільш успішна практика інтеграційного тестування базується на інкрементальному підході, що дозволяє уникнути проблем проведення разових тестів, пов'язаних з тестуванням результатів чергового тривалого етапу робіт, коли кількість виявлених дефектів приводить до серйозної переробки коду (традиційно, негативний досвід випуску й тестування тільки великих релізів називають “big bang”).

4) Функціональне тестування.

Функціональне тестування об'єкта тестування планується й проводиться на основі вимог до тестування, заданих на етапі визначення вимог. В якості вимог виступають бізнес-правила, діаграми use-case, бізнес-функції, а також при наявності, діаграми активності. Мета функціональних тестів полягає у тому, щоб перевірити відповідність розроблених графічних компонентів установленим вимогам.

Необхідно виконати набір тестів, використовуючи як вірні значення, так і свідомо помилкові, для підтвердження правильного функціонування, за наступними критеріями:

– продукт адекватно реагує на всі вхідні дані (виводяться очікувані результати у відповідь на дані, що вводять правильно);

– продукт адекватно реагує на неправильні вхідні дані (з'являються відповідні повідомлення про помилки)[4].

5) Навантажувальне тестування, тестування продуктивності.

Тестування продуктивності - спеціалізовані тести перевірки задоволення специфічних вимог, що пред'являються до параметрів продуктивності. Існує особливий підвид таких тестів, коли робиться спроба досягнення кількісних меж, обумовлені характеристиками самої системи і її операційного оточення. Як критерії для проведення таких тестів, є вимоги до програмного продукту, висунуті в його специфікації. Однією із цілей виконання аналізу продуктивності є її збільшення. Якщо виявити модулі, які виконуються частіше або довше інших, їх можна переробити й тим самим підвищити швидкість роботи всієї системи.

При навантажувальних іспитах (load testing) перевіряється реакція програми на граничні умови експлуатації - тестування на максимальний обсяг вхідних даних, випробування у тяжкому режимі (різке збільшення навантаження).

6) Конфігураційне тестування.

У випадках, якщо програмне забезпечення створюється для використання різними користувачами, даний вид тестування спрямований на перевірку поведінки та працездатності системи в різних конфігураціях.

7) Тестування зручності й простоти використання.

Ціль даного типу тестування - перевірити, наскільки легко кінцевий користувач системи може її освоїти, включаючи не тільки функціональну складову - саму систему, але і її документацію; наскільки ефективно користувач може виконувати завдання, автоматизація яких здійснюється з використанням даної системи; нарешті, наскільки добре система застрахована (з погляду потенційних збоїв) від помилок користувача. Для повноти даного виду тестування й виключенні людського фактора оцінюючого рекомендується використати заздалегідь складені й стандартизовані оцінні форми. Вони можуть собою представляти список вимог, що висуваються тестованою системою.

8) Регресійне тестування.

Це вид тестування, що проводиться при перевірці нової версії програми. Набір регресійних тестів найчастіше автоматизований.

Визначення успішності регресійних тестів (IEEE 610-90 "Standard Glossary of Software Engineering Terminology") говорить: "повторне вибіркове тестування системи або компонент для перевірки зроблених модифікацій не повинне приводити до

непередбачуваних ефектів". На практиці це означає, що якщо система успішно проходила тести до внесення модифікацій, вона повинна їх проходити й після внесення таких.

9) Альфа- і бета-тестування (Alpha and beta testing).

Перед тим, як випускається програмне забезпечення, як мінімум, воно повинне проходити стадії альфа (внутрішнє пробне використання) і бета (пробне використання із залученням відібраних зовнішніх користувачів) версій. Звіти про помилки, що надходять від користувачів цих версій продукту, обробляються відповідно до певних процедур, що включають підтверджуючі тести (будь-якого рівня), проведені фахівцями групи розробки[6].

1.2 Основні інструменти навантажувального тестування

За допомогою навантажувального тестування, можна швидко та якісно протестувати основний та важливий функціонал програми. Використовується у випадках, коли необхідно збільшити число користувачів або перевірити межу масштабованості системи, знайти потенційно вузьке місце системи з огляду продуктивності [1].

1.2.1 Навантажувальний програмний продукт HP Load Runner

LoadRunner - один з найпотужніших інструментів для тестування являє собою потужний інструмент для тестування великих інформаційних систем, але його використання для невеликих Web-серверів і баз даних навряд чи буде виправданим та доцільним, з огляду на високу вартість.

Інструментарій LoadRunner (рис. 1.1) компанії Mercury Interactive призначений для аналізу продуктивності інформаційних систем рівня підприємства. Цим продиктована наявність ряду особливостей, відсутніх у вже розглянутих систем. Так, LoadRunner підтримує тестування великої кількості мережних інфраструктур й їхніх компонентів, включаючи Web-сервери, бази даних, компоненти COM, JavaBeans, віртуальні машини Java та ін. Це робить LoadRunner коштовним інструментом для аналізу роботи гетерогенних мережних середовищ, типових для більшості великих інформаційних

систем. Важливою властивістю з погляду аналізу роботи бізнесів-додатків є й можливість відстеження виконання транзакцій.

Ще одна якість, необхідне для системи тестування такого класу, - масштабованість для створення високих рівнів навантаження, що в LoadRunner досягається шляхом використання розподілених обчислень, а також технології TurboLoad, мінімізує використання центрального процесора тестуючого комп'ютера кожним з віртуальних користувачів. Завдяки цьому система з 10 високопродуктивних серверів може формувати навантаження, еквівалентним декільком сотням тисяч користувачів або мільйонам запитів у день.

Тестування Web-застосунків, як й у більшості інших систем аналогічного призначення, засновано на використанні сценаріїв, що описують послідовність дій віртуальних користувачів. Підтримуються з'єднання з використанням SSL. Сценарії описуються у форматі XML і можуть бути створені з використанням браузера, а також за допомогою спеціальної утиліти Virtual User Generator.

Підвищення реалістичності тестування досягається шляхом підтримки аутентифікації віртуальних користувачів, параметризації тестових сценаріїв, використання випадкових інтервалів очікування між запитами, імітації різних швидкостей з'єднання, емуляції кешування даних браузером, а також імітації інших особливостей роботи клієнта.

Збір і подання результатів тестування в реальному часі здійснюються за допомогою моніторів. Передбачено монітори для визначення параметрів продуктивності обчислювальної мережі підприємства в цілому, окремих мережних пристроїв, операційних систем, а також розповсюджених мережних додатків: IIS, Apache, iPlanet, Oracle, DB2, SQLServer й ін. LoadRunner надає можливість вимірювати безліч параметрів (у тому числі, кількість віртуальних користувачів, число оброблюваних запитів у секунду, час відповіді, кількість завершених аварійно транзакцій).

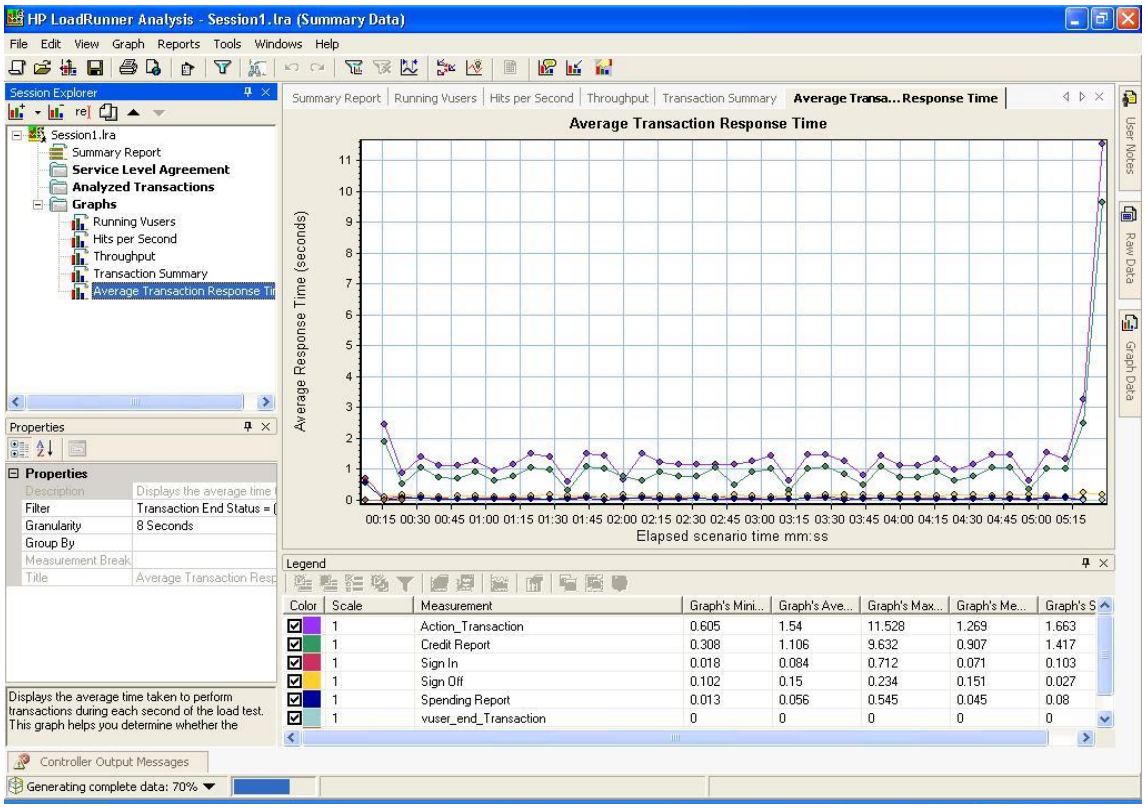


Рисунок 1.1 — Інтерфейс програми HP Load Runner

1.2.2 Програмний продукт SilkPerformer

SilkPerformer - платний інструмент для атоматизованого навантажувального тестування Web-застосунків різного рівня складності. Даний інструмент являється потужним та легким у використанні для навантаження та стрес-тестування корпоративного класу. Візуальний сценарій та можливість тестування декілька програм з тисячами одночасних віртуальних користувачів дозволяє ретельно перевірити корпоративне ПЗ та надійність, продуктивність та масштабування, перш ніж вони будуть встановлені, незалежно від їх розміру та складності.

Ключовими перевагами даного інструмента являються: значно знижені витрати на дефекти в багаторівневих корпоративних додатках шляхом тестування функціональності, сумісності та продуктивності віддалених компонентів на початку циклу розробки, ще до побудови клієнтських додатків. Можна швидко генерувати навантажувальні скрипти для Web -служб, .NET, EJB і Java RMI об'єктів. Крім того, є можливість використовувати модульне тестування в середовищах Java і .NET.

Технологія TrueLog для HTML, XML, SQL, Oracle Forms, Citrix, TCP / IP, UDP і даних на основі протоколу забезпечує повний візуальний аналіз першопричин з точки зору кінцевих користувачів. TrueLogs візуально відтворює дані, які користувачі отримують протягом навантажувальних тестів. Для HTML-сторінок включаються всі впроваджені об'єкти. Це дозволяє візуально аналізувати поведінку програми, як поводяться помилки в ході випробувань. Крім того, присутній докладна статистика часу відгуку як окремих запитів, так і цілих транзакцій.

Навантажувальні скрипти в SilkPerformer (рис. 1.2) описані вбудованою мовою BDL (Benchmark Description Language). Алгоритмічна мова, має весь необхідний набір констант і методів для виконання будь-яких завдань, які можуть виникнути на шляху до створення необхідного навантажувального сценарію. Ознайомитися з мовою можна прямо з довідки SilkPerformer. Мова інтуїтивно зрозуміла та нескладна.

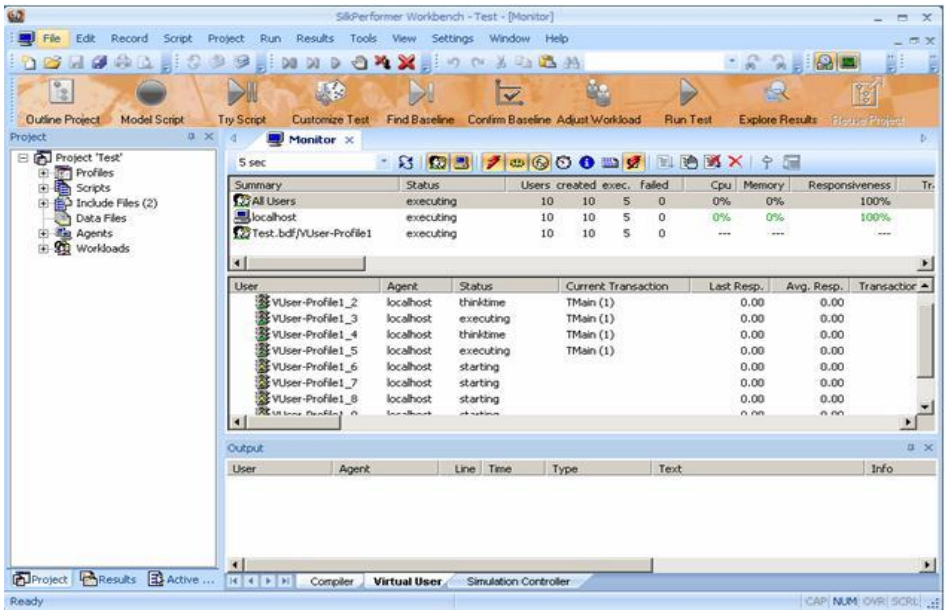


Рисунок 1.2 — Інтерфейс програми SilkPerformer

1.2.3 Програмний продукт Selenium

Selenium - це об'єктно-орієнтований JavaScript додаток, що 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 й Mozilla Firefox для Microsoft Windows, Linux й Apple Macintosh.

У рамках проекту Selenium також випускається інструмент Selenium IDE, що представляє 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 й відтворювати скрипти, що представляють собою звичайні HTML-сторінки з однією таблицею, що містить команди.

В IDE існує велика кількість варіантів запуску тестового сценарію. Ви можете виконати тестовий сценарій цілком, зупинити і відновити його, виконати в покроковому режимі, виконати тільки одну команду, над якою зараз йде робота, а також запустити весь набір тестів. Selenium IDE (рис.1.3) дозволяє дуже гнучко працювати з тестовими сценаріями.

Перевагами Selenium є:

– кросплатформове й кросбраузерне тестування;
– тести Selenium виконуються безпосередньо в браузері;
– виконання довільного JavaScript-коду на сторінці тестованого додатка;
– можливість «накликати» тест (запис дій користувача в браузері);
– доступність (безкоштовне поширення).

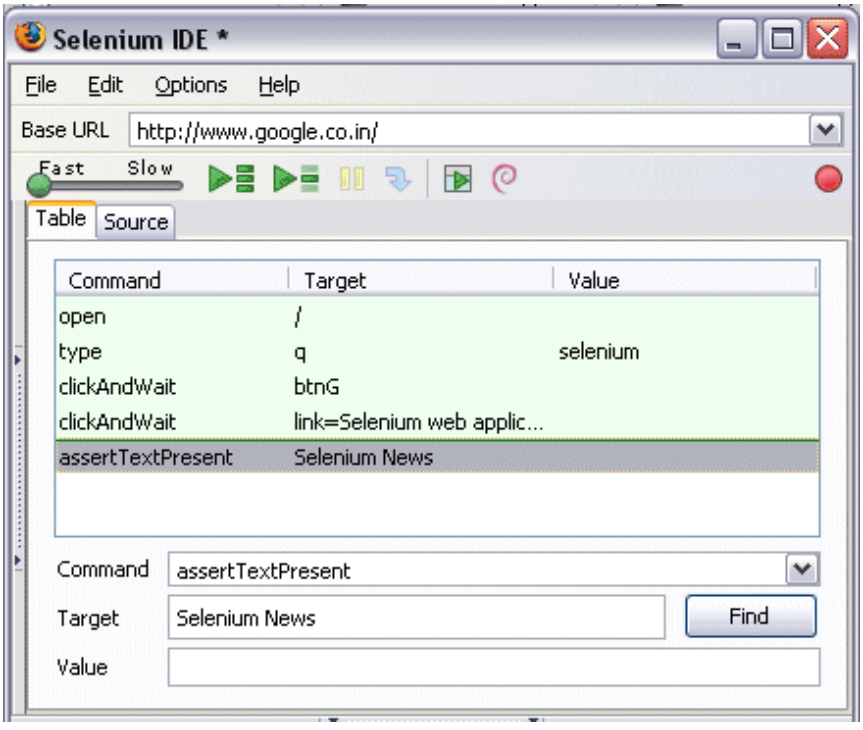


Рисунок 1.3 — Інтерфейс програми SeleniumIDE

1.2.4 Програмний продукт Rational Performance Tester

Rational Performance Tester - це інструмент, розроблений для тестування Web-зазстосунків з метою виявлення та виправлення проблем з продуктивністю до розгортання. Rational Performance Tester допомагає точно визначати вузькі місця системи до її розгортання за допомогою емуляції одночасної роботи заданого числа користувачів і генерування звітів, які чітко визначають погано функціонуючі Web-сторінки, URL і транзакції.

Функції високого рівня включають в себе детальне планування тестування на рівні активності користувача і моделі використання кожної з їх груп. Rational Performance Tester (рис. 1.4) також надає можливість автоматичної організації пулу даних, що дозволяє змінювати набір тестових даних, які використовуються кожним змодельованим користувачем. За допомогою вікна, що нагадує браузер, інтегрованого з редактором тестів, ви можете переглянути Web-сторінки, до яких здійснюється доступ під час запису тесту. Крім цього, досвідчені тестувальники можуть скористатися опцією вставки користувальницького Java-коду в тести продуктивності для виконання таких дій, як розширений аналіз даних та синтаксичний аналіз запиту.

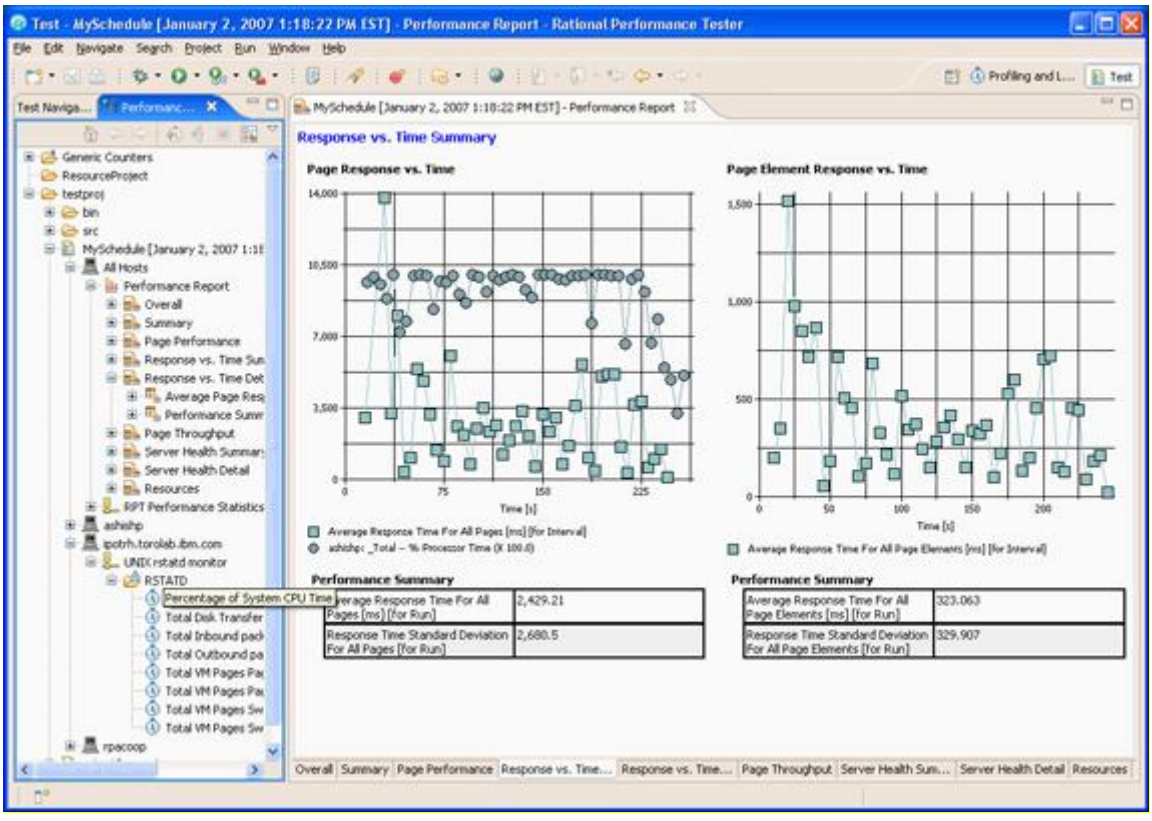


Рисунок 1.4 — Інтерфейс програми Rational Performance Tester

1.3 Постановка завдання дослідження

Навантажувальне тестування дозволяє нам виявити та прослідкувати поведінку системи при різних навантаженнях. Дуже багато систем, на сьогодні, можуть перевірити вашу систему але жодна з них не зробить автоматичний аналіз та діагностику, на основі отриманих даних. Системи, які вже існують та використовуються багатьма людьми, не дуже ефективні, ви повинні самі вказати сценарій або дані за якими тестувати. Дана система автоматизованого тестування зможе сама проаналізувати сайт та на основі оброблених даних зробити висновок, наскільки веб-система швидка та продуктивна.

Виходячи із проведеного системного аналізу системи - «Тестування програмного забезпечення», а також аналізу проблеми автоматизації оцінки результатів тестування, сформулюємо завдання дослідження даної дипломної роботи:

– сформулювати завдання автоматизації оцінки результатів тестування;

– розробити метод рішення завдання автоматизації оцінки результатів тестування;

– розробити програмний продукт, що автоматизує оцінку результатів тестування;

– провести аналіз ефективності запропонованого методу автоматизації оцінки результатів тестування;

– розробити й впровадити інформаційно-аналітичну систему, що дозволила автоматизувати оцінку результатів тестування web-застосунків.

2 АВТОМАТИЗАЦІЯ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Більшість програмних продуктів, що випускаються сьогодні, є веб-орієнтованими програмами, розрахованими на роботу в інтернет-браузері. Ефективність тестування подібних додатків відрізняється в різних компаніях та організаціях. В епоху високої інтерактивності і взаємодії в процесі розробки програм, коли багато організацій використовують методологію Agile в тій чи іншій формі, автоматизація тестування часто стає необхідністю. Під автоматизацією навантажувального тестування мається на увазі використання інструментів для того, щоб багаторазово виконувати повторювані тести для програми, що тестується. Регресійне тестування є найбільш типовим прикладом застосування цього підходу.

Автоматизоване тестування володіє безліччю достоїнств, пов'язаних головним чином з високою швидкістю виконання тестів і можливістю виконувати однотипні тести знову і знову.

2.1 Основні принципи навантажувального тестування інформаційних систем

Основними принципами навантажувального тестування являються: оцінка продуктивності та працездатності

Навантажувальне тестування, це тестування, у ході якого перевіряється продуктивність і працездатність системи при різному навантаженні й при різних системних ресурсах. Як мета тестування висувається перевірка на відповідність характеристик системи значенням, необхідним специфікацією, загальноприйнятим нормам. Для навантаження системи при тестуванні використають різні вхідні дані й варіанти тестового оточення. Властиво навантажувальне тестування може бути розділене на кілька типів [3].

1) Тестування продуктивності.

Тестування продуктивності виконується з метою визначити тимчасові характеристики системи при заданому робочому навантаженні. Тестування продуктивності має кілька цілей. Воно показує, що система відповідає критеріям

продуктивності; указує на частини, що мають найменшу продуктивність; може продемонструвати переваги даної системи над аналогічною. При розробці систем, критичних до продуктивності, даний вид тестування стартує на самому початку розробки й виробляється аж до розгортання системи.

2) Тестування одночасного доступу до даних (Contention Tests)

Цей вид тестування націлений на знаходження вузьких місць у продуктивності (блокування, витоку пам'яті, «пробуксовки» у віртуальній пам'яті), викликаних певною кількістю віртуальних користувачів, що звертаються до тим самим ресурсів [6].

Тест визначає, який максимальний сплеск у часі виконання транзакцій може витримати додаток без помилок. Виміряються мінімальний, середній і максимальний час на кожну дію.

3) Тестування навантаження.

При навантажувальному тестуванні виробляється перевірка працездатності системи при зміні навантаження на систему: кількості користувачів, транзакцій і т.п. Дозволяє визначити границі, при яких характеристики системи залишаються на прийнятному рівні, що дозволяє продовжувати роботу з нею.

4) Тестування стійкості (Endurance "Soak" "Longevity" Tests)

Цей вид навантажувального тестування перевіряє, чи може система підтримувати постійну кількість віртуальних користувачів, що виконують транзакції, максимально використовуючи пропускну здатність системи.

Даний тест дозволяє виявити «накопичувальні» помилки в коді, а також трапляються час від часу події й аномалії. Також перевіряються події, які спрацьовують за розкладом [5].

Однією із цілей тестування є визначення схильності системи до деградації (наприклад, наявність витоків пам'яті). Для цього система залишається під навантаженням на тривалий час, а потім перевіряється, чи змінилися її тимчасові характеристики, розміри, займані в пам'яті.

5) Стресове тестування.

Ціль стресового тестування визначити, при яких навантаженнях системи відбувається її відмова, причини цієї відмови, поводження системи в цьому випадку, коректність збереження й обробки даних. Існує кілька підходів до реалізації стресового тестування - це збільшення навантаження на систему аж до відмови й урізування ресурсів системи (оперативної пам'яті, дискового простору, процесорних ресурсів). Можна комбінувати дані підходи для якнайшвидшого досягнення критичної крапки.

Стресове тестування дозволяє визначити мінімальні величини ресурсів, необхідних для роботи додатка, оцінити граничні можливості системи і виявити фактори, що обмежують ці можливості (тобто знайти «вузькі місця» системи). Крім того, метою стресового тестування може бути виявлення помилок у серверному додатку, а також тестування здатності системи до збереження цілісності даних в аварійних ситуаціях і до відновлення після таких ситуацій.

6) Об'ємне тестування.

Об'ємне тестування, або тестування обсягу - вид тестування, при якому на обробку в систему подаються більші обсяги даних і вивчається поводження системи при цьому. Наприклад, обробка більших файлів у текстовому, більших зображень у графічному редакторі.

7) Тестування масштабованості.

Цей вид навантажувального тестування має на увазі повторення попередніх видів тестів на різних конфігураціях серверного/мережного встаткування з метою визначення найефективнішого за ціною варіанта, що буде задовольняти поставленим вимогам продуктивності.

2.2 Характеристики навантажувального тестування

Навантажувальне тестування дозволяє оцінити безліч різних характеристик, що ставляться до сценарію тесту, до web-застосунка, до серверних і мережних характеристик системи, тестується. Представимо всі групи характеристик й їхній перелік у вигляді кваліметричної моделі властивостей інформаційної системи (рис. 2.1).

Під час виконання сценарію віртуальні користувачі (далі, VU - Virtual Users) генерують дані, начебто вони виконують транзакції. Характеристики сценарію дозволяють визначити загальну поведінку VU протягом сценарію:

1) відносний час (Relative Time) - тимчасова шкала, у якій за нуль прийнятий час початку тесту;

2) кількість віртуальних користувачів, загальний або в певний час тесту (активних, що очікують, зупинених, зупинених з помилкою).

1) Середній час завантаження сторінки (усього контента).

Даний показник характеризує медіанне значення тимчасового ряду, складений із середнього часу завантаження сторінки протягом навантажувального тесту. Протягом

тесту при різних кількостях користувачів, що підключають, обчислюємо середній час завантаження сторінки, для всіх користувачів, що одночасно перебувають на сайті в цей час. Таким чином, становимо часовий ряд, а далі для зазначеного ряду обчислюємо медіанне значення, отримане значення і є значення метрики.

Для даної метрики діє критерій: чим менше значення цього показника, тим краще. Виходячи із цього введена лінгвістична змінна, котра містить три терма:

– - відмінно (час завантаження сторінки перебуває в діапазоні 0 до 3 секунд);

– - нормально (час завантаження сторінки перебуває в діапазоні від 2 до 10 секунд);

– - погано (коли час завантаження сторінки перевищує 6 секунд).

Для відповідних термів визначені види функцій приналежності й у такий спосіб задана лінгвістична змінна середній час завантаження сторінки, представлена на рисунку 2.1.

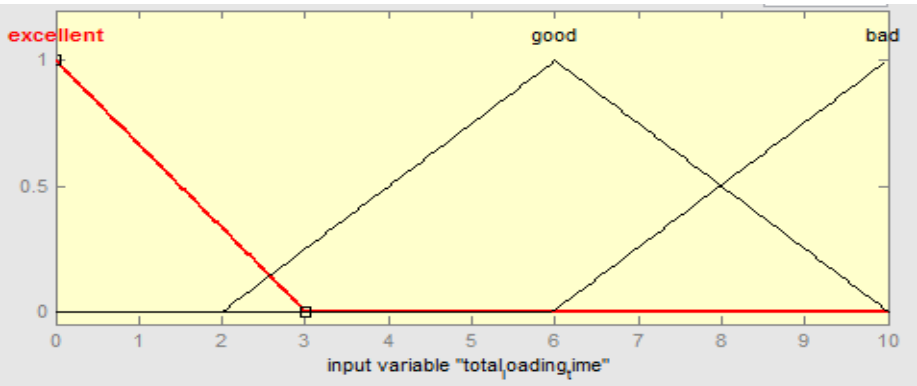


Рисунок 2.1 - Вид функцій приналежності для змінної «середній час завантаження сторінки»

2) Середній час завантаження html коду сторінки.

Ці характеристики подають інформацію про використання ресурсів Web-серверів Apache, Microsoft IIS, iPlanet/Netscape, and iPlanet (SNMP), а також серверів баз даних, серверів ERP/CRM і серверних додатків.

Лінгвістична змінна даної метрики, що містить три терма:

– - відмінно (час завантаження сторінки перебуває в діапазоні 0 до 2 секунд);

– - нормально (час завантаження сторінки перебуває в діапазоні від 1 до 5 секунд);

– - погано (коли час завантаження сторінки перевищує 3 секунди).

Для відповідних термів визначені види функцій приналежності й у такий спосіб задана лінгвістична змінна середній час завантаження сторінки, представлена на рисунку 2.2.

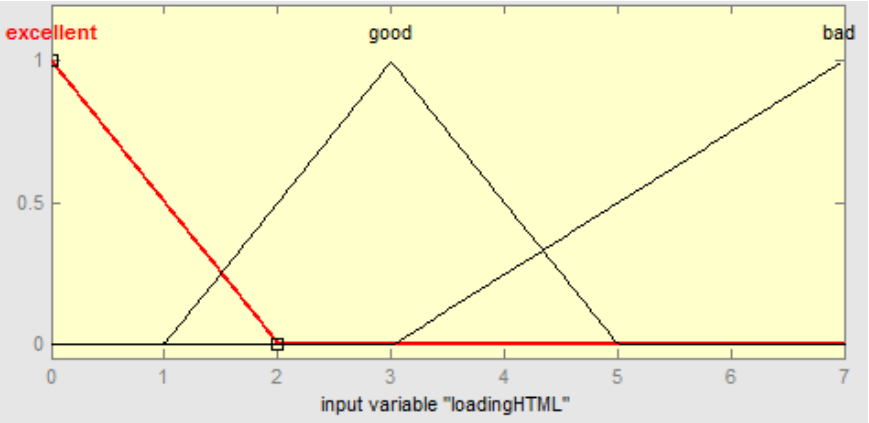


Рисунок 2.2 - Вид функцій приналежності для змінної «середній час завантаження html коду сторінки»

3) Максимальний відносний час виконання.

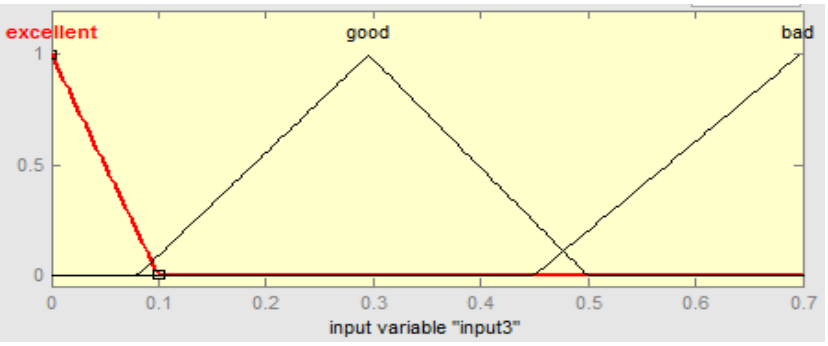
Лінгвістична змінна даної метрики, що містить три терма:

– відмінно (час завантаження сторінки перебуває в діапазоні 0 до 0,1 секунд);

– нормально (час завантаження сторінки перебуває в діапазоні від 0,08 до 0,5 секунд);

– погано (коли час завантаження сторінки перевищує 3).

Для відповідних термів визначені види функцій приналежності й у такий спосіб задана лінгвістична змінна середній час завантаження сторінки, представлена на рисунку 2.3.



Рисунку 2.3 - Вид функцій приналежності для змінної «Максимальний відносний час виконання»

4) Кореляція користувачів/час.

Визначається як коефіцієнт кореляції між числом відвідувачів і часом завантаження сторінок. У випадку, якщо коефіцієнт близький до 1, те це відображає, що при збільшенні числа відвідувачів час завантаження сторінок росте пропорційно. Виходить, що починаючи з деякого числа користувачів, час завантаження буде перевершувати пропоновані стандартом границі.

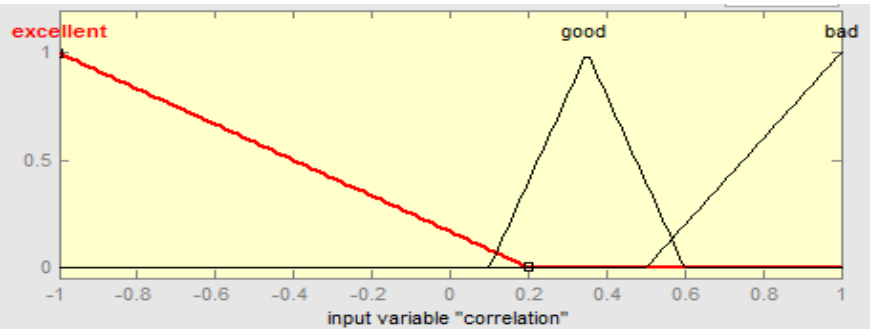
Лінгвістична змінна даної метрики, що містить три терма:

– відмінно (час завантаження сторінки перебуває в діапазоні -1 до 0,2);

– нормально (час завантаження сторінки перебуває в діапазоні від 0,2 до 0,6);

– погано (коли час завантаження від 0,5 до 1).

Для відповідних термів визначені види функцій приналежності й у такий спосіб задана лінгвістична змінна середній час завантаження сторінки, представлена на рисунку 2.4.



Рисунку 2.4 - Вид функцій приналежності для змінної «Кореляція користувачів/час»

5) Швидкість росту часу завантаження сторінки

Визначається як кут нахилу регресійної прямої, побудованої для ряду часу завантаження сторінки. Якщо тангенс кута нахилу близький до нуля, то це означає, що час завантаження сторінки не збільшується із часом і при збільшенні числа відвідувачів. Якщо ж тангенс кута нахилу позитивний і прагне до нескінченності, то це означає, що з ростом числа відвідувачів, час завантаження сторінки не стаціонарно, а росте.

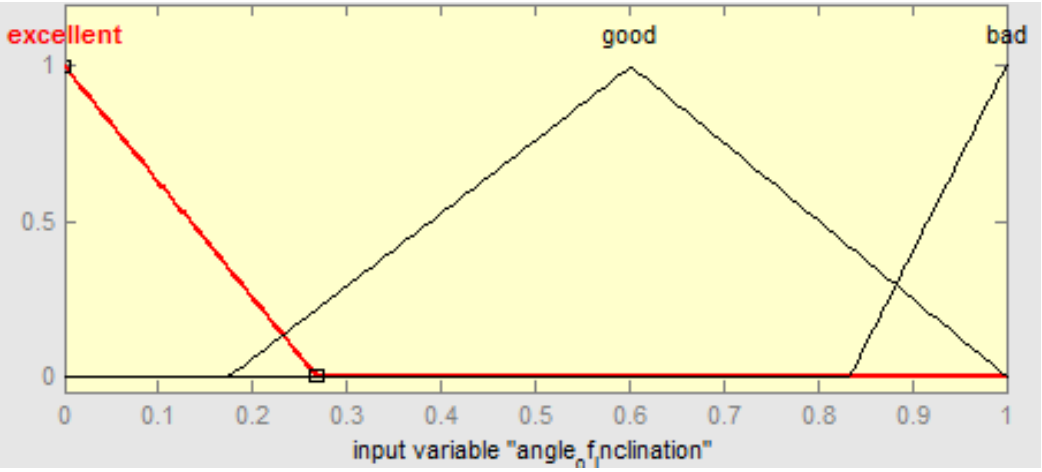
Лінгвістична змінна даної метрики, що містить три терма:

– відмінно (час завантаження сторінки перебуває в діапазоні 0 до 0,2679);

– нормально (час завантаження сторінки перебуває в діапазоні від 0,1763 до 1);

– погано (коли час завантаження сторінки перевищує 0,834).

Для відповідних термів визначені види функцій приналежності й у такий спосіб заданий лінгвістична змінна середній час завантаження сторінки, представлена на рисунку 2.5.



Рисунку 2.5 - Вид функцій приналежності для змінної «Швидкість росту часу завантаження сторінки»

2.3 Вимоги до продуктивності web-застосунків

Чіткі вимоги до web – застосунка, що тестується :

1) кількість користувачів, які будуть використовувати даний web-застосунок одночасно – це вимога визначає рівень навантаження, що повинне витримувати застосунок. Тут же враховується поводження користувачів при його використанні, визначається частота виконуваних дій і час бездіяльності (Think Time), що йде на вивчення інформації на сторінці. З огляду на ці фактори, визначається відповідна кількість віртуальних користувачів VU_{def}, що буде емулювати поводження реальних користувачів;

2) максимально припустимий час завантаження сторінок (RT_{def} – Response Time) – граничний поріг часу відгуку для кожної сторінки web-застосунка. Це вимога звичайно вводиться у відсотковому відношенні: наприклад, 95% сторінок сайту повинні вантажитися не більше 6 секунд. Це найважливіша вимога з погляду кінцевого користувача, тому що при збільшенні часу завантаження однієї сторінки в нього виникає роздратування, його увага перестає бути сконцентроване на сайті, він може взагалі втратити до нього інтерес;

3) гранична пропускна здатність сервера (Th_{def} – Throughput) – максимальна кількість байтів інформації в секунду, які може передавати сервер. У кожного web-застосунка є максимальний поріг швидкості передачі даних, що обумовлений топологією мережі або шириною каналу підключення до інтернету. При досягненні максимального

порога можуть виникнути затримки в часі відгуку на запити користувачів, збої в роботі системи;

4) максимально припустимий розмір однієї сторінки - звичайно допускаються сторінки розміром не більше 500Кб, тому що при більших значеннях користувачеві доводиться чекати занадто великий час для повного завантаження сторінки.

3 МЕТОДИ АНАЛІЗУ РЕЗУЛЬТАТІВ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі будуть розглянуті методи, основні правила та алгоритм аналізу навантажувального тестування.

3.1 Метод побудови нечіткого виводу

На даний момент обчислення, що використовують переваги нечітких систем, показали себе досить потужним і ефективним засобом для вирішення багатьох завдань контролю і управління в складних програмних і апаратних комплексах.

3.1.1 Основні правила виводу в нечіткій логіці

Припустимо, що присутні в правилах modus ponens й modus tollens судження характеризуються деякими нечіткими множинами. Таким способом ми одержуємо узагальнене правило виводу modus ponens й узагальнене правило виводу modus tollens, таблиці 3.1-3.4. У наступному викладі залежності типу “Якщо *A*, те *B*” будемо записувати в символіці класичної мови програмування ALGOL у вигляді IF *A* THEN *B*. Із цієї причини в деяких російськомовних формулах будуть зустрічатися англійські терміни[8].

Таблиця 3.1 - Правило виводу modus ponens

Умова імплікації	A $A \rightarrow B$
Вивід	B

Таблиця 3.2 - Правило виводу modus tollens

Умова імплікації	B $A \rightarrow B$
Вивід	A

Таблиця 3.3 - Узагальнене (нечітке) правило modus ponens

Умова імплікації	x це $A`$ **IF** x це A **THEN** y це B
Вивід	y це $B`$

де A, $A` \subseteq \mathbf{X}$ й $B$, $B` \subseteq \mathbf{X}$ – нечіткі множини, у той час як x й y – так називані лінгвістичні змінні.

Лінгвістичними називаються змінні, значення яких являють собою слова або судження природною мовою. Як приклади можна привести вираження типу “мала швидкість”, “помірна температура” або “парубок”. Подібні вираження можна формалізувати приписуванням їм деяких нечітких множин. Варто підкреслити, що лінгвістичні змінні крім словесних значень можуть мати й чисельні значення - так само, як звичайні математичні змінні.

Таблиця 3.4 - Узагальнене (нечітке) правило modus tollens

Умова імплікації	y це $B`$ **IF** x це A **THEN** y це B
Вивід	x це $A`$

де A, $A` \subseteq \mathbf{X}$ й $B$, $B` \subseteq \mathbf{X}$ – нечіткі множини в той час як x й y – так звані лінгвістичні змінні.

3.1.2 Нечітке управління

Для багатьох додатків, пов'язаних з управлінням технологічними процесами, необхідна побудова моделі розглянутого процесу. Знання моделі дозволяє підібрати відповідний регулятор (модуль керування). Однак нерідко побудова коректної моделі являє собою важку проблему, що вимагає іноді введення різних спрощень.

Застосування теорії нечітких множин для керування технологічними процесами не припускає знання моделей цих процесів. Варто тільки сформулювати правила поведінки у формі нечітких умовних суджень типу **IF…THEN**...

На рис. 3.1 представлена типова структура модуля нечіткого управління. Він складається з наступних компонентів: бази правил, блоку фаззификации, блоку виробітку рішення, блоку дефаззифікації.

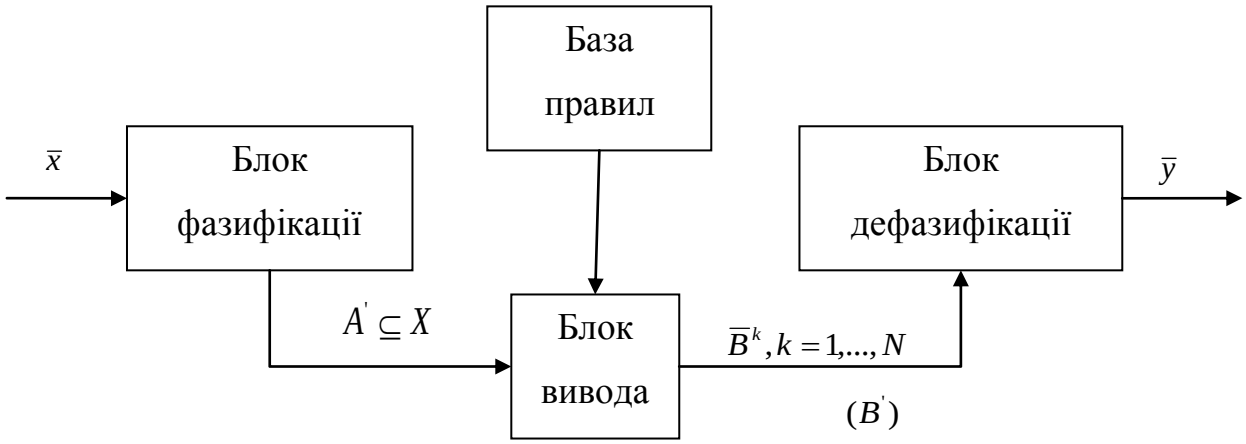


Рисунок 3.1 - Модуль нечіткого керування

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

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

де:

N – кількість нечітких правил,

A_i^k – нечіткі множини

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

B_i^k – нечіткі множини

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

$x_1, x_2, \ldots, x_n$ – вхідні змінні лінгвістичної моделі, причому

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

$y_1, y_2, \ldots, y_m$ – вихідні змінні лінгвістичної моделі, причому

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

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

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

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

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

Зауважимо, що кожне правило виду (3.1) складається із частини **IF**, називаною посилкою, і частини **THEN**, називаної наслідком. Посилка правила містить набір умов, тоді як наслідок містить вивід. Змінні $x = (x_1, x_2, \ldots, x_n)^T$ й y можуть приймати як лінгвістичні, так і числові значення. Якщо ввести позначення

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

те правило (3.1) можна представити у вигляді нечіткої імплікації

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

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

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

При проектуванні модулів нечіткого управління варто оцінювати достатність кількості нечітких правил, їхня несуперечність та наявність кореляції між окремими правилами.

3.2 Нечіткі правила

Опишемо тепер правила, які побудовані для системи автоматизації навантажувального тестування. Відстежувальними змінними є: x_1 – час завантаження з елементами сторінки; x_2 – середній час завантаження html коду сторінки; x_3 – швидкість одержання/передачі даних, x_4 – кореляція користувачів/час, x_5 – швидкість росту часу завантаження сторінки.

Якщо порахувати усі можливі комбінації правил, то їхня кількість більше 500 , нижче буде один із прикладів правила.

На рисунку 3.2 зображено, як виглядає набір правил у програмі MATLAB.

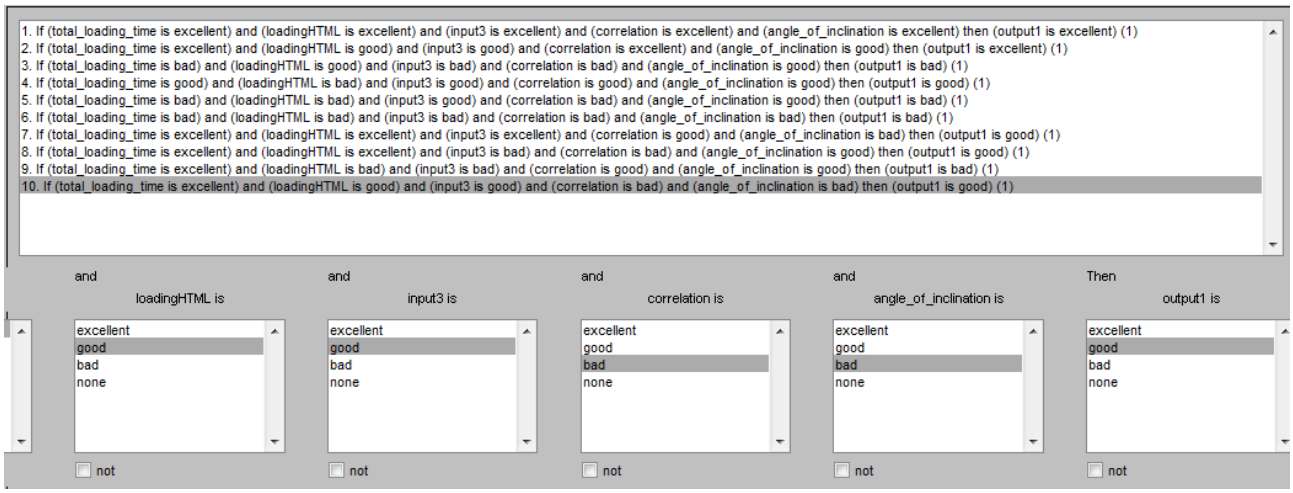


Рисунок 3.2 – Фрагмент набору правил у програмі MATLAB

Правило. Визначає оптимальний режим функціонування web-застосунка, тобто коли змінні x_1 (час завантаження з елементами сторінки), x_2 (середній час завантаження html коду сторінки), x_3 (швидкість одержання/передачі даних), x_4 (кореляція користувачів/час) і x_5 (швидкість росту часу завантаження сторінки) приймають мінімальні значення рисунок 3.3.

Рисунок 3.3 – Графічне зображення правила

У результаті ми отримали 0.147, що вказує нам на відмінну роботу сайта.

3.3 Алгоритм автоматичної побудови нечітких правил

Мова таблиці рішень (ТР) дозволяє задавати набір простих продукційних правил $\{R_j\}, j=1..m$ на основі пропозиціональної логіки: $(R_j) C_1 \wedge C_2 \wedge \ldots \wedge C_n \rightarrow A_1 \wedge A_2 \wedge \ldots \wedge A_k$. Множини вхідних термів (умов) $\{C_i\}$ і вихідних термів (дій) є загальними для всіх правил в ТР. Такий набір правил може бути представлений в табличній формі, що й дало назву даному формальному апарату.

Незважаючи на існування стандарту, в різних роботах наводяться різні описи як структури ТР, так і їх семантики.

Метою даної роботи була побудова формального опису нечітких таблиць рішень (НТР), що базуються на класичних ТР, але здатних працювати як з дискретними, так і з безперервними і нечіткими входами і виходами. Ідея такої гібридизації з'явилася в літературі в середині 90-х років XX століття [11-12], проте в тих публікаціях була чітко сформульована формальна модель апарату НТР, нечіткі входи таблиць розглядалися у відриві від стандартних дискретних, що не дозволяє говорити про ефективне узагальнення апарату ТР на нечіткий випадок. Для виведення по НТР використовувалися стандартні загальні схеми нечітких міркувань, які не використовують особливості НТР для побудови більш ефективних спеціалізованих алгоритмів виводу.

Визначимо атрибут як трійку: $a_i = (Dom_i, Val_{i,}, \varphi_i) \in A$

де — $Dom_i = \{d\}$ — множина вхідних значень (домен) атрибута (потенційно незліченну);

— $Val_i = \{v_i\}$ — кінцева множина вихідних значень атрибута a_i;

— $\varphi_i : Dom_i \times Val_i \mapsto [0.0,1.0]$— тотальна функція відповідності значень атрибута a_i, оцінює ступінь істинності $\varphi_i(d, v_i)$ вихідного значення $v_l \in Val_i$ для елемента домену;

— $A = \{a_i\}$ – множина всіх атрибутів заданої структури.

У цю структуру укладаються кілька базових класів атрибутів:

булеві атрибути виду $a_B = (Dom_B, Val_B, \varphi_B)$,

де — $Dom_B = Val_B = \{True, False\}$;

— $\varphi_B(d, v) = \begin{cases} 1.0, якщо\ d = v \\ 0.0\ інакше \end{cases}$;

N-значні дискреційні атрибути виду $a_N = (Dom_N, Val_N, \varphi_N)$,

де — $Dom_N = \{1, 2, \ldots, n\}, n \geq 2$;

— $Val_N = \{v_1, v_2, \ldots, v_n\}$;

— $\varphi_N(d, v_i) = \begin{cases} 1.0, & якщо\ d = i \\ 0.0, & інакше \end{cases}$;

речові атрибути виду $a_C = (Dom_C, Val_C, \varphi_C)$,

де — $Dom_C = [domMin, domMax] \subset \Re$;

— $Val_C = \{\Delta_1, \Delta_2, ..., \Delta_n\}$, n ≥2, де Val_C є розбиттям Dom_C;

— $\varphi_C(d, \Delta_i) = \begin{cases} 1.0, & якщо\ d \in \Delta_i \\ 0.0, & інакше \end{cases}$;

нечіткі атрибути виду $a_F = (Dom_F, Val_F, \varphi_F)$,

де — $Dom_F = [domMin, domMax] \subset \Re$;

— $Val_F = \{v_1, v_2, \ldots, v_n\}, n \geq 2$, де $v_l = (Name_l, Dom_F, A_l)$ – стандартне визначення нечіткої змінної з ім'ям Name1, характеризується нечіткою множиною $A_l = \{d, \mu_l(d)\}$ з функцією приналежності $\mu_l : Dom_F \mapsto [0.0, 1.0]$ [10];

— $\varphi_F(d, v_l) = \mu_l(d)$.

Приклад набору трапецієподібних функцій приналежності значень нечітких атрибутів (рис. 3.4). Приклад описує нечіткий атрибут «#Jobs in Queue», який приймає три значення: Low, Normal і High. Доменом атрибуту є відрізок [0, 8].

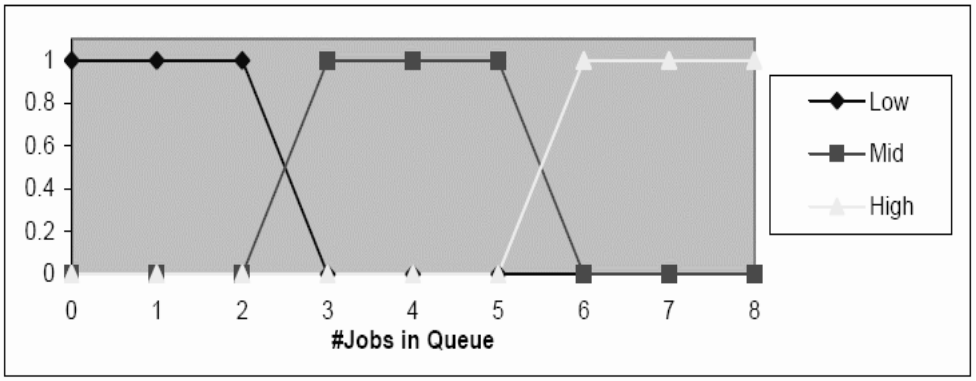


Рисунок 3.4 - Приклад функцій приналежності

У загальному випадку для завдання функцій приналежності можуть використовуватися довільні Z, S, П-функції. Надалі, кажучи про певний атрибуті $a \in A$, будемо звертатися до його компонентів також у функціональному стилі, тобто Dom(a), Val(a).

Визначимо формально поняття нечіткої таблиці рішень (НТР). НТР є трійка FDT = (C,D,F),

де — $C = \{c_i\} \subset A$ – кінцева множина умовних атрибутів (умов);

— $D = \{d_i\} \subset A$ – кінцева множина вирішальних атрибутів (рішень, дій);

— $F : \prod_{c_i \in C} Val(c_i) \times \prod_{d_i \in D} Val(d_i) \mapsto (0.0, 1.0]$ – узагальнена вирішальна функція (функція впевненості).

Введемо додаткові поняття:

1. Вхідним вектором (ситуацією), що належить множині можливих ситуацій $Dom_C(FDT) = \prod_{c_i \in C} Dom(c_i)$, називається вектор $\bar{\mathbf{x}} = (x_i) \in \prod_{c_i \in C} Dom(c_i)$

2. Вихідним вектором (рішенням), що належить множині рішень $Dom_D(FDT) = \prod_{d_i \in D} Dom(d_i)$, називається вектор $\bar{\mathbf{y}} = (y_i) \in \prod_{d_i \in D} Dom(d_i)$.

Тоді процес прийняття рішення на основі НТР можна представити як відображення вхідних ситуацій на рішення при посередництві НТР:

$\bar{x} \xrightarrow{FDT} \bar{y}, \quad \bar{x} \in Dom_C(FDT), \bar{y} \in Dom_D(FDT)$.

Знання, представлені у вигляді функції, можуть бути представлені і у вигляді набору продукційних правил спеціального виду. Нехай дані $FDT = (C, D, F)$, $\overline{\mathbf{v_C}} = (v_{C,i}) \in \prod_{c_i \in C} Val(c_i)$, $\overline{\mathbf{v_D}} = (v_{D,i}) \in \prod_{d_i \in D} Val(d_i)$ та відомо, що $F(\overline{\mathbf{v_C}}, \overline{\mathbf{v_D}}) = cf$. Остання рівність еквівалентно завданню вирішального правилом види:IF $\bigwedge_{c_i \in C} (c_i \text{ is } v_{C,i})$ THEN $\bigwedge_{d_i \in C} (d_i \text{ is } v_{D,i})$ з коефіцієнтом впевненості cf .

Тут в антецеденте і консеквента правила записуються нечіткі кон'юнкції оцінок значень атрибутів НТР, а саме правило є не достовірним, а правдоподібним з коефіцієнтом впевненості $cf \in (0.0, 1.0]$, рівним значенню функції впевненості F від значень атрибутів. При $cf = 1.0$ правило є достовірним.

Як і у випадку класичних ТР, які правила можуть бути представлені в табличному вигляді, де вирішальні правила записуються в шпальтах, а рядки таблиці відповідають умовним і вирішальним атрибутам НТР. На входах умовної частини НТР застосовується символ байдужості «*», якщо значення умовного атрибута несуттєво в правилі. В останньому рядку записуються коефіцієнти впевненості правил. НТР, що описує вибір стратегії упорядкування завдань у черзі обслуговуючого пристрою. Для прийняття рішення використовуються 5 умовних атрибута: x_1 (час завантаження з елементами сторінки), x_2 (середній час завантаження html коду сторінки), x_3 (швидкість

одержання/передачі даних), x_4 (кореляція користувачів/час) і x_5 (швидкість росту часу завантаження сторінки).

Класичні НТР є окремим випадком пропонованого в роботі апарату НТР, де використовуються тільки булеві і дискретні атрибути, а коефіцієнти впевненості всіх правил покладаються рівними 1.0.

Хоча НТР містять як нечіткі, так і дискретні, безперервні входи, процес прийняття рішень повинен базуватися саме на схемі нечіткого виведення, узагальнюючої інші випадки. Для нечітких пропозіціональних правил довільної форми відомі кілька основних схем нечіткого виводу [10]. В якості базової схеми для виведення на НТР візьмемо схему Цукамото (Tsukamoto). Вона має невисоку обчислювальну складність, що дозволяє застосовувати її в ситуаціях з обмеженим лімітом часу на прийняття рішення, у тому числі в експертній системі підтримки прийняття рішень реального часу (ЕСППР РЧ). Крім того, вона широко використовується в існуючих додатках нечіткої логіки, її надійність і ефективність перевірені практикою. Пряме використання висновку за Цукамото для прийняття рішень з НТР неможливо, потрібна адаптація базового алгоритму. Розглянемо основні кроки пропонованого в роботі алгоритму виведення по НТР.

На етапі фазифікації необхідно перевести компоненти вектора ситуації $\bar{\mathbf{x}} = (x_i) \in Dom_C(FDT)$ у значення відповідних атрибутів. Для цього обчислюється множина $ValSet_i(x_i)$ результатів застосування функції відповідності значень φ_i до вхідного значення x_i та всім значенням атрибута $v \in Val_i$: $ValSet_i(x_i) = \{(x_i, \varphi_i(x_i, v)) : v \in Val_i\}$.

На етапі агрегування умов здійснюється обчислення коефіцієнтів активації вирішальних правил в НТР на підставі множин $ValSet_i(x_i)$. Коефіцієнт активації Act правила Rule, описаного вище виду, залежить від оцінок істинності значень умовних атрибутів і коефіцієнта впевненості правила відповідно до формули $Act = cf * \prod_{c_i \in C} \varphi_i(x_i, v_{C,i})$. Відповідно до Цукамото нечітка кон'юнкція в антецеденте правила інтерпретується тут як речовий твір. Правила, коефіцієнт активації яких більше нуля, називаються активними і враховуються на наступних кроках процесу виведення.

Наступні етапи активації висновків і обчислення вихідних значень виконуються окремо для кожного вирішального атрибута $d_i \in D$ в НТР. Етап активації висновків застосовується лише до нечітких та речових атрибутів. В ході цього для кожного активного правила Rule, обчислюється попереднє початкове значення $w_{ij} \in Dom(d_i)$

вирішального атрибута $d_i \in D$ за значенням атрибута $v_{D,i} \in Val(d_i)$ на завершення цього правила, як центр його інтервальної оцінки за формулою $w_{ij} = \frac{\inf(\Delta w_{ij}) + \sup(\Delta w_{ij})}{2}$. Сама интервальная оцінка обчислюється за формулою $\Delta w_{ij} = \{y \in Dom(d_i) : \varphi_i(y, v_{D,i}) \geq Act_j\}$. На завершальному етапі проводиться остаточне обчислення компонентів вектора рішення $\bar{\mathbf{y}} = (y_i) \in Dom_D(FDT)$. Для нечіткого чи речового вирішального атрибута $d_i \in D$ значення рішення $y_i \in Dom(d_i)$ задається формулою $y_i = \sum_{j=1}^{n} Act_j * w_{ij} \Big/ \sum_{j=1}^{n} Act_j$, де підсумовування ведеться по всім активним правилам в НТР (принцип центру ваги для одноточечних множин, Centre of Gravity for Singletons, [10]). Для булевого або дискретного вирішального атрибута $d_i \in D$ рішення отримуємо відповідно до формули $y_i = \arg \max_{y \in Dom(d_i)} \left(\sum_{j=1}^{n} Act_j * \varphi_i(y, v_{D,i}) \right)$.

За рішення приймається той елемент домену вирішального атрибута, який характеризується максимальною сумою коефіцієнтів активації правил, в яких вказано значення атрибута, асоційоване з даним елементом домену.

3.4 Загальна схема запропонованого методу аналізу web-застосунку

Для того щоб краще зрозуміти запропонований метод аналізу, нижче буде приведена схема за якою проводиться навантажувальне тестування систем та побудова безпосередньо самих правил.

Для того щоб побудувати систему правил, нам потрібно знайти сайти з різними показниками працездатності при великих навантаженнях.

1) Вибір списку найбільш рейтингових сайтів (Alexa.com).

Сайт з гарними показниками та відмінною роботою ми можемо знайти в топі найкращих Alexa.com та менш стійкий сайт до великих навантажень.

2) Формування сценаріїв навантажувального тестування для кожного типу перевірки.

На даному етапі ми визначаємо за якими варіантами тестового оточення та вхідними даними буде проводитись тестування. Крім того, для кожного типу тестування (розділ 2.1) доцільно задати свій сценарій поведінки групи користувачів. Так, наприклад, при стрес тесті необхідно в короткий проміжок часу подати велике навантаження, а для об‘ємного тесту потрібно подавати плавне навантаження.

3) Проведення навантажувального тестування для кожного сайту.

Для кожного сайту зі списку, що складено в пункті 1.1 згідно сценаріям за п. 1.2 проводиться навантажене тестування.

4) Обчислення метрик по обробленим сайтам.

Після проведення навантажувального тестування ми отримаємо графіки та числові ряди середнього часу завантаження сторінки.

Виходячи з отриманих даних формуються основні метрики, завдяки яким ми зможемо робити аналіз систем.

5) Побудова системи правил.

На основі отриманих статистичних даних, ми можемо побудувати системи правил, взявши за гарний результат сайту, який працює швидко, продуктивно, а той який повільно відповідає на запити та довго завантажується.

В результаті етапів з 1 пункту алгоритму, ми отримаємо систему правил, що дозволяє в подальшому для невідомих сайтів проводити автоматично аналіз та діагностувати.

Аналіз сайту та рекомендації :

- висновок на нечіткіх правилах за обчисленими в попередньому розділі метрик;
- формування списку рекомендацій.

Після навантажувального тестування ми отримаємо дані , підставляємо їх до визначених нами метрик, та отримуємо терм (відмінно, нормально, погано) по кожній лінгвістичній змінній(середній час завантаження сторінки, середній час завантаження html коду сторінки, швидкість росту часу завантаження сторінки, залежність від кількості користувачів на сайті).

На основі проведеного аналізу, ми можемо скласти рекомендації, що до удосконалення систем, які показали себе не як найкраще. Сформувати ряд пропозицій та засобів для підвищення продуктивності роботи сайту (змешити кількість http-запитів, оптимізувати кешування та ін.).

Рисунок 4.1 – Перший етап методу

Рисунок 4.2 – Другий етап методу

4 ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗАЦІЇ ОБРОБКИ РЕЗУЛЬТАТІВ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ WEB-ЗАСТОСУНКІВ

Головною метою систем атоматизованої обробки інформації - це узагальнити та перетворити вхідну інформацію для отримання відомостей, які необхідні для прийняття рішення.

У даному розділі ми розглянемо реалізацію автоматизованої обробки результатів тестування web-застосунків, отримання результатів навантажувального тестування, обробка даних та висновок.

4.1 Отримання результатів навантажувального тестування

В даній роботі було проведено навантажувальне тестування по наступних сайтах:

1) http://www.microsoft.com/ - сайт корпорації Microsoft;
2) http://snu.edu.ua/ - головний сайт Східноукраїнського національного університету ім.В.Даля;
3) http://cse.turion.info/ сайт кафедри КНІ Східноукраїнського національного університету ім.В.Даля;
4) https://snu.edu.ua/rasp/ сайт розкладу Східноукраїнського національного університету ім.В.Даля;
5) http://sd.ua/ - медіа портал;
6) http://comsvit.com.ua/ - сайт фірми Comsvit;
7) http://in.skyline.kh.ua/ - медіа портал.

Навантажувальне тестування проводиться за допомогою програми WAPT. Сценарій навантажувального тестування на сайти передбачував провести навантаження віртуальних користувачів від 5 до 175. В результаті ми отримуємо звіт у вигляді HTML коду та файлу .csv, данні з якого ми будемо використовувати надалі. Результати у вигляді графіку можна побачити на рисунках 4.1 – 4.7.

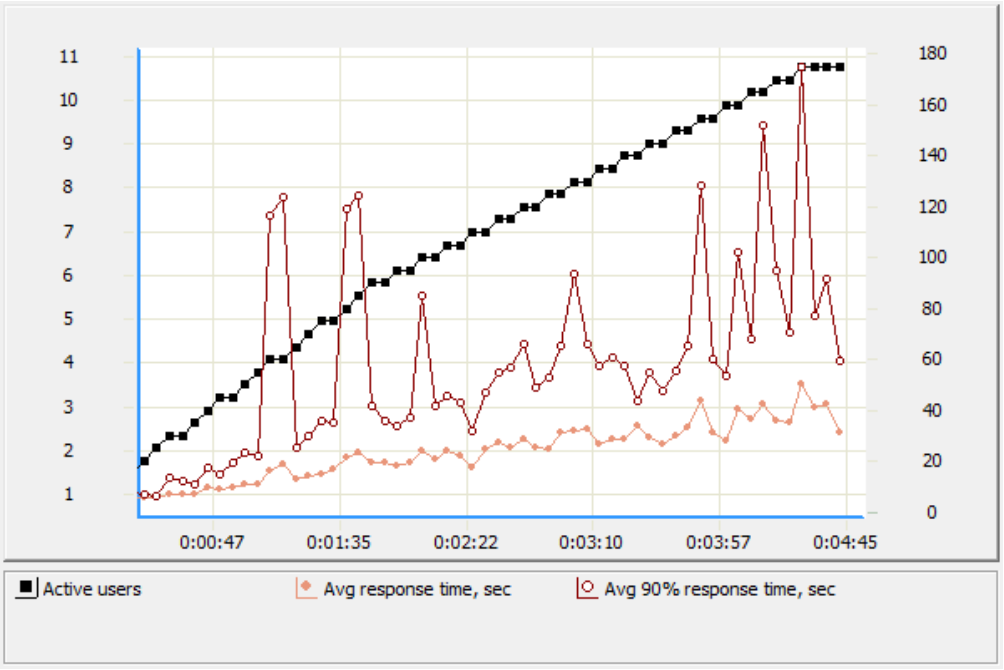


Рисунок 4.1 - Результати навантажувального тестування сайту http://www.microsoft.com/

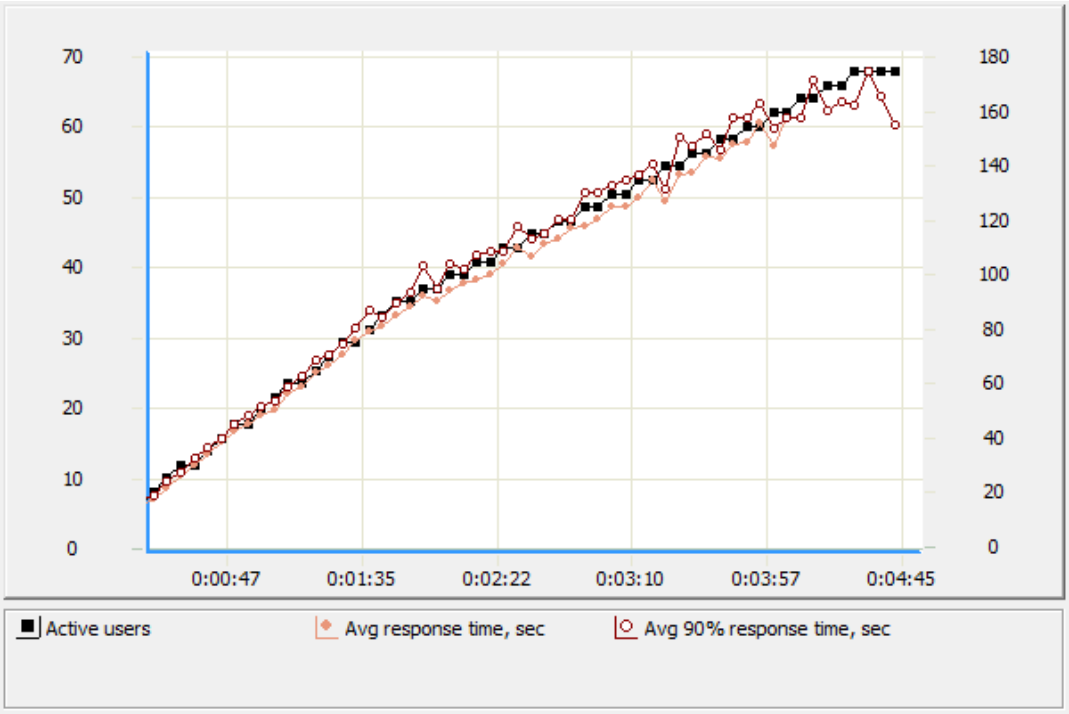


Рисунок 4.2 - Результати навантажувального тестування сайту http://snu.edu.ua

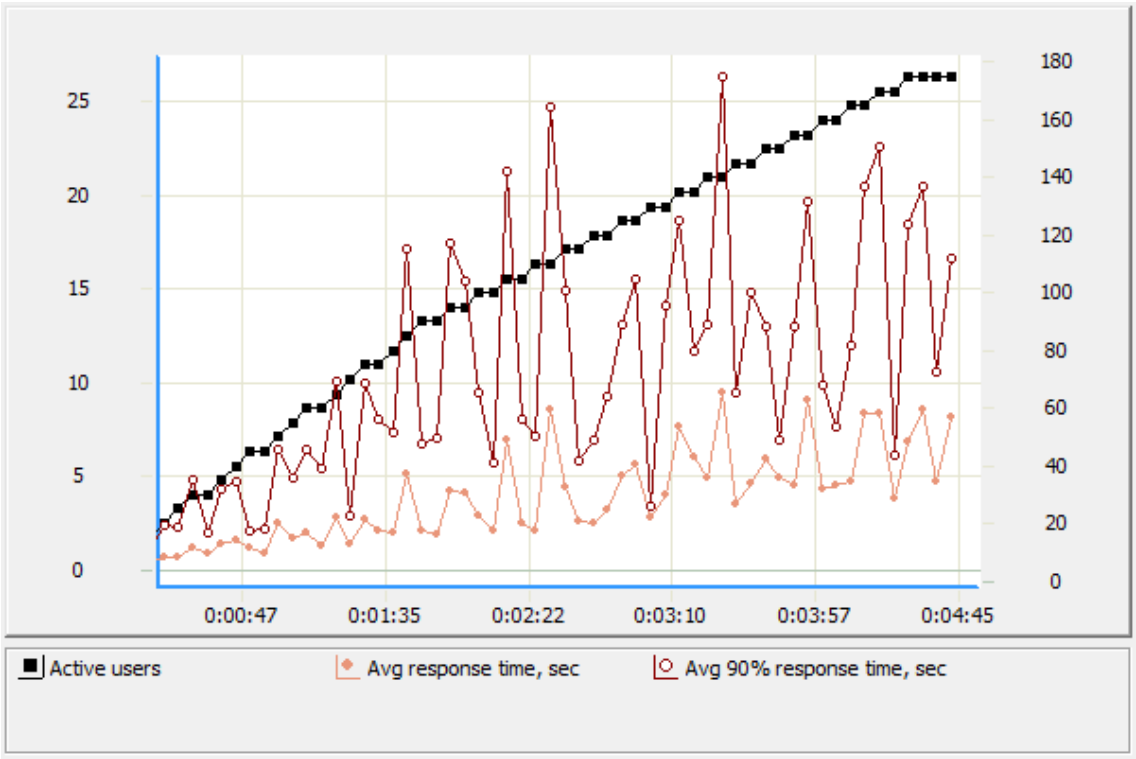


Рисунок 5.3 - Результати навантажувального тестування сайту http://cse.turion.info/

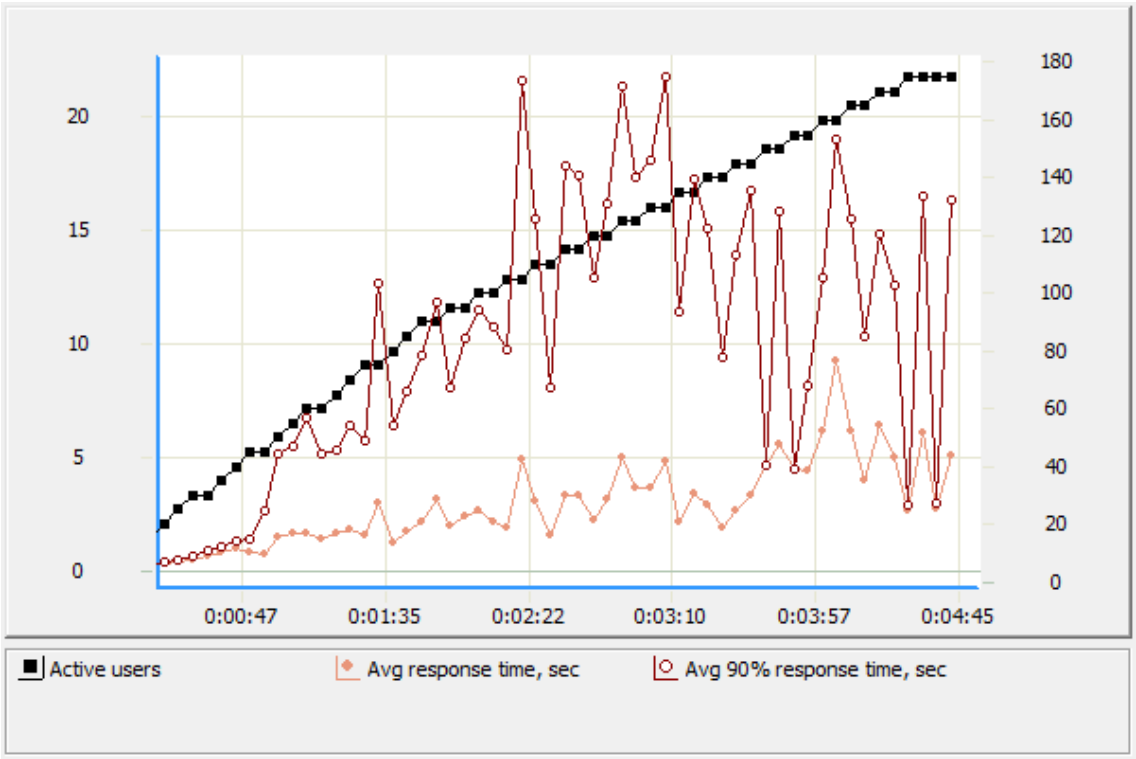


Рисунок 5.4 - Результати навантажувального тестування сайту https://snu.edu.ua/rasp/

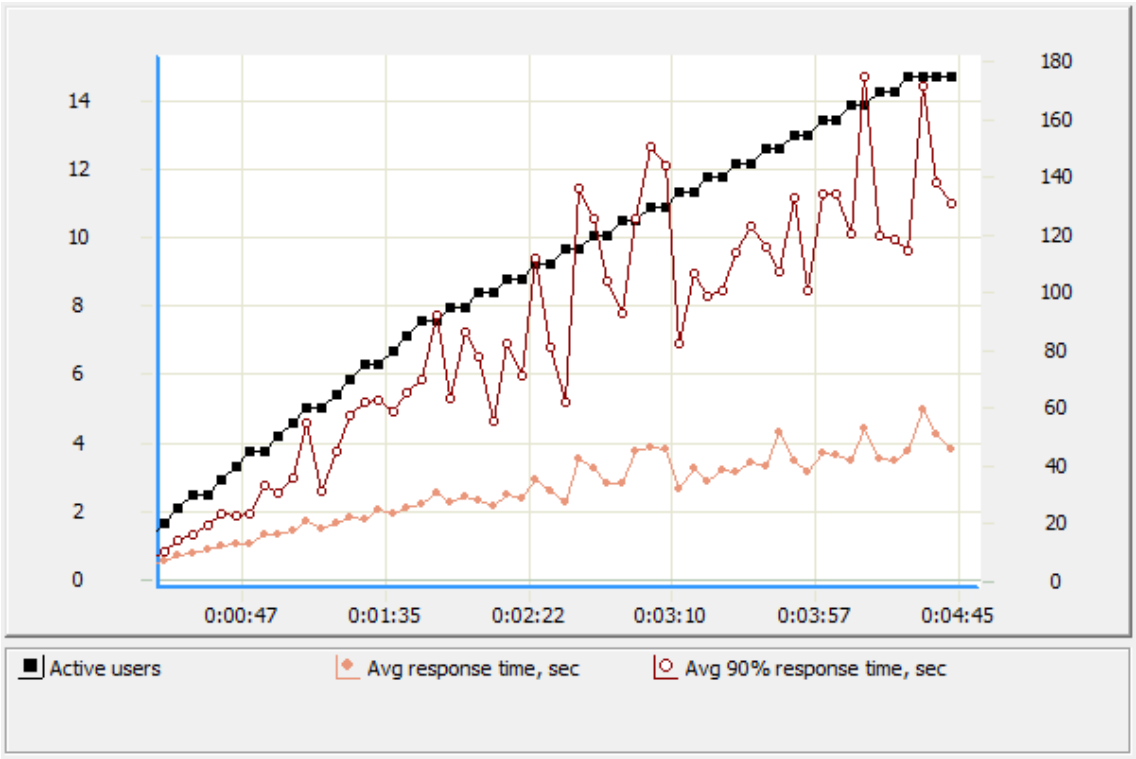


Рисунок 4.5 - Результати навантажувального тестування сайту http://sd.ua/

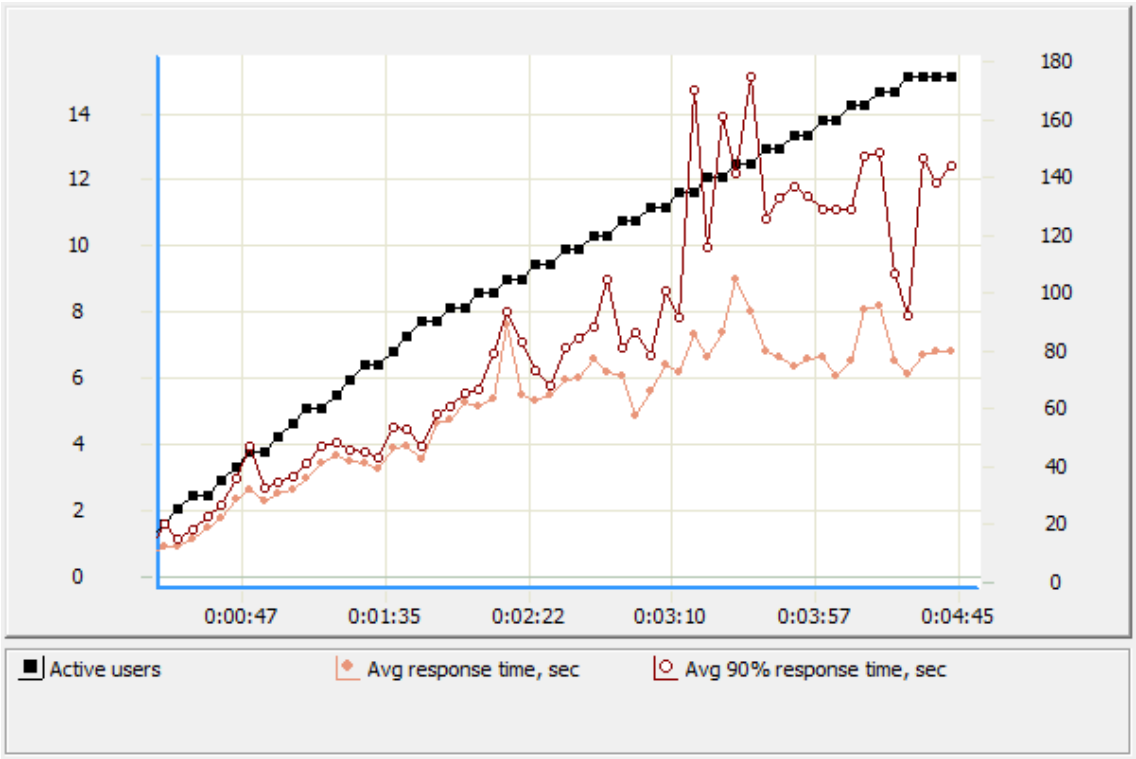


Рисунок 4.6 - Результати навантажувального тестування сайту http://comsvit.com.ua/

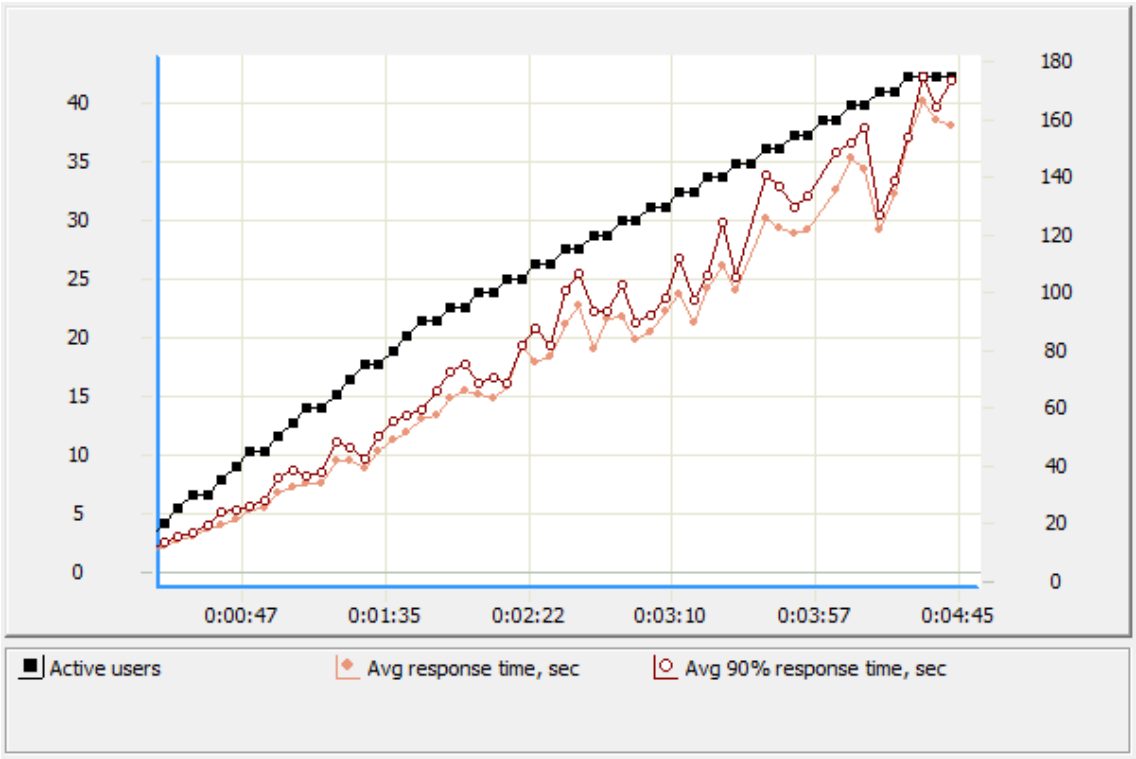


Рисунок 4.7 - Результати навантажувального тестування сайту http://in.skyline.kh.ua/

4.2 Обробка даних навантажувального тестування та висновок на основі нечіткої логіки.

У даному розділі ми отримаємо висновок про ефективність розглянутих систем, на основі нечіткої логіки та правил, які побудовані для прововедення автоматизованного аналізу.

4.2.1 Нечіткі правила

Наведемо тепер правила, які побудовані для системи автоматизації навантажувального тестування. Відстежуваними змінними є: x_1 - час завантаження з елементами сторінки; x_2 - час завантаження без елементів сторінки; x_3 - залежність від кількості користувачів на сайті; x_4 - швидкість росту часу завантаження сторінки. Функції приналежності нечітких множин для цих змінних задані в главі 2.2. На рисунку 4.8 задано 24 правила для цих змінних, за допомогою яких можна визначити ефективність сайту. З

рисунку видно що є три види приналежності змінних це Well, Good, Bed, що відповідно дорівнюють добре, нормально, погано. Для того щоб було зрозуміліше наведемо перше правило у наступному вигляді: якщо x_1, x_2, x_3 та x_4 приймають значення добре то Y також добре.

Для наглядності на рисунку 4.9 всі ці правила представлені в графічному варіанті.

1. If (x1 is Well) and (x2 is Well) and (x3 is Well) and (x4 is Well) then (Y is Well) (1)
2. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
3. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Bad) then (Y is Bad) (1)
4. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
5. If (x1 is Good) and (x2 is Good) and (x3 is Well) and (x4 is Well) then (Y is Good) (1)
6. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Good) then (Y is Bad) (1)
7. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Well) then (Y is Good) (1)
8. If (x1 is Good) and (x2 is Good) and (x3 is Bad) and (x4 is Bad) then (Y is Bad) (1)
9. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Bad) then (Y is Good) (1)
10. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Good) then (Y is Bad) (1)
11. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Well) then (Y is Bad) (1)
12. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Bad) then (Y is Bad) (1)
13. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Well) then (Y is Bad) (1)
14. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Good) then (Y is Bad) (1)
15. If (x1 is Well) and (x2 is Well) and (x3 is Well) and (x4 is Good) then (Y is Well) (1)
16. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Well) then (Y is Well) (1)
17. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
18. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Well) then (Y is Good) (1)
19. If (x1 is Good) and (x2 is Good) and (x3 is Well) and (x4 is Good) then (Y is Good) (1)
20. If (x1 is Good) and (x2 is Good) and (x3 is Bed) and (x4 is Good) then (Y is Bad) (1)
21. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Bad) then (Y is Bad) (1)
22. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Bad) then (Y is Bad) (1)
23. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Well) then (Y is Well) (1)
24. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Good) then (Y is Good) (1)

Рисунок 4.8 - Нечіткі правила

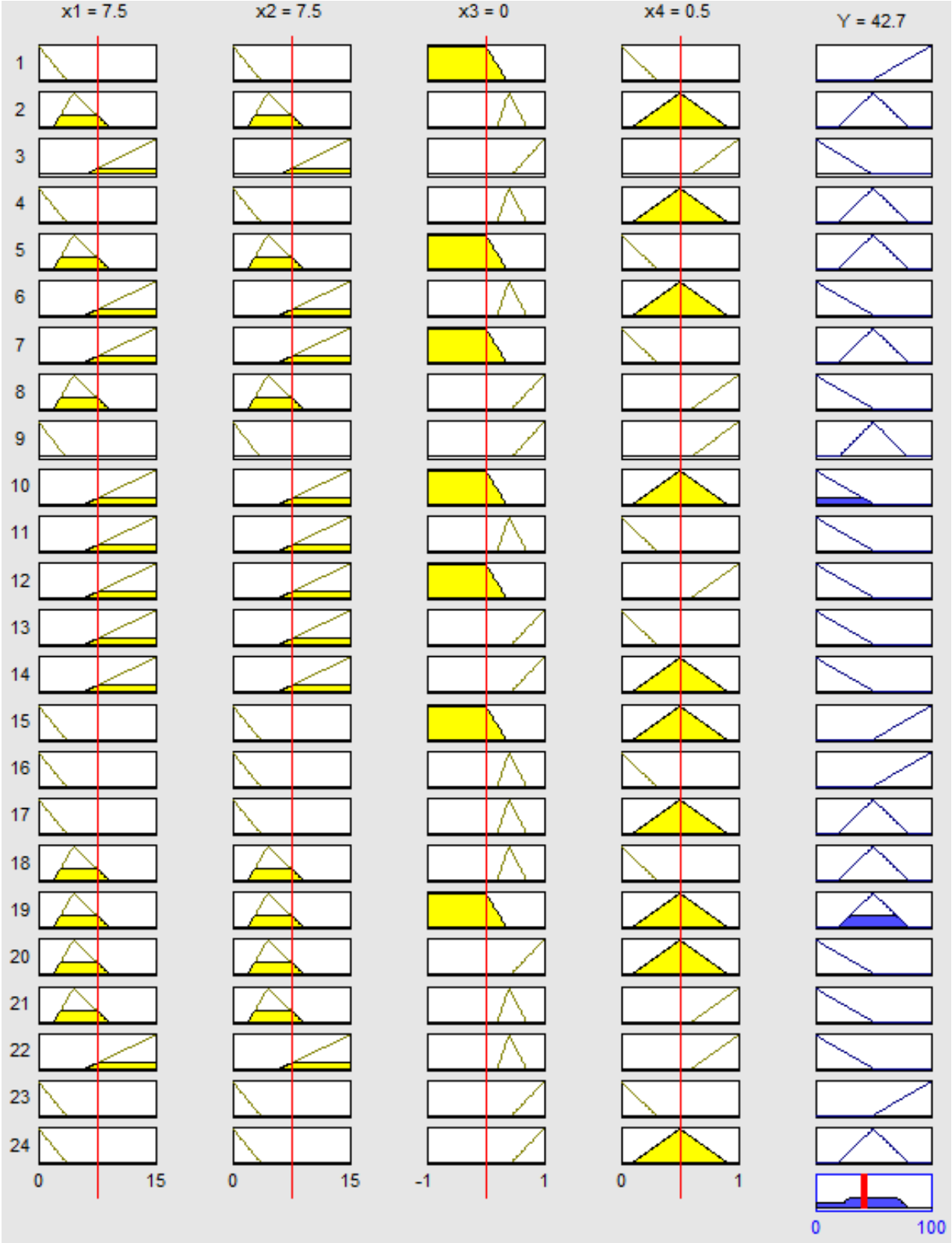


Рисунок 4.9 – Графічне зображення нечітких правил

4.2.2 Обробка даних за результатами навантаженого тестування

Наступним кроком являється обробка даних отриманих в результаті навантажувального тестування. Для цього ми будемо використовувати розроблену програму. До програми потрібно передати файл з даними. Програма їх обробить і видасть звіт, в якому буде вказано кількість користувачів, та 4 параметри, які ми будемо використовувати в нечіткій логіці. Цими параметрами являються: x_1 - час завантаження з елементами сторінки; x_2 - час завантаження без елементів сторінки; x_3 - залежність від кількості користувачів на сайті; x_4 - швидкість росту часу завантаження сторінки. Результати можна побачити на рисунках 4.10-4.16

Рисунок 5.10 - результати обробки даних навантажувального тестування для сайту http://www.microsoft.com/

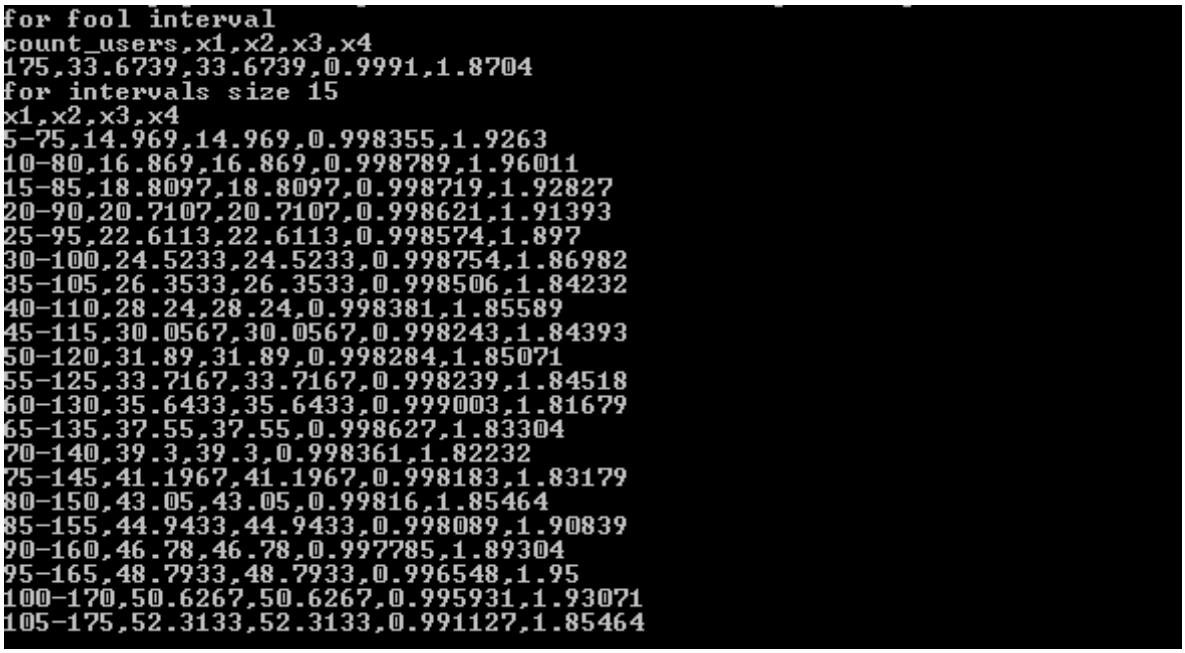


Рисунок 5.11 - результати обробки даних навантажувального тестування для сайту http://snu.edu.ua/

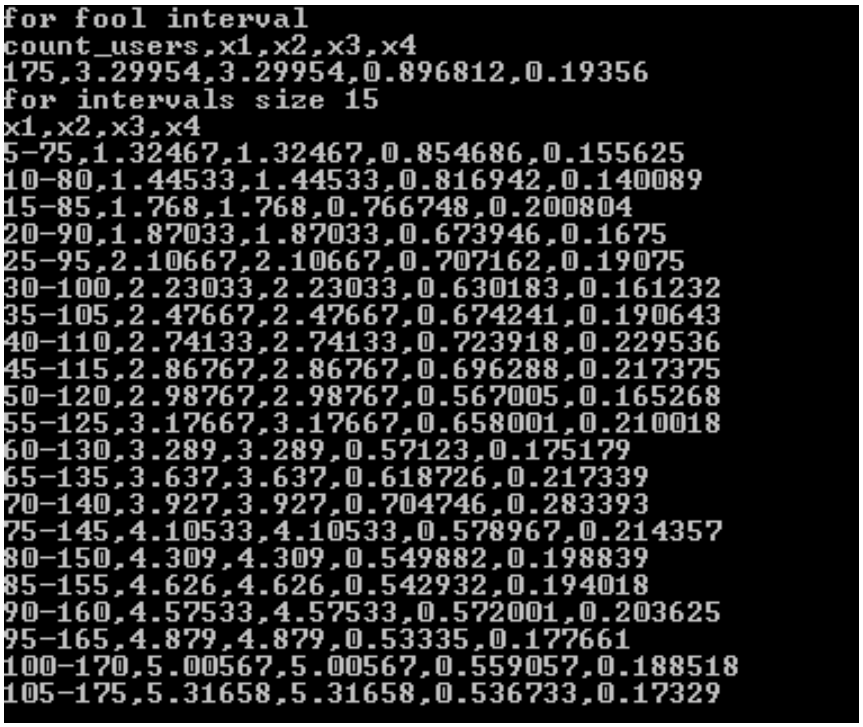


Рисунок 5.12 - результати обробки даних навантажувального тестування для сайту https://snu.edu.ua/rasp/

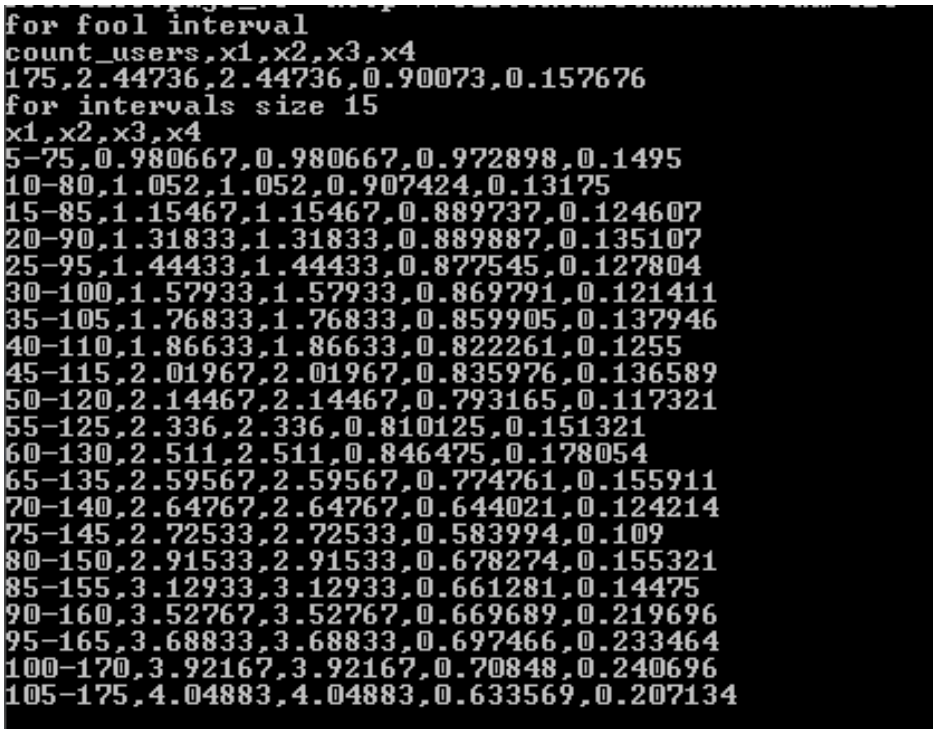


Рисунок 4.13 - результати обробки даних навантажувального тестування для сайту http://cse.turion.info/

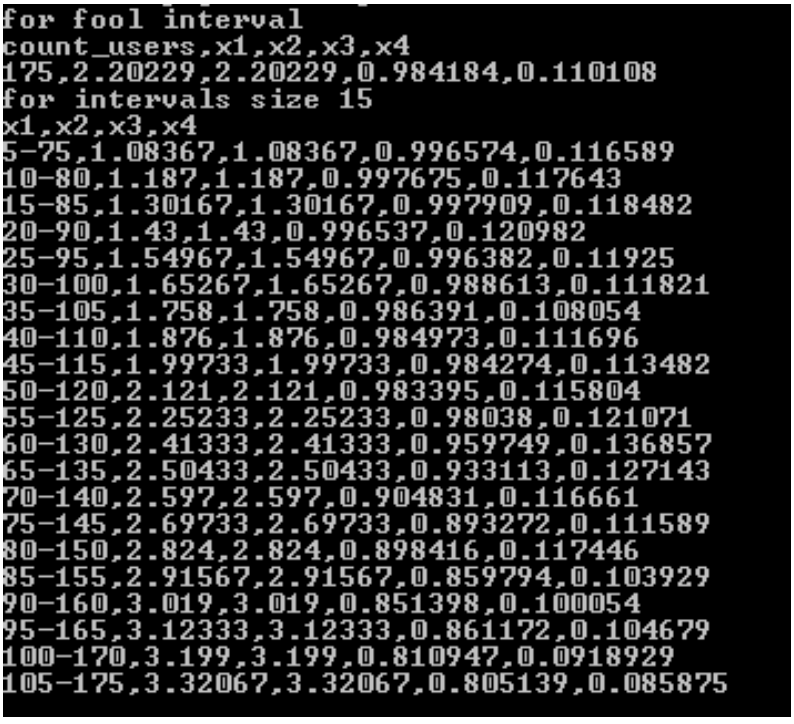


Рисунок 4.14 - результати обробки даних навантажувального тестування для сайту http://sd.ua/

```
Profile7.page_19: http://comsvit.com.ua/[1]
for fool interval
count_users,x1,x2,x3,x4
175,4.30618,4.30618,0.965264,0.222572
for intervals size 15
x1,x2,x3,x4
5-75,1.978,1.978,0.981809,0.260411
10-80,2.21933,2.21933,0.981411,0.259411
15-85,2.45333,2.45333,0.977203,0.250232
20-90,2.68967,2.68967,0.972956,0.236357
25-95,2.96267,2.96267,0.970586,0.244464
30-100,3.253,3.253,0.971949,0.241768
35-105,3.60333,3.60333,0.946193,0.267714
40-110,3.843,3.843,0.941214,0.260446
45-115,4.08833,4.08833,0.947385,0.270321
50-120,4.349,4.349,0.952688,0.280839
55-125,4.545,4.545,0.920692,0.255286
60-130,4.771,4.771,0.904685,0.236911
65-135,5.00733,5.00733,0.90787,0.241375
70-140,5.233,5.233,0.917424,0.252214
75-145,5.567,5.567,0.912799,0.284625
80-150,5.794,5.794,0.876422,0.250714
85-155,5.96433,5.96433,0.81022,0.212589
90-160,6.12467,6.12467,0.708833,0.163196
95-165,6.33867,6.33867,0.66602,0.134179
100-170,6.494,6.494,0.63607,0.122054
105-175,6.59275,6.59275,0.522043,0.0916384
```

Рисунок 4.15 - результати обробки даних навантажувального тестування для сайту http://comsvit.com.ua/

```
Profile8.page_9: http://in.skyline.kh.ua/[1]
for fool interval
count_users,x1,x2,x3,x4
175,15.1593,15.1593,0.983081,1.03844
for intervals size 15
x1,x2,x3,x4
5-75,5.05833,5.05833,0.993028,0.713554
10-80,5.78233,5.78233,0.993788,0.743
15-85,6.515,6.515,0.994415,0.756732
20-90,7.305,7.305,0.993928,0.781643
25-95,8.16833,8.16833,0.990563,0.834589
30-100,8.99033,8.99033,0.991376,0.850964
35-105,9.94067,9.94067,0.987666,0.907589
40-110,10.884,10.884,0.988017,0.948446
45-115,12.042,12.042,0.976542,1.04498
50-120,13.033,13.033,0.978209,1.06266
55-125,13.9683,13.9683,0.98087,1.08057
60-130,14.913,14.913,0.979907,1.07384
65-135,15.912,15.912,0.978387,1.0575
70-140,16.9547,16.9547,0.98183,1.10821
75-145,17.9227,17.9227,0.977535,1.07614
80-150,19.2633,19.2633,0.969522,1.13786
85-155,20.4467,20.4467,0.970119,1.16036
90-160,21.8267,21.8267,0.965925,1.22
95-165,23.2633,23.2633,0.962236,1.30196
100-170,24.3033,24.3033,0.954247,1.26
105-175,25.8567,25.8567,0.949217,1.34679
```

Рисунок 4.16 - результати обробки даних навантажувального тестування для сайту http://in.skyline.kh.ua/

Далі ми підставляємо ці данні до наших нечітких правил і робимо висновки. Результати по кожному з сайтів приведені далі:

1) для http://www.microsoft.com/ вихідним значенням при кількості 175 віртуальних користувачів буде 80.8 тобто сайт гарно витримує нагрузку;

2) для http://snu.edu.ua/ вихідним значенням при кількості 175 віртуальних користувачів буде 16.3 тобто це погано, і необхідно провести додатковий аналіз по інтервалам. При кількості 75 віртуальних користувачів значення ефективності буде також 16.3 тобто погано. Можна зробити висновок що сайт написано погано, на ньому скоріш за все не має хешування, а час відклику росте з кількістю користувачів;

3) для https://snu.edu.ua/rasp/ вихідним значенням при кількості 175 віртуальних користувачів буде 33.6.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

5.1 Аналіз потенційних небезпечних і шкідливих виробничих чинників проектованого об’єкту, що мають вплив на персонал

У даному дипломному проекті розробляється програмне забезпечення. Розроблене програмне забезпечення орієнтоване на роботу з персональним комп'ютером. Експлуатовані для вирішення внутрішньовиробничих завдань ПЕОМ типу IBM PC мають наступні характеристики:

споживана потужність	220 Вт;
робоча напруга	220 В;
напруга джерел живлення	+12 В; - 12 В; +5 В;
робоча частота	50 Гц.

Виходячи з приведених характеристик, вочевидь, що для людини існує небезпека поразки електричним струмом, унаслідок недбалого поводження з комп'ютером і порушення правил експлуатації, залишення частин ПЕОМ, що знаходяться під напругою, відкритими або знятих для ремонту вузлів.

Відповідно до [13] до легкої фізичної роботи відносяться всі види діяльності, виконувані сидячи і ті, що не потребують фізичної напруги. Робота користувача ПК відноситься до категорії 1а.

При роботі на ПЕОМ користувач піддається ряду потенційних небезпек. Унаслідок недотримання правил техніки безпеки при роботі з машиною (невиконання огляду відкритих частин ПЕОМ, що знаходяться під напругою або знятих для ремонту вузлів) для користувача існує небезпека поразки електричним струмом.

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;
- блоки ПЕОМ і друку, що знаходяться в ремонті.

Ще одна проблема полягає у тому, що спектр випромінювання комп'ютерного монітора включає рентгенівську, ультрафіолетову і інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння мала, оскільки цей вид випромінювання поглинається речовиною екрану. Проте велику увагу

слід приділяти біологічним ефектам низькочастотних електромагнітних полів (аж до порушення ДНК).

Відповідно до [14], при обслуговуванні ПЕОМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

– підвищене значення напруги в електричному ланцюзі, замикання якої може відбутися через тіло людини;
– підвищений рівень статичної електрики;
– підвищений рівень електромагнітних випромінювань;
– підвищена або знижена температура повітря робочої зони;
– підвищений або знижений рух повітря;
– підвищена або знижена вологість повітря;
– відсутність або недостатність природного світла;
– підвищена пульсація світлового потоку;
– недостатня освітленість робочого місця;
– підвищений рівень шуму на робочому місці;
– розумове перенапруження;
– емоційні навантаження;
– монотонність праці.

5.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм чинить на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Тяжкість поразки людини електричним струмом залежить від цілого ряду чинників:

– значення сили струму;
– електричного опору тіла людини і тривалості протікання через нього струму;

– роду і частоти струму;

– індивідуальних властивостей людини і навколишнього середовища.

Розроблений дипломний проект передбачає наступні технічні способи і засоби, що застерігають людину від ураження електричним струмом:

– заземлення електроустановок;

– занулення;

– захисне відключення;

– електричне розділення мережі;

– використовування малої напруги;

– ізоляція частин, що проводять струм;

– огорожа електроустановок.

Занулення зменшує напругу дотику і обмежує години, протягом яких людина, ткнувшись до корпусу, може потрапити під дію напруги.

Струм однофазного короткого замикання визначається по наближеній формулі:

$$I_K = \frac{\text{Uф}}{\text{Zп} + \frac{\text{Zт}}{3}}\ , \qquad (5.1)$$

де Uф - номінальна фазна напруга мережі, В;

Zп - повний опір петлі, створене фазними і нульовими дротами, Ом;

Zт - повний опір струму короткого замикання на корпус, Ом.

Згідно таблиці 4 [15]: Zт /3 = 0,1 Ом.

Для провідників і жил кабелю для розрахунку повного опору петлі використовуємо формулу(5.2.) :

$$\text{Zп} = \sqrt{\text{Rп}^2 + \text{Xп}^2}\ , \qquad (5.2)$$

де Rп = Rф + Ro - сумарний активний опір фазного Rф і нульового Ro дротів, Ом;

Xп - індуктивний опір паяння дротів, Ом.

Перетин 1 км мідного дроту S = 2.5 мм, тоді згідно таблицям 5 і 6 [22], має такий опір:

Xп = 0,11 Ом;

Rф = 7,55 Ом;

Ro = 7,55 Ом.

Отже, Rп = 7,55 + 7,55 = 15,1 Ом.

Тоді по формулі (5.2) знаходимо повний опір петлі :

$$Zп = \sqrt{15,1^2 + 0,11^2} \approx 15,1 \text{ (Ом).}$$

Струм однофазного короткого замикання рівний:

$$Iк = \frac{220}{15,1 + 0,1} = 14,47 \text{ (А).}$$

Дія плавкої вставки на ПЕОМ забезпечується, якщо виконується співвідношення:

$$Iк \geq k * Iн, \quad (5.3)$$

де Iн - номінальний струм спрацьовування плавкої вставки, А;

k - коефіцієнт кратності нелінійного струму Iн, А.

Коефіцієнт кратності нелінійного струму Iн розраховується по формулі (5.4.) :

$$Iн = P / U, \quad (5.4)$$

де P = 220 Вт - споживана потужність;

U = 220 В - робоча напруга;

k = 3 А - для плавких вставок.

Отже, Iн = 220 / 220 = 1 А.

Підставивши значення у вираз (5.3), одержимо:

$$14,47 > 3*1.$$

Таким чином, доведено, що апарат забезпечить спрацьовування(і захист) при підвищенні номінального струму.

5.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Вимоги до виробничих приміщень встановлюються [23], ДБН, відповідними ГОСТами і ОСТамі з урахуванням небезпечних і шкідливих чинників, що утворюються в процесі експлуатації електроустаткування.

Підвищення працездатності людини і збереження її здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень визначається діючими на організм людини поєднаннями температури, вологості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводити до порушення систем кровообігу, нервової і потовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

Відповідно до [13] встановлюють оптимальну і допустиму температуру, відносну вологість і швидкість руху повітря в робочій зоні . За відсутності надмірного тепла, вологи, шкідливих речовин в приміщенні досить природної вентиляції.

У приміщенні для виконання робіт операторського типу(категорія 1а), пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (табл.5.1).

Таблиця 5.1 - Санітарні норми мікроклімату робочої зони приміщень для робіт категорії 1а.

Пора року	Температура, С	Відносна вологість, %	Швидкість руху повітря, м/с
Холодна	22…24	40…60	0,1
Тепло	23…25	40…60	0,1

У приміщенні, де знаходиться ПЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (з пристроєм вентиляційних каналів в перекриттях будівлі і вертикальних шахт) й установленого промислового кондиціонера фірми Mitsubishi, який дозволяє вирішити переважну більшість завдань по створінню та підтримці необхідних параметрів повітряного середовища. Цей метод забезпечує приток потрібної кількості свіжого повітря, визначеного в ДБН (30 $м^3$ в годину на одного працівника).

Шум на виробництві має шкідливу дію на організм людини. Стомлення операторів через шум збільшує число помилок при роботі, призводить до виникнення травм. Для

оператора ПЕОМ джерелом шуму є робота принтера. Щоб усунути це джерело шуму, використовують наступні методи. При покупці принтера слід вибирати найбільш шумозахисні матричні принтери або з великою швидкістю роботи(струменеві, лазерні). Рекомендується принтер поміщати в найбільш віддалене місце від персоналу, або застосувати звукоізоляцію та звукопоглинання(під принтер підкладають демпфуючі підкладки з пористих звукопоглинальних матеріалів з листів тонкої повсті, поролону, пеноплену).

При роботі на ПЕОМ, проектом передбачені наступні методи захисту від електромагнітного випромінювання : обмеження часом, відстанню, властивостями екрану.

Обмеження години роботи на ПЕОМ складає 3,5-4,5 години. Захист відстанню передбачає розміщення монітора на відстані 0,4-0,5 м від оператора. Передбачений монітор 20" TFT, Samsung 2043BW відповідає вимогам стандарту TCO'03.

TCO'03 пред’являє жорсткі вимоги в таких областях: ергономіка (фізична, візуальна і зручність користування), енергія, випромінювання (електричних і магнітних полів), навколишнє середовище і екологія, а також пожежна та електрична безпека, які відповідають всім вимогам [16].

Для зниження стомлюваності та підвищення продуктивності праці обслуговуючого персоналу в колірній композиції інтер'єру приміщень для ПЕОМ дипломним проектом пропонується використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

У проекті передбачається використання сумісного освітлення. У світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Як штучне освітлення необхідно використовувати штучне робоче загальне освітлення. Для загального освітлення необхідно використовувати люмінесцентні лампи. Вони володіють наступними перевагами: високою світловою віддачею, тривалим терміном служби, хоча мають і недоліки: високу пульсацію світлового потоку.

При експлуатації ПЕОМ виробляється зорова робота. Відповідно до [20] ця робота відноситься до розряду 5а. При цьому нормоване освітлення на робочому місці(Ен) при загальному освітленні рівна 200 лк.

Приміщення завдовжки 12 м, шириною 10 м, заввишки 4 м обладнується світильниками типу ЛП02П, оснащеними лампами типу ЛБ зі світловим потоком 3120 лм кожна.

Виконаємо розрахунок кількості світильників в робочому приміщенні завдовжки a=12 м, шириною b=10 м, заввишки з=4 м, використовуючи формулу (5.5) розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку:

$$n = (E \cdot S \cdot Z \cdot k)/(F \cdot U \cdot M), \quad (5.5)$$

де F - світловий потік = 3120 лм;

E - максимально допустима освітленість робочих поверхонь = 200 лк;

S - площа підлоги = 120 м2;

Z - поправочний коефіцієнт світильника = 1,2;

k - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників = 1,5;

n - кількість світильників;

U - коефіцієнт використовування освітлювальної установки = 0,6;

M - кількість ламп у світильнику =2.

Отже, $n = (200 \cdot 120 \cdot 1{,}2 \cdot 1{,}5)/(3120 \cdot 0{,}6 \cdot 2) = 12$.

Виходячи з цього, рекомендується використовувати 12 світильників. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. 5.1.

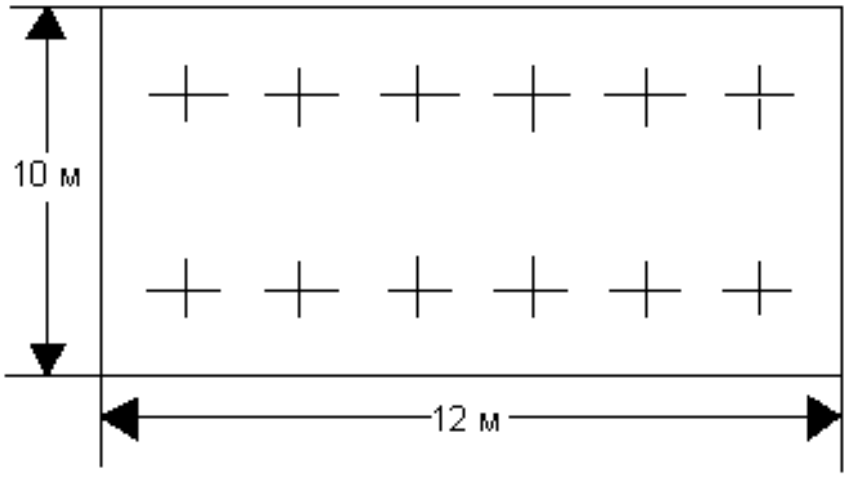


Рисунок 5.1 - Схема розташування світильників

5.4 Рекомендації по пожежній безпеці

Пожежі в приміщеннях, де встановлена обчислювальна техніка, представляють небезпеку для життя людини. Пожежі також пов'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що у свою чергу спричиняє за собою порушення ходу технологічного процесу.

Пожежа може виникнути при наявності горючої речовини та внесення джерела запалювання в горюче середовище. Пальними матеріалами в приміщеннях, де розташовані ПЕОМ, є:

– поліамід - матеріал корпусу мікросхеми, горюча речовина, температура самозаймання аерогелю 420 °С ;

– полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, температура самозаймання 530 °С, кількість енергії, що виділяється при згоранні - 18000 - 20700 кДж/кг;

– стеклотекстоліт ДЦ - матеріал друкарських плат, важкозаймистий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

– пластика кабельний №489 - матеріал ізоляції кабелю, горючий матеріал, показник горючості більш 2.1;

– деревина - будівельний і обробний матеріал, матеріал з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, теплота згорання 18731 - 20853 кДж/кг, температура запалювання 399 °С, схильна до самозаймання.

Згідно [29] приміщення відносяться до категорії В (пожежовибухонебезпечним) і згідно правилам побудови електроустановок простір усередині приміщення відноситься до вогненебезпечної зони класу П - IIа (зони, розташовані в приміщеннях, в яких зберігаються тверді горючі речовини).

Потенційними джерелами запалення при роботі ПЕОМ є:

- іскри при замиканні і розмиканні ланцюгів;
- іскри і дуги коротких замикань;
- перегріви від тривалого перевантаження і наявності перехідного опору.

Продуктами згорання, що виділяються при пожежі, є : оксид вуглецю, сірчистий газ, оксид азоту, синильна кислота, акропеін, фосген, хлор та ін. При горінні пластмас, окрім звичайних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол та ін., що шкідливо впливають на організм людини.

Для захисту персоналу від дії небезпечних і шкідливих чинників пожежі проектом передбачається застосування промислового протигаза з коробкою марки В(жовта).

Пожежна безпека об’єктів народного господарства регламентується [17] і забезпечується системами запобігання пожежам і протипожежному захисту. Для успішного гасіння пожеж вирішальне значення має швидке виявлення пожежі і своєчасний виклик пожежних підрозділів до місця пожежі.

Зменшити горюче навантаження не представляється можливим, тому проектом передбачається застосувати наступні способи і їх комбінації для запобігання утворенню(внесення) джерел запалення :

– застосування устаткування, що задовольняє вимогам електростатичної безпеки;

– застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалення;

– виключення можливості появи іскрового заряду статичної електрики в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення;

– підтримка температури нагріву поверхні машин, механізмів, устаткування, пристроїв, речовин і матеріалів, які можуть увійти до контакту з пальним середовищем, нижче гранично допустимої, становить 80% якнайменшої температури самозаймання пального.

– заміна небезпечних технологічних операцій більш безпечними;

– ізольоване розташування небезпечних технологічних установок і устаткування;

– зменшення кількості пальних і вибухонебезпечних речовин, що знаходяться у виробничих приміщеннях;

– запобігання можливості утворення пальних сумішей на лінії, вентиляційних системах і ін.;

– механізація, автоматизація та справність(потокова) виробництва;

– суворе дотримання стандартів і точне виконання встановленого технологічного режиму;

– запобігання можливості появи в небезпечних місцях джерел запалення;

– запобігання розповсюдженню пожеж і вибухів;

– використовування устаткування і пристроїв, при роботі яких не виникає джерел запалення;

– виконання вимог сумісного зберігання речовин і матеріалів;

– наявність громовідводу;

– ліквідація можливості самозаймання речовин і матеріалів .

Для запобігання пожежі в обчислювальних центрах проектом пропонується виконання наступних вимог :

– електроживлення ЕОМ повинно мати автоматичне блокування відключення електроенергії на випадок зупинки системи охолоджування і кондиціонування;

– система вентиляції обчислювальних центрів повинна бути обладнана блокуючими пристроями, що забезпечують її відключення на випадок пожежі;

– робочі місця повинні бути оснащені пожежними щитами, сигналізацією, засобами для сповіщення про пожежну небезпеку (телефонами), медичними аптечками для надання першої медичної допомоги, розробленим планом евакуації.

Для зниження пожежної небезпеки в приміщеннях використовуються первинні засоби гасіння пожеж, а також система автоматичної пожежної сигналізації, яка дозволяє знайти початкову стадію загоряння, швидко і точно оповістити службу пожежної охорони про час і місце виникнення пожежі.

Відповідно до правил пожежної безпеки для промислових підприємств приміщення категорії В підлягають устаткуванню системами автоматичної пожежної сигналізації. Проектом передбачається застосування датчика типу ІДФ - 1(димовий фотоелектричний датчик), оскільки специфікою пожеж обчислювальної техніки і радіоапаратури є, в першу чергу, виділення диму, а потім - підвищення температури.

При виникненні пожежі в робочому приміщенні обслуговуючий персонал зобов'язаний негайно вжити заходи по ліквідації пожежі. Для ліквідації пожежі використовують вогнегасники (хімічно-пінні, пінні для повітря ОП-5, ОП-6, ОП-9, вуглекислотні ОУ-5), пісок, пожежний інвентар (сокири, ломи, багри, шерстяну або азбестову ковдри). Як засіб індивідуального захисту проектом передбачається використовування промислового протигаза з маскою, фільтруючої коробки В.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу правилам пожежної безпеки.

5.5 Вплив на навколишнє середовища

В даний час зростає кількість комп'ютерної техніки в усіх галузях діяльності людини. Багато користувачів і виробників помиляються, вважаючи, що зі зменшенням і удосконаленням комп'ютерів, зменшиться їх негативний вплив на навколишнє середовище.

На даний момент найбільш суворим з існуючих світових стандартів екологічності для комп'ютерної техніки є стандарт TCO-99. У порівнянні з попередніми він містить додаткові обмеження по частині екології, ергономіки, енергоспоживання і емісії пристроїв.

Організація по захисту навколишнього середовища Greenpeace з 2006 року оцінює виробників електроніки за кількістю важких металів і отруйних речовин, наприклад інгібіторів горіння, використовуваних ними при виробництві (інгібітор - речовина, присутність якого в невеликих кількостях призводить до запобігання або уповільнення процесів горіння або корозії; інгібітори знижують швидкість хімічних реакцій або пригнічують їх). Однак навіть оцінки такої організації, як Greenpeace, не можуть претендувати на об'єктивність. Адже в одних випадках вона використовує перевірену інформацію, що стосується, наприклад, заходів щодо утилізації відходів, а в інших спирається тільки на дані виробника. А якщо компанія не повідомляє ніяких відомостей, то автоматично опиняється на нижніх рядках рейтингу. Крім того, енергетичні витрати на виробництво і перевезення продукції також необхідно враховувати при оцінці екологічної ефективності. Адже часи, коли техніка виготовлялася тільки на одному заводі, давно пройшли. Сьогодні окремі комплектуючі закуповуються на різних підприємствах по всьому світу, після чого здійснюється складання пристроїв. Тому найчастіше навіть самі компанії не можуть знати, які шкідливі речовини потрапляють в атмосферу при виготовленні їх продукції і які саме метали або токсини в ній містяться.

ЖК-екрани - один з джерел парникових газів, які набагато шкідливіше діоксиду вуглецю. Рідкокристалічні монітори швидко знайшли популярність, прийшовши на зміну громіздким ЕПТ-моделям. І це не дивно, адже вони мають тонкі корпуса і споживають значно менше електроенергії. За іншим аспектам екологічної безпеки дисплеї на основі рідких кристалів також вважалися проривом, тому що в них не використовувався газ, що містить свинець. Досить довго ніхто не звертав уваги на застосовуваний для чищення РК-панелей трехфторістий азот (NF3), і тільки в середині 2008 року вченими було доведено наявність даної хімічної речовини в атмосфері. Відкриття було вражаючим: порівняно з діоксидом вуглецю (CO2) NF3 має в 17 000 разів більше активного парникового газу, а його атмосферний час напіврозпаду може складати від 550 до 740 світлових років (у CO2 - від 30 до 40 років). Закону, який обмежував би рівень викиду NF3, поки не існує.

Виявлення енерговитрат є таким же проблематичним процесом, як і визначення кількості матеріалів, придатних для вторинної переробки, і важких металів, що містяться в пристроях. Таким чином, надійним показником екологічності залишається тільки рівень енергоспоживання.

Полівінілхлорид, що позначається зазвичай абревіатурою ПВХ, - це різновид пластику, що застосовується в самих різних цілях. З нього зроблена зовнішня оболонка кабелів, якими з'єднуються пристрої, він оточує електричний провід портативного комп'ютера. Це дешевий, міцний і вельми поширений матеріал. Разом з тим, за словами IT- аналітика «Грінпіс» Кейсі Харрелл, «ПВХ - найгірший з пластиків». Він є причиною виникнення гормонального дисбалансу, проблем в репродуктивної сфері та різних форм раку. Полівінілхлорид практично неможливо правильно утилізувати. Внаслідок старий матеріал виявляється зазвичай на звалищі з відходами або, того гірше, спалюється з метою вилучення мідних жил і інших цінних компонентів. При його згорянні утворюється вкрай шкідливий канцерогенний діоксин. Звалища і хімічні поховання забруднюють джерела води. Єдиний спосіб правильно утилізувати ПВХ полягає в тому, щоб відправити його в центр небезпечних відходів.

Залишається лише сподіватися, що настане час, коли технології будуть допомагати людині, не завдаючи незворотної шкоди здоров'ю навколишнього середовища.

ВИСНОВКИ

У даній роботі було проведено навантажувальне тестування сайтів та проаналізовані отриманні результати.

Аналіз показав, що не всі сайти гарно справляються з максимальними навантаженнями та виконують поставлені норми продуктивності. Були замічені й проаналізовані проблемні компоненти сайту. При більших навантаженнях з'являються можливі проблеми із сервером баз даних, які збільшують час завантаження сторінок сайту.

Детально розписана схема запропонованого методу навантажувального тестування. На основі отриманих графіків та числових рядів, були сформовані основні метрики за якими проводиться аналіз. Були отримані статичні дані та побудована система правил., що дозволяє в подальшому для невідомих сайтів проводити автоматично аналіз та діагностування. Визначили слабкі місця системи та сформували рекомендації та пропозиції для підвищення продуктивності роботи сайту

Була проаналізована проблема недостатньої надійності програмного забезпечення, що готується до випуску компанією по виробництву програмного забезпечення. У процесі аналізу в многокритериальном просторі було розроблено кілька сценаріїв рішення проблеми. Показано, що найбільш прийнятним сценарієм рішення є проведення навантажувального тестування з метою виявити проблемні місця й помилки розроблювальної системи.

У розділі «Охорона праці» виконано аналіз потенційних небезпек при роботі із засобами обчислювальної техніки і механізмами, розроблені заходи щодо техніки безпеки, заходи, які забезпечують виробничу санітарію і гігієну праці, розраховане штучне освітлення, виконані рекомендації по пожежній безпеці, розглянутий можливий вплив на навколишнє середовище.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

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

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

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

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

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

6) ANSI/IEEE Std 829-1998. IEEE Standart Glossary for Software Test Documentation [Текст]. - IEEE Computer Society Press, 1998.

7) Методичні вказівки до виконання розділу "Безпеку життя і діяльності людини" у дипломних проектах (роботах) для студентів усіх форм навчання інституту “Комп’ютерних інформаційних технологій” [Текст] / Упоряд.: В.А.Айвазов, Н.Л.Березуцька, Б.В.Дзюндзюк, А.В.Мамонтів, Т.Є.Стиценко. - Харків: ХНУРЕ, 2003. - 56 с.

8) Леоненков, Ф. Нечёткое моделирование в среде MATLAB и fuzzyTECH [Текст] / Ф.Леоненков – СПб.:БХВ-Петербург, 2005. – 736 с.

9) Охорона праці в галузі. Конспект лекцій для студентів усіх спеціальностей інституту інформаційні компьютерні технології [Текст]:/ Упоряд. В.А. Айвазов, Б.В. Дзюндзюк, Т.Е. Стиценко. - Харків: ХТУРЕ, 1999. - 82с.

10) Леоненков, А.В. Нечёткое моделирование в среде MATHLAB и fuzzyTECH [Текст] / А.В. Леоненков – СПб.: БХВ-Петербург, 2003.

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

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

13) ГОСТ 12.1.005-88. Міждержавний стандарт. Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони

14) ГОСТ 12.0.003-74 Небезпечні і шкідливі виробничі фактори. Класифікація

15) ДСТУ 7237:2011 Національний стандарт України. Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту

16) ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

17) ГОСТ 12.1.004-91. Пожежна безпека. Загальні вимоги .

18) ДБН В.2.5-67. Опалення вентиляція та кондиціонування.

19) ГОСТ 12.1.006-84. Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю

20) ДБН В.2.5-28-2006. Природне і штучне освітлення.

21) ГОСТ 12.4.009-83. Пожежна техніка для захисту об'єктів. Основні види. Розміщення і обслуговування.

22) ДСТУ Б В.1.1-36-2016. Визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною безпекою.

23) ДСП 173-96. Державні санітарні правила планування та забудови населених пунктів

24) Симметрон. Электронные компоненты. Каталог 2002, 2002г. – 192с.

ДОДАТОК А.
Електронні плакати

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТУ ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМПЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

Аналіз навантажувального тестування інформаційної системи

Виконав:
ст.гр. КН-17зм
Торопов О.М.

Керівник:
доц.Недзельський Д.О.

Актуальність роботи

Більшість сучасних компаній мають власні веб-сайти і веб-додатки, від яких безпосередньо залежить функціональність фірми.

Проблеми, пов'язані з недостатньою продуктивністю веб-серверів, ведуть до відмови клієнтів від використання веб-додатків.

У зв'язку з цим необхідне проведення навантажувального тестування, з метою поліпшення продуктивності веб-серверів.

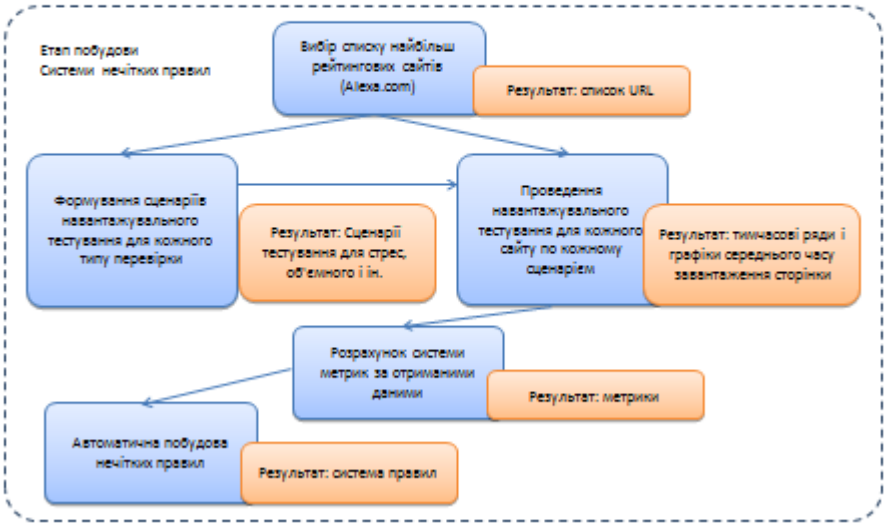
В результаті навантажувального тестування можна розрахувати досить велика кількість показників, але для того, щоб в подальшому видати рекомендації щодо поліпшення якості інформаційної системи, аналітику необхідно буде проаналізувати досить великий обсяг інформації.

Саме тому запропонований підхід по автоматизації процесу обробки результатів тестування навантаження.

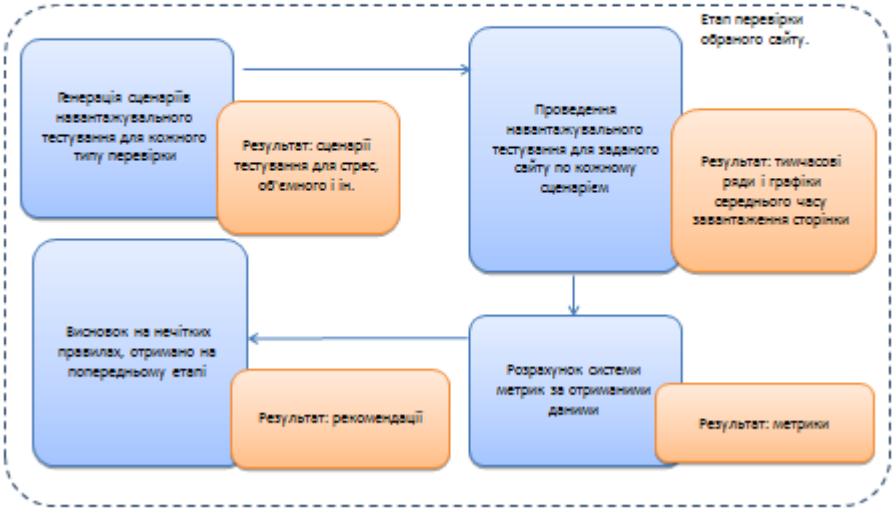
Результати навантажувального тестування

Постановка задачі

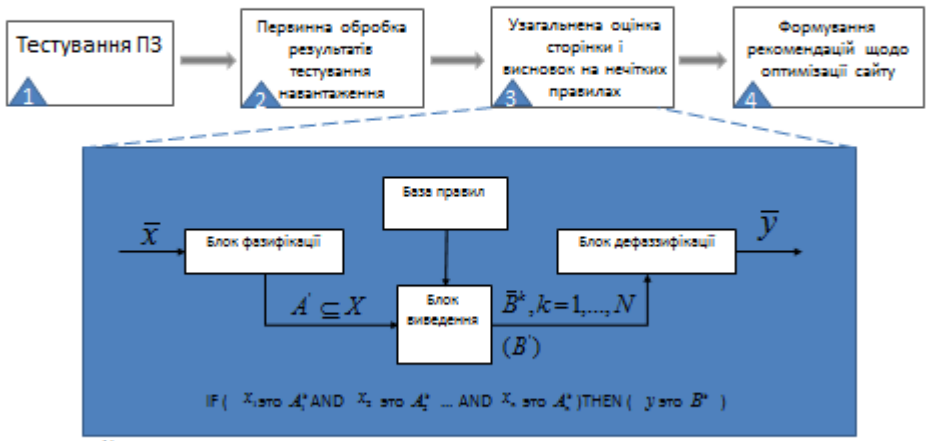
Перший етап



Другий етап



Метод автоматизації обробки результатів тестування навантаження програмного забезпечення



N – количество нечетких правил;
A_i^k – входные данные;
B_i^k – выходные данные;

$x_1, x_2, ..., x_n$ – входные переменные;
$y_1, y_2, ..., y_n$ – выходные переменные.

10

Аналіз результатів

Нечіткі правила

x_1 – час завантаження з елементами сторінки;

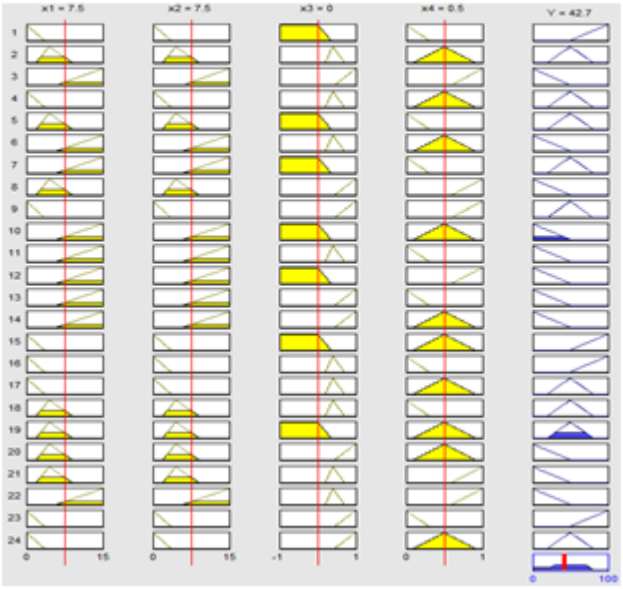
x_2 - час завантаження без елементів сторінки;

x_3 - залежність від кількості користувачів на сайті;

x_4 - швидкість росту часу завантаження сторінки.

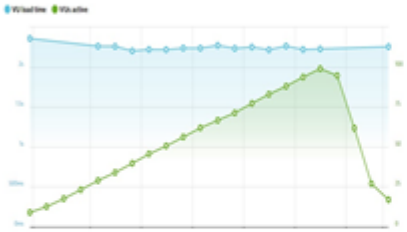
1. If (x1 is Well) and (x2 is Well) and (x3 is Well) and (x4 is Well) then (Y is Well) (1)
2. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
3. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Bad) then (Y is Bad) (1)
4. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
5. If (x1 is Good) and (x2 is Good) and (x3 is Well) and (x4 is Well) then (Y is Good) (1)
6. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Good) then (Y is Bad) (1)
7. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Well) then (Y is Good) (1)
8. If (x1 is Good) and (x2 is Good) and (x3 is Bad) and (x4 is Bad) then (Y is Bad) (1)
9. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Bad) then (Y is Good) (1)
10. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Good) then (Y is Bad) (1)
11. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Well) then (Y is Bad) (1)
12. If (x1 is Bad) and (x2 is Bad) and (x3 is Well) and (x4 is Bad) then (Y is Bad) (1)
13. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Well) then (Y is Bad) (1)
14. If (x1 is Bad) and (x2 is Bad) and (x3 is Bad) and (x4 is Good) then (Y is Bad) (1)
15. If (x1 is Well) and (x2 is Well) and (x3 is Well) and (x4 is Good) then (Y is Well) (1)
16. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Well) then (Y is Well) (1)
17. If (x1 is Well) and (x2 is Well) and (x3 is Good) and (x4 is Good) then (Y is Good) (1)
18. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Well) then (Y is Good) (1)
19. If (x1 is Good) and (x2 is Good) and (x3 is Well) and (x4 is Good) then (Y is Good) (1)
20. If (x1 is Good) and (x2 is Good) and (x3 is Bed) and (x4 is Good) then (Y is Bad) (1)
21. If (x1 is Good) and (x2 is Good) and (x3 is Good) and (x4 is Bad) then (Y is Bad) (1)
22. If (x1 is Bad) and (x2 is Bad) and (x3 is Good) and (x4 is Bad) then (Y is Bad) (1)
23. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Well) then (Y is Well) (1)
24. If (x1 is Well) and (x2 is Well) and (x3 is Bad) and (x4 is Good) then (Y is Good) (1)

Графічне відображення нечітких правил



Аналіз результатів тестування навантаження

Snu.edu.ua

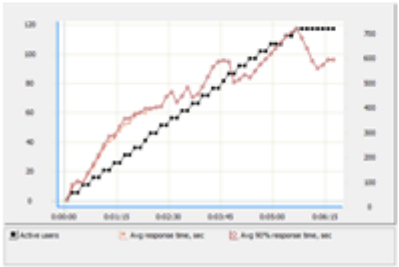


Результати тестування сайту:

- *кореляція користувачів і часу 0,0029 ;*
- *максимальне відносне час виконання 1,80 сек ;*
- *швидкість росту часу 0,0062;*
- *середній час завантаження сторінки 1,7 сек ;*
- *середній час завантаження з елементами сторінки 1,7 сек.*

- *Висновок: у- Good*

www.kture.kharkov.ua



Результати тестування сайту:

- *кореляція користувачів і часу 0,00 ;*
- *максимальне відносне час виконання 120 сек ;*
- *швидкість росту часу 0,032;*
- *середній час завантаження сторінки 45,4 сек ;*
- *середній час завантаження з елементами сторінки 60,4.*

- *Висновок: у- Bad*

Висновки

- *Сформульовано задачу автоматизації оцінки результатів тестування навантаження програмного забезпечення.*
- *Розроблено систему показників якості процесу створення програмного забезпечення.*
- *На основі нечіткої логіки запропонований метод автоматизації оцінки результатів тестування навантаження програмного забезпечення.*
- *Побудований висновок і отримані рекомендації щодо поліпшення якості інформаційної системи.*