**Національний технічний університет України**

**«Київський політехнічний інститут**

**ім. ІГОРЯ СІКОРСЬКОГО»**

Інститут телекомунікаційних систем

(повна назва інституту/факультету)

Кафедра «Інформаційно-телекомунікаційних мереж»

(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ Л. С. Глоба

(підпис) (ініціали, прізвище)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2017 р.

**Дипломна робота**

**на здобуття ступеня бакалавра**

з напряму підготовки 6.050903 «Телекомунікації»

(код і назва)

на тему: Розподілені обрахунки методом MapReduce на мобільній платформі.

Виконав (-ла): студент (-ка) 4-го курсу, групи ТІ-32

(шифр групи)

Павленко Владислав Миколайович\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові) (підпис)

Керівник Доцент , к.т.н., старший науковий співробітник,\_\_\_\_\_\_\_\_\_\_

Алєксєєв М. О.

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2017 року

**Національний технічний університет України**

**«Київський політехнічний інститут ім. Ігоря Сікорського»**

Інститут телекомунікаційних систем

(повна назва)

Кафедра «Інформаційно-телекомунікаційних мереж»

(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050903 «Телекомунікації»

(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ Л.С.Глоба

(підпис) (ініціали, прізвище)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 р.

**ЗАВДАННЯ**

**на дипломну роботу студенту**

Павленку Владиславу Миколайовичу

(прізвище, ім’я, по батькові)

1. Тема роботи: Розподілені обрахунки методом MapReduce на мобільній платформі ,

керівник роботи Алєксєєв Микола Олександрович ,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07» квітня 2017 р. №1290-с

2. Термін подання студентом роботи 14 червня 2017 р.

3. Вихідні дані до роботи:

1. Інформація про вже створені системи розподілених обрахунків.
2. Принципи розподіленої роботи гетерогенних систем
3. Порівняльні статистики алгоритмів розподіленої роботи

3. Зміст роботи

1. Провести аналіз вже створених систем розподілених обрахунків для персональних комп’ютерів.
2. Запропонувати модель управління завданнями для розподіленого обрахування.
3. Розглянути особливості застосування запропонованої моделі для мобільних пристроїв.

4. Розробити та розгорнути макет робочого серверу по розподіленню завдань та клієнтський додаток для виконання обрахунків.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

5. Дата видачі завдання 10 вересня 2016 року.

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  дипломної роботи | Термін виконання  етапів роботи | Примітка |
| 1 | Дослідження та вивчення отриманого завдання | 10.09.16 – 01.10.16 |  |
| 2 | Аналіз методології побудов систем розподілених обрахунків | 01.10.16 – 09.11.16 |  |
| 3 | Аналіз існуючих рішень розподілених обрахунків | 09.11.16 – 12.01.17 |  |
| 4 | Аналіз підходів до розробки систем розподілених обрахунків | 12.01.17 – 16.02.17 |  |
| 5 | Аналіз підходів до побудови гетерогенних систем | 17.01.17 – 01.02.17 |  |
| 6 | Підготовка тезисів (ПТ-17) | 01.02.17 – 01.03.17 |  |
| 7 | Створення прототипу розподіленої системи, опис компонентів | 01.03.17 – 01.04.17 |  |
| 8 | Розробка алгоритму балансування задач між мобільними пристроями | 01.11.16 – 01.03.17 |  |
| 9 | Розробка алгоритму визначення закінчення роботи над завданням | 01.03.17 – 26.03.17 |  |
| 10 | Розробка розподіленої системи | 01.11.16 – 01.05.17 |  |
| 11 | Підготовка тексту диплому | 15.04.17 – 16.06.17 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)

**РЕФЕРАТ**

Робота містить 53 сторінки, 27 рисунків та 2 таблиць. Було використано 15 джерел

**Мета роботи**: Запропонувати модель гетерогенної системи розподілених обрахунків для використання на мобільних платформах.

Проведено аналіз алгоритмів балансування задач, розглянуто переваги та приклади їх використання в існуючих програмних продуктах.

Розробити модель розподіленої системи враховуючи особливості реалізації на мобільних платформах. Сформовано порівняльну характеристику різних алгоритмів розрахунку.

Створити модель для демонстрування роботи розробленої системи та використаних алгоритмів.

**Ключові слова:** розподілені обрахунки, гетерогенна система, алгоритм балансування завдань.

**ABSTRACT**

The work contains 53 pages, 27 figures and 2 tables, 15 sources have been used.

**Objective**: Suggest distributed heterogeneous system model calculations for using on mobile platforms.

The analysis of balancing algorithms, discussed advantages and examples of their use in existing software products.

Develop distributed computing model including analyzed information using the principles of distributed calculation. Formed a comparison characteristics of different calculation algorithms.

Created model for the demonstration of the developed system and implemented algorithms.

**Keywords**: distributed calculations, heterogeneous system, algorithm balancing tasks.

Зміст

[ВСТУП 10](#_Toc454016370)

[РОЗДІЛ 1 Аналіз методів оцінки стану людини за психологічними показниками 13](#_Toc454016371)

[1.1 Методи оцінки психофізіологічного стану людини 13](#_Toc454016372)

[1.1.1 Метод кольорової преференції 15](#_Toc454016373)

[1.1.2 Модифікований метод Дембо - Рубінштейн 16](#_Toc454016374)

[1.1.3 Метод комплексного дослідження психофізіологічних характеристик 17](#_Toc454016375)

[1.2 Проблематика сучасних досліджень стану людини 18](#_Toc454016376)

[1.3 Автоматизовані методи оцінки психофізіологічного стану людини 18](#_Toc454016377)

[1.3.1 Використання штучних нейронних мереж 18](#_Toc454016378)

[1.3.2 Використання експертних систем 20](#_Toc454016379)

[Висновки 22](#_Toc454016380)

[РОЗДІЛ 2 Інтелектуальна технологія оцінки психофізіологічного стану людини 24](#_Toc454016381)

[2.1 Вимоги до технології 24](#_Toc454016382)

[2.2 Структура інтелектуальної технології 25](#_Toc454016383)

[2.2.1 Середовище БЗ 26](#_Toc454016384)

[2.2.3 Підсистема прийняття рішень 27](#_Toc454016385)

[2.2.4 Підсистема вводу/виведення 28](#_Toc454016386)

[2.2.5 Підсистема обґрунтування 28](#_Toc454016387)

[Висновки 29](#_Toc454016388)

[РОЗДІЛ 3 Алгоритм визначення кольорової преференції 30](#_Toc454016389)

[Висновки 32](#_Toc454016390)

[РОЗДІЛ 4 Програмна реалізація інформаційної системи оцінки людини 33](#_Toc454016391)

[4.1 Постановка завдання та вимоги до системи 33](#_Toc454016392)

[4.2 Діаграма класів системи та їх опис 33](#_Toc454016393)

[4.3 Діаграма сутностей бази даних та їх опис 38](#_Toc454016394)

[4.4 Структура розробленої інформаційної системи 39](#_Toc454016395)

[4.5 Практичне застосування розробленої системи 41](#_Toc454016396)

[Висновки 48](#_Toc454016397)

[ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ 49](#_Toc454016398)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 50](#_Toc454016399)

[Додаток А Блок-схема методу визначення кольорової преференції 52](#_Toc454016400)

**перелік скорочень**

|  |  |
| --- | --- |
| MongoDB | Mongoose Data Base |
| JSON | JavaScript Object Notation |
| MVC | Model-View-Controller |
| SQL  Wi-Fi | Structured Query Language  Wireless Fidelity |
| БД | База даних |
| ПЗ | Програмне забезпечення |
| ПК | Персональний комп’ютер |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# ВСТУП

**Актуальність.** В сучасному світі з кожним днем зростає обсяг інформації яка потребує обробки. Розробка і використання розподілених систем дозволяє виконувати обрахунки над великими масивами даних паралельно, що значно пришвидшує отримання кінцевого результату.

Інформація має неоднорідний зміст і потребує різних методів обробки. Для впровадження цих методів в систему розподілених обрахунків існують різні підходи до їх побудови. На даний момент найпопулярнішою є технологія MapReduce яка дозволяє охопити найбільш широкий спектр типів завдань. За даним алгоритмом працюють багато розподілених систем, які мають неоднорідну структуру та абстрагуються від певних залежностей платформи на якій вони виконуються, що в свою чергу дозволяє вільно розповсюджувати ПЗ та з легкістю модифікувати його під деякі особливості поставленого на обробку завдання.

Сучасні мобільні телефони (планшети, плеєри, ультрабуки) мають потужні процесорні можливості що наближає їх до повноцінних ПК. Більшу частину працюючого часу, мобільний пристрій знаходиться в режимі очікування, який можна використовувати в корисних цілях. Через велику їх розповсюдженість і постійний зв’язок з Інтернетом за допомогою популярних Wi-Fi мереж або 3G, мобільні пристрої надають можливість об’єднувати їх у системи розподілених обрахунків. Беручи до уваги неоднорідність структури та різні ОС на пристроях, технологія MapReduce добре підходить для реалізації розподіленої системи.

**Об’єкт роботи:** мобільна платформа.

**Предмет роботи:** побудова системи розподілених обрахунків.

**Мета роботи**: підвищення ефективності розподілених систем за рахунок використання мобільних пристроїв.

Для досягнення мети було поставлено та вирішено такі **основні задачі:**

1. Провести аналіз принципів побудови систем розподілених обрахунків та виконати огляд існуючих зразків їх реалізації;
2. Дослідити алгоритми роботи та запропонувати технологію реалізації системи.
3. Розглянути особливості застосування запропонованої реалізації для мобільної платформи.

4. Розробити ПЗ для встановлення на мобільні пристрої та серверну частину системи для автоматизації роботи над завданнями.

5. Створити розподілену систему обрахунків з використанням розроблених додатків та серверу налаштованих в одній мережі.

**Теоретичний результат дослідження:**

1. Проведено аналіз принципів побудови систем розподілених обрахунків та досліджено можливість їх використання для обрахунку інформації.
2. Запропоновано алгоритм реалізації системи розподілених обрахунків із застосуванням технології MapReduce.
3. Розглянуто особливості побудови запропонованої системи для мобільних платформ .

**Практичний результат роботи:**

1. Розроблено систему розподілених обрахунків, яка базується на технології MapReduce.

2. Створено мобільний додаток для встановлення на мобільні платформи.

# РОЗДІЛ 1

# Провести аналіз принципів побудови систем розподілених обрахунків та виконати огляд існуючих зразків їх реалізації.

# 1.1 Розподілені обчислення. Мета та цілі їх застосування

Розподілені обчислення (розподілена обробка даних) — це спосіб розв'язання трудомістких обчислювальних завдань з використанням двох і більше комп'ютерів, об'єднаних в мережу. Розподілені обчислення є окремим випадком паралельних обчислень, тобто одночасного розв'язання різних частин одного обчислювального завдання декількома процесорами одного або кількох комп'ютерів. Тому необхідно, щоб завдання, що розв'язується було сегментоване — розділене на підзадачі, що можуть обчислюватися паралельно. При цьому для розподілених обчислень доводиться також враховувати можливу відмінність в обчислювальних ресурсах, які будуть доступні для розрахунку різних підзадач. Проте, не кожне завдання можна «розпаралелити» і прискорити його розв'язання за допомогою розподілених обчислень.

У цьому визначенні обмовляються два моменти. Перший відноситься до апаратури: всі машини автономні. Другий стосується програмного забезпечення: користувачі думають, що мають справу з єдиною системою. Важливо обидва моменти. Можливо, замість того щоб розглядати визначення, розумніше буде зосередитися на важливих характеристиках розподілених систем. Перша з таких характеристик полягає в тому, що від користувачів приховані відмінності між комп'ютерами і способи зв'язку між ними. Те ж саме відноситься і до зовнішньої організації розподілених систем. Іншою важливою характеристикою розподілених систем є спосіб, за допомогою якого користувачі і додатки одноманітно працюють в розподілених системах, незалежно від того, де і коли відбувається їх взаємодія.

Розподілені системи повинні також відносно легко піддаватися розширенню, або масштабуванню. Ця характеристика є прямим наслідком наявності незалежних комп'ютерів, але в той же час не указує, яким чином ці комп'ютери насправді об'єднуються в єдину систему. Розподілені системи зазвичай існують постійно, проте деякі їх частини можуть тимчасово виходити з ладу. Користувачі і додатки не повинні повідомлятися про те, що ці частини замінені або полагоджені або що додані нові частини для підтримки додаткових користувачів або додатків.

Для того, щоб підтримати представлення різних комп'ютерів і мереж у вигляді єдиної системи, організація розподілених систем часто включає додатковий рівень програмного забезпечення, що знаходиться між верхнім рівнем, на якому знаходяться користувачі і додатки, і нижнім рівнем, що складається з операційних систем, як показано на рис. 1. Відповідно, така розподілена система зазвичай називається системою проміжного рівня (middleware).



Рисунок 1

Метою використання розподілених обчислювальних систем є пришвидшення процесу обробки інформації методом дроблення задачі на частини та паралельного виконання їх на процесорах різних машин.

Поява й розвиток розподілених архітектур пов’язані з інтенсивним розвитком технічних і програмних засобів. У цих архітектурах функціональні компоненти інформаційної системи розподіляються по наявних вузлах залежно від поставлених цілей і завдань. Можна виділити шість основних характеристик архітектури розподілених систем:

1. **Спільне використання ресурсів**. Розподілені системи дозволяють спі­ль­не використання апаратних та програмних ресурсів, наприклад жорстких ди­сків, принтерів, файлів, компіляторів та інше, об’єднаних засобами мережі. Оче­видно, що розподіл ресурсів можливий і в багатокористувацьких системах, але в цьому випадку за надання ресурсів і їх керування повинен керувати цент­ральний процесор.
2. **Відкритість**. Це можливість розширювати систему шляхом добавля­ння нових ресурсів. Розподілені системи – це відкриті системи, до яких приєд­нуються апаратне і програмне забезпечення від різних виробників.
3. **Паралельність**. В розподілених системах декілька процесів можуть одночасно виконуватися на різних комп’ютерах в мережі. Ці процеси можуть взаємодіяти один з одним під час їх виконання.
4. **Масштабованість**. В принципі всі розподілені системи є масштабова­ними: щоб система відповідала новим вимогам, її можна нарощувати за допо­мо­гою добавляння нових обчислювальних ресурсів. Але на практиці нарощу­вання може обмежуватися мережею, яка об’єднує окремі комп’ютери системи. Якщо приєднати багато нових машин, то пропускна здатність мережі може ви­явитися недостатньою.
5. **Відмовостійкість**. Наявність декількох комп’ютерів і можливість ду­б­лювання інформації означає, що розподілені системи стійкі до певних апарат­них і програмних помилок. Більшість розподілених систем у випадку помилки, як правило, можуть підтримувати хоча б частково функціональність. Повний збій в системі відбувається тільки у випадку мережевих помилок.
6. **Прозорість**. Ця властивість означає, що користувачам надано повніс­тю прозорий доступ до ресурсів і в той же час приховано інформацію про роз­поділ ресурсів у системі. Однак, в багатьох випадках конкретні знання про ор­ганізацію системи допомагає користувачу краще використовувати ресурси.

До недоліків розподілених систем варто віднести:

1. **Складність**. Розподілені системи складніші від централізованих. На­багато складніше зрозуміти і оцінити властивості розподілених систем в ціло­му, а також тестувати ці системи. Наприклад, продуктивність системи залежить від швидкості роботи одного процесора, а від смуги пропускання мережі і шви­д­кодії різних процесорів. Переміщаючи ресурси з одної частини мережі в іншу, можемо радикально вплинути на продуктивність системи.
2. **Безпека**. Як правило доступ до системи можемо отримати з декількох різних машин, повідомлення в мережі можуть переглядатися або перехоплюва­тися. Тому, в розподіленій системі набагато складніше підтримувати безпеку.
3. **Керованість**. Система може складатися з різнотипних комп’ютерів, на яких можуть бути встановлені різні версії операційних систем. Помилка однієї машини не розповсюджується на інші машини з непередбачуваними наслід­ка­ми. Тому необхідно значно більше зусиль, щоб керувати і підтримувати сис­те­му в робочому стані.
4. **Непередбачуваність**. Як відомо всім користувачам Web-мережі, реак­ція розподілених систем на певні події непередбачувана і залежить від повного завантаження системи, її організації і мереженого навантаження. Оскільки ці всі параметри можуть постійно змінюватися, час , затрачений на виконання за­питу користувача, в той чи іншій момент може суттєво різнитися.

Всі вони пов’язані в першу чергу зі складною структурою, різноплановим устаткуванням і складною системою розподілу прав доступу. Необхідно враховувати всі з них, інакше розроблена інформаційна система не зможе функціонувати в рамках очікуваних параметрів.

В розподіленій системі різні системні компоненти можуть бути реалізо­вані на різних мовах програмування і виконуватись на різних типах процесорів. Моделі даних, подання інформації і протоколи взаємодії – все це необов’язково буде однотипним в розподіленій системі. Отже для розподілених систем необ­хідне таке програмне забезпечення, яке могло би керувати цими різнотипними частинами та гарантувати взаємодію і обмін даними між ними. *Проміжне про­грамне забезпечення* відноситься власне до такого класу ПЗ. Воно знаходиться якби посередині між різними частинами розподілених компонент системи.

# Види розподілених архітектур

Існує величезна кількість стандартів для створення правильної й надійної архітектури, а також для розробки й інтеграції програмних систем. Застосування цих стандартів істотно збільшить шанси на успішне створення системи і її подальше безвідмовне функціонування, однак раціональність їхнього застосування повинна визначатися до моменту початку робіт, оскільки складність системи при їхній інтеграції може істотно зрости.

На першому етапі процесу розробки архітектури система розбивається на декілька взаємодіючих підсистем. На самому абстрактному рівні архітектуру системи можемо зобразити графічно за допомогою блок-схеми, в якій окремі підсистеми подаються окремими блоками. Якщо підсистему також можемо роз­ділити на декілька частин, то на діаграмі ці частини відображаються в середині великого блоку.

Існують такі види розподілених архітектур:

* Модель репозиторія;
* Модель клієнт/сервер;
* Модель абстрактної машини;

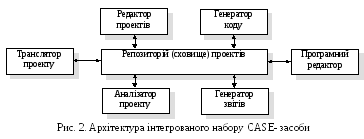
# Модель репозиторія

Для того щоб підсистеми, які входять в систему, працювали ефективно, між ними повинен проходити обмін інформацією. Обмін можемо організувати двома способами :

* Всі дані які використовуються спільно зберігаються в центральній ба­зі даних, доступній всім підсистемам. Модель системи, основана на спільно­му використання бази даних, часто називають моделлю репозиторія.
* Кожна підсистема має свою власну базу даних. Взаємо обмін даними між підсистемами відбувається за допомогою передачі повідомлень.

Більшість систем, що опрацьовують великі об’єми даних, організовані на основі бази даних, яка використовується спільно, або репозиторія. Прикла­дом може бути система управління інформацією, система автоматизованого проектування і CASE-засоби.

На рис. 2. подано приклад архітектури інтегрованого набору CASE-інструментів, яка базується на сумісному використанні репозиторію.



Широке використання дана модель отримала після того, як була застосована для підт­ри­мки розробки систем, написаних на мові Ада.

Спільне використання репозиторію має як переваги, так і недоліки:

* Очевидно, спільне використання великих об’ємів даних ефективне, оскільки непотрібно передавати дані із однієї підсистему в іншу.
* З іншої сторони, підсистеми повинні бути узгоджені з моделлю репо­зиторію даних. Це завжди потребує компромісу між вимогами, поставленими до кожної підсистеми. Компромісне рішення може знизити їх продуктивність. Якщо форми даних нових підсистем не підходять під узгоджену модель пода­них даних, інтегрувати такі підсистеми складно або неможливо.
* Підсистемі в якій створюються дані, не потрібно знати, як ці дані ви­користовуються в інших підсистемах.
* Оскільки у відповідності з узгодженою моделлю даних генеруються великі об’єми інформації, модернізація таких систем ускладнена. Переведення системи на нову модель даних буде дорогим і складним, а іноді неможливим.
* В системах з репозиторієм такі засоби, як резервне копіювання, забе­зпечення безпеки, керування доступом та відновлення даних, централізовані, оскільки входять в систему управління репозиторієм. Ці засоби виконують тіль­ки свої основні операції і не займаються іншими операціями.
* З іншої сторони, до різних підсистем ставляться різні вимоги, віднос­но безпеки, відновлення та резервування даних. В моделі репозиторію до всіх підсистем застосовується однакова політика.
* Модель спільного використання репозиторію прозора: якщо нові під­системи сумісні із узгодженою моделлю даних, їх можемо безпосередньо інтег­рувати в систему.
* Однак складно розмістити репозиторій на декількох машинах, оскіль­ки можуть виникнути проблеми, пов’язані з надлишковістю та порушенням ці­лісності даних.

В даній моделі репозиторій є пасивним елементом, а управління покла­дене на підсистеми, які використовують дані з репозиторію. Для систем штуч­ного інтелекту розроблено альтернативний підхід. Він базується на моделі «ро­бочої області», яка ініціює підсистеми тоді, коли конкретні дані стають доступ­ними. Такий підхід можливо використовувати до систем, в яких форма даних добре структурована.

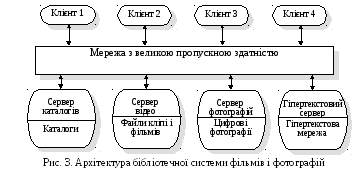
# Модель клієнт/сервер

Модель архітектури клієнт/сервер – це модель розподіленої системи, в якій показано розподіл даних і процесів між декількома процесорами.

Модель містить три основних компоненти:

1. Набір автономних серверів, які надають сервіси іншим підсистемам. Наприклад, сервер друку, який надає послуги друку, файлові сервери, які нада­ють сервіси управління файлами, і сервер-компілятор, який надає сервіси ком­пі­­ляції вихідних кодів програми.
2. Набір клієнтів, які викликають сервіси, що надаються серверами. В контексті системи клієнти є звичайними підсистемами. Дозволяється парале­ль­не виконання декількох екземплярів клієнтської програми.
3. Мережа, за допомогою якої клієнти отримують доступ до сервісів. В принципі немає ніякої заборони на те, щоб клієнти і сервери запускались на од­ній машині. На практиці модель клієнт/сервер в такій ситуації не використову­ється.

Клієнти повинні знати імена доступних серверів та сервісів, які вони на­дають. В той же час серверам непотрібно знати ні імена клієнтів, ні їх кількість. Клієнти отримують доступ до сервісів, які надає сервер, за допомогою віддале­ного виклику процедур.

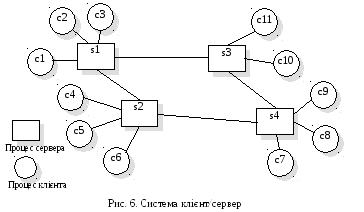


Приклад системи, організованої за типом моделі клієнт/сервер, зображе­но на рис. 3. Це багатокористувацька гіпертекстова система, призначена для підтримки бібліотеки відеофільмів і фотографій. В ній міститься декілька серверів, які розміщують різні типи медіа файлів і керують ними. Відео файли необхідно передавати швидко і синхронно, але з відносно малою роздільною здатністю. Вони можуть зберігатись у стисненому стані. Фотографії повинні передаватись з високою роздільною здатністю. Каталоги повинні забезпечувати роботу з множиною запитів і підтримувати зв'язок із використанням гіпертекс­тової системи. Тут клієнтська програма є просто інтегрованим інтерфейсом користувача.

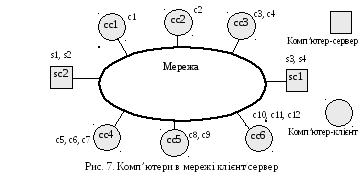
Підхід клієнт/сервер можна використовувати при реалізації систем, що базуються на репозиторії, який підтримується як сервер системи. Підсистеми, які мають доступ до репозиторію, є клієнтами. Але звичайно кожна підсистема керує власними даними. Під час роботи сервери і клієнти обмінюються даними, але при обміні великими об’ємами даних можуть виникнути проблеми, пов’я­зані з пропускною здатністю мережі. Правда, з розвитком все більш швидких мереж ця проблема втрачає своє значення.

Найбільш важлива перевага моделі клієнт/сервер полягає в тому, що вона є розподіленою архітектурою. ЇЇ ефективно використовувати в мережевих системах з множиною розподілених процесорів. В систему легко добавити новий сервер і інтегрувати її з останньою частиною системи або ж обновити сервери, не діючі на інші частини системи.

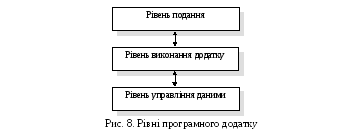
В архітектурі клієнт/сервер програмні додатки моделюються як набір се­рвісів, що надаються серверами, і множина клієнтів, які використовують дані сервіси. Клієнти повинні знати про доступні сервери, хоча і не можуть мати уя­в­лення про існування інших клієнтів. Як видно з рис. 6., на якій зображено мо­дель розподіленої архітектури клієнт/сервер, клієнти і сервери подають різні процеси.



В системі між процесорами і процесами не обов’язково повинно втри­муватися відношення «один до одного». На рис. 7. зображено фізичну архітек­туру системи, що складається із шести клієнтських машин і двох серверів. На них запускаються клієнтські і серверні процеси, зображені на рис. 6. В загаль­ному випадку, говорячи про клієнтів і сервери , маємо на увазі скоріше логічні процеси, ніж фізичні машини, на яких виконуються процеси.

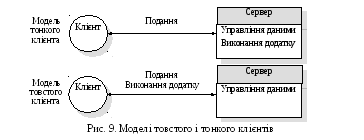


Архітектура клієнт/сервер повинна відображати логічну структуру прог­рамного додатку, який проектується .



На рис. 8 пропонується ще один погляд на програмні додатки, струк­ту­рований у вигляді трьох рівнів. Рівень подання забезпечує інформацію для ко­ристувачів і взаємодію з ними. Рівень виконання додатку реалізує логіку робо­ти додатку. Нарівні управління даними виконуються всі операції з даними. В централізованих системах між даними рівнями немає чіткого розподілу. Однак при проектуванні розподілених систем необхідно розділити ці рівні, щоб роз­класти кожен рівень на різні комп’ютери.

Самою простою архітектурою клієнт/сервер є дворівнева, в якій додатки складаються із сервера (або множини серверів) і групи клієнтів. Існує два види такої архітектури рис. 9.



**Модель тонкого клієнта**. В даній моделі всі робота додатку і керуван­ня даними виконуються на сервері. На клієнтській машині виконується тільки ПЗ рівня подання.

**Модель товстого клієнта**. В даній моделі сервер тільки керує даними. На клієнтській машині реалізована робота додатку і взаємодія з користувачем системи.

Тонкий клієнт дворівневої структури – самий простий спосіб переведе­ння існуючих централізованих систем в архітектуру клієнт/сервер. Користу­ва­цький інтерфейс в даних системах «переселяється» на персональний комп’ю­тер, а сам програмний додаток виконує функції сервера, тобто виконує всі про­цеси додатку і керує даними. Модель тонкого клієнта можемо також реалізу­ва­ти там , де клієнти подають собою звичайні мережеві пристрої, а не персональні комп’ютери або робочі станції. Мережеві пристрої запускають Internet-броузер і користувацький інтерфейс, реалізований в середині системи.

Головний недолік моделі тонкого клієнта – велика завантаженість сер­вера та мережі. Всі обчислення виконуються на сервері, а це може призвести до значного мереженого трафіку між клієнтом і сервером. В сучасних комп’ю­те­рах достатньо обчислювальної потужності, але вона практично не використо­ву­ється в моделі тонкого клієнта.

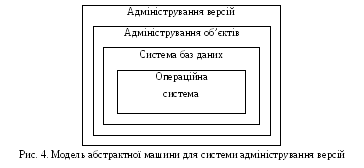
Модель товстого клієнта використовує обчислювальну потужність ло­кальних машин: і рівень виконання додатку, і рівень подання розміщуються на клієнтський сервер. Сервер в даному випадку, за сутністю, є сервером транзак­цій, який керує всіма трансакціями баз даних. Прикладом архітектури такого типу бути системи банкоматів, в яких банкомат є клієнтом, а сервер – центра­льним комп’ютером, який обслуговує базу даних розрахунку з клієнтами.

# Модель абстрактної машини

Модель абстрактної машини (багаторівнева модель) моделює взаємодію підсистем. Вона організовує систему у вигляді переліку рівнів, кожен з яких на­дає свої сервіси. Кожен рівень визначає абстрактну машину, машинна мова якої (сервіси, які надаються рівнем) використовуються для реалізації наступного рів­ня абстрактної машини. Наприклад, найбільш поширений спосіб реалізації мови програмування полягає у визначенні «мовної машини» і компіляції прог­рам, які написані на даній мові, в код цієї машини. На наступному кроці транс­ляції код абстрактної машини конвертується на реальний машинний код.

Відомим прикладом такого підходу може бути OSI (Open SyStem Iterco­nnection – взаємодія відкритих систем, на основі семирівневої моделі протоко­лів передачі даних у відкритих системах) мережевих протоколів. Іншим прик­ладом є трирівнева модель середовища програмування на мові Ада. На рис. 4. зображена подібна модель і показано, як за допомогою моделі абстрактної ма­шини можемо подати систему адміністрування версій.

Система адміністрування версій базується на управлінні версіями об’є­к­тів і надає засоби для повного управління конфігурацією системи. Для підтрим­ки засобів управління конфігурацією використовують систему адміністрування об’єктів, яка підтримує систему баз даних і сервіси управління об’єктами.



В свою чергу, в системі баз даних підтримуються різні сервіси, наприклад керу­вання транзакціями, зміщення назад, відновлення та керування доступом. Для керування базами даних використовують засоби операційної системи і її фай­лова система.

Багаторівневий підхід забезпечує покроковий розвиток системи – при розробці будь-якого рівня сервіси які надаються є доступними користувачам. Крім того, така архітектура легко змінюється і встановлюється на різні плат­фор­ми. Зміни інтерфейсу довільного рівня впливають тільки на суміжний рі­вень. Так в багаторівневий системах залежність від машинної платформи лока­лізовано на внутрішніх рівнях.

Недоліком багаторівневого підходу є доволі складна структура системи. Основні засоби, такі як керування файлами, необхідні всім абстрактним маши­нам, надаються внутрішніми рівнями. Тому сервісам, що запитуються користу­вачем, можливо, необхідний доступ до внутрішніх рівнів абстрактної машини. Така ситуація викликає руйнування моделі, оскільки зовнішній рівень залежить не тільки від попереднього рівня, але і від біль низьких рівнів.

Об’єкти можуть розміщуватися на різних комп’ютерах в мережі і взає­модіяти за допомогою проміжного ПЗ. За аналогією з системною шиною, яка дозволяє приєднувати різні пристрої і підтримувати взаємодію між апаратними засобами, проміжне ПЗ можемо розглядати як шину програмного забезпечення. Вона являє собою набір сервісів, які дозволяють об’єктам взаємодіяти один з одним, добавляти або видаляти їх із системи. Проміжне ПЗ називають брокером запитів до об’єктів. Його задача – забезпечити інтерфейс між об’єктами.

Нижче перераховано переваги архітектури розподілених об’єктів:

1. Розробники системи можуть не поспішати з прийняттям рішення від­носно того, де і як будуть надаватися сервіси. Об’єкти, що надають сервіси, в довільному вузлі мережі. Тобто, різниця між моделями товстого і тонкого кліє­нтів стають несуттєвими, оскільки немає необхідності планувати розміщен­ня об’єктів для виконання об’єкту.
2. Системна архітектура достатньо відкрита, що дозволяє при необхід­но­сті добавляти в систему ресурси. Окрім того, зауважимо, що стандарти про­гра­м­ної шини постійно вдосконалюються, що дозволяє об’єктам, які написані на різних мовах програмування, взаємодіяти та надавати сервіси один одному.
3. Гнучкість та масштабованість системи. Для того щоб справитися з си­стемними навантаженнями, можна створювати екземпляри системи з однако­ви­ми сервізами, які будуть надаватися різними об’єктами або різними екземп­ля­рами об’єктів. При збільшені навантаження, в систему можемо добавити нові об’єкти, не припиняючи при цьому роботу інших її об’єктів.
4. Існує можливість динамічної пере конфігурації системи засобами об’є­ктів, які мігрують в мережі за запитами. Об’єкти, які надають сервіси, мо­жуть мігрувати на той же процесор, що і об’єкти, які запитують сервіси, тим самим підвищуючи продуктивність системи.

В процесі проектування систем архітектуру розподілених об’єктів мо­же­мо використовувати двояко.

* У вигляді логічної моделі, яка дозволяє розробникам структурувати і планувати систему. В цьому випадку функціональність додатку описується тільки в термінах і комбінаціях сервісів. В подальшому розробляються способи надання сервісів за допомогою декількох розподілених об’єктів. На цьому рівні, як правило, проектуються великомодульні об’єкти, які надають сервіси, що ві­дображають специфіку конкретної області додатку. Наприклад, в програму об­ліку роздрібної торгівлі можемо включити об’єкти, які б вели облік стану запа­сів, відслідковували б взаємодію з клієнтами, проводили класифікацію товару та інше.
* Як гнучкий підхід до реалізації системи клієнт/сервер. В даному ви­па­дку логічна модель системи – це модель клієнт/сервер, в якій клієнти і сервери реалізовані як розподілені об’єкти, які взаємодіють засобами програмної шини. При такому підході легко замінити систему, наприклад дворівневу на багато­рі­вневу. В даному випадку ні сервер, ні клієнт не можуть бути реалізовані в одно­му об’єкті, однак можуть складатися з множини невеликих об’єктів, кожен з яких надає певний сервіс.

Головним недоліком архітектур розподілених об’єктів є те, що їх склад­ніше проектувати, ніж системи клієнт/сервер. Тобто, системи клієнт/сервер на­дають більш природний підхід до створення розподілених систем. В ньому ві­дображаються взаємовідношення між людьми, при яких одні люди користую­ться послугами інших людей, які спеціалізуються на наданні конкретних пос­луг. Набагато складніше розробити систему у відповідності з архітектурою роз­поділених об’єктів, оскільки індустрія створення ПЗ поки що не має достатньо­го досвіду в проектуванні та розробці велико модульних об’єктів.

# Приклади реалізацій розподілених систем

# 1.2.1 Огляд застосунку «Гора» створеного для навчання працівників фірми.

Одним з прикладів застосування принципів гейміфікації в сфері бізнесу є створення і використання дуже вдалого ігрового проекту «Гора». Цей застосунок призначений для навчання працівників відділу продажу певним спеціальним знанням. Застосунок використовується всередині комерційної камнанії для підготовки її персоналу.

Загалом застосунок має архітектуру розподіленої системи за схемою клієнт — сервер. Застосунок створений на базі платформи Knowledge Guru ©. Ця платформа використовується для створення додатків, які використовують навчально-орієнтовані ігри, адаптивне навчання і мікро-навчання для підвищення збереження знань і підвищення продуктивності працівників [http://www.bottomlineperformance.com/knowledge-guru/ 14/05 ]. Організації використовують цю платформу, щоб допомогти своїм співробітникам зменшуючи зусилля підвищувати свій кваліфікаційний рівень. Розділ «вивчення науки», що вбудований в платформу, допомагає учням підвищувати свою впевненість і компетентність протягом декількох хвилин в день, працюючи з застосунком на смартфоні, планшеті або на персональному комп`ютері. Тренери використовують веб-додаток для швидкого створення і редагування ігор. Також платформа надає двигун аналітики, що допомагає знайти і усунути прогалини продуктивності.

Користувачі мають програмний інтерфейс у вигляді клієнтської програми. При запуску програми відбувається авторизація клієнта, для входу в обліковий запис гри з використанням інтернету. Велика частина программи працює в «хмарі». Коли гравці входять в гру в перший раз, вони отримують розповідь, котра пояснює, принцип гри. За сценарієм гравець повинен піднятися на гору по кожній з трьох стежок. Стежки представляють собою окремі теми. Гра складається з трьох шляхів на вершину гори. Щоб доставити сувій мудрості до Гуру необхідно пройти всі три шляхи (Рис. 1.1 ).

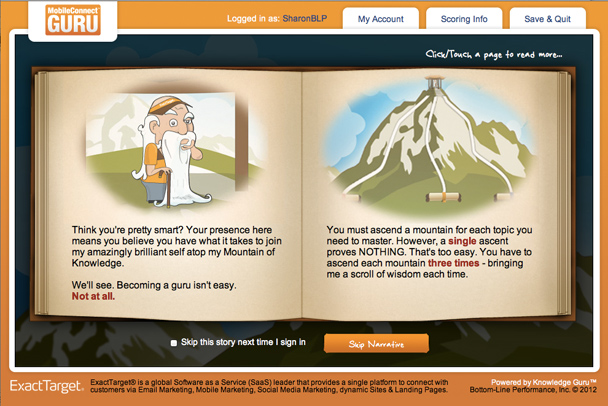


Рис. 1.1 Гуру та три шляхи на гору

Кожна стежка має мету навчання, і певну тему. Гравці бачать свій рахунок, що відображає їх відповідають на кожне питання гри (Рис. 1.2 ). Якщо гравці відповідають правильно, їх оцінка зростає. Якщо вони відповідають неправильно, навпаки спадає. Ігровий движок використовується для створення та розміщення «Mobile Connect» гри. Та має функції, що дозволяють відстежувати дії певного характеру. Це дозволяє керівнику, навчання робітників чи іншим компетентним особам побачити, як гравці виконують певні завдання. Доступні спеціальні можливості, що ґрунтуються на основі цих результатів. Можливо навіть дізнатися детальніше як конкретний гравець виконує завдання і визначити, чого він досяг, що було складно для нього, і скільки часу він провів, граючи в ігри створені за допомогою платформи.



Рис 1.2 Насіння позначає кількість балів.

Безпосередня перевага гри в порівнянні з традиційними інструментами навчання — його привабливість. Персонал зацікавлений грати, тоді як працівники не завжди хочуть бути присутніми на тренуванні.

Повторення є ключем до успіху в цій платформі, а також з використанням ігрових елементів і використанням історії питань. Процес навчання повторюється кілька разів, і це підвищує якість навчання.

Поєднання різних ігрових елементів, а також документовані результати бізнесу показує, що можна успішно використовувати гейміфікацію для мотивації і навчання робітників, з метою навчання та ознайомлення їх з новим продуктом і функціями.

[https://www.learningsolutionsmag.com/articles/1240/the-gamification-of-sales-force-training] 14/05 Карла Каппа The Gamification of Sales Force Training

# 1.2.2 Огляд за стосунку освітнього змісту «Орегонський шлях» .

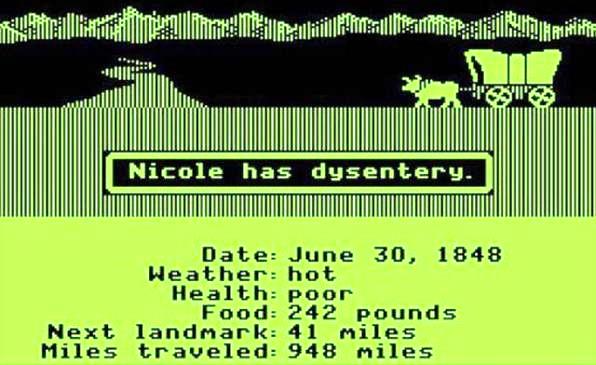
Однією з найстаріших ігор що використовувалась в освітніх закладах є «The Oregon Trail» (Орегонська стежка). Ця гра широко використовувалася в американських школах вже з 70-х років 20-го століття. Цікаво зазначити, що гра користувалася величезною популярністю в американських дітей. Внаслідок чого було створено декілька версій, що продовжують ідею гри. Автором програмного коду є Білл Хейнеманн. Також до складу розробників входив шкільний вчитель на ім’я Дон Равич. В подальшому він працював на державний освітній комп'ютерний консорціум Міннесоти. Саме цей консорціум і займався поширенням гри в навчальні заклади. В «Творчому журналі» було опубліковано програмний код гри.

Рис 1.3 Зображення ігрового процесу «огеронського шляху»

Щодо ігрового процесу гравцю було потрібно керувати групою американських поселенців певної професії, що подорожують по Орегонській стежці.

Метою гри було дати дітям можливість навчитися раціонально підходити до вирішення різних завдань. Наприклад гравецю потрібно було обрати швидкість тяглової худоби. Чим більшою вона була, тим швидше гравець міг дістатися до Орегона, а також отримати додаткові бали. Також, це дозволяло закінчити мандрівку раніше настання зими, з наближенням якої гра ускладнювалась. Але висока швидкість приводила до частої поломки фургона або буйвола. Також учень повинен був раціонально підійти до розподілу продовольства між членами загону. Голодування могло призвести до захворювання, а також до летальних випадків.

На шляху могли трапитися дружні або ворожі індіанці. Це призводило до поповнення або втрати припасів. Серед інших раптових подій можна відзначити напад розбійників, епідемії (найгірше була дизентерія), урожай диких фруктів, зміну погоди, псування харчів. Більш того, на березі кожної річки гравець міг заплатити за переправу на поромі, переправитися вбрід, що загрожувало поломкою фургона, або ж дочекатися відливу — це забирало багато часу.

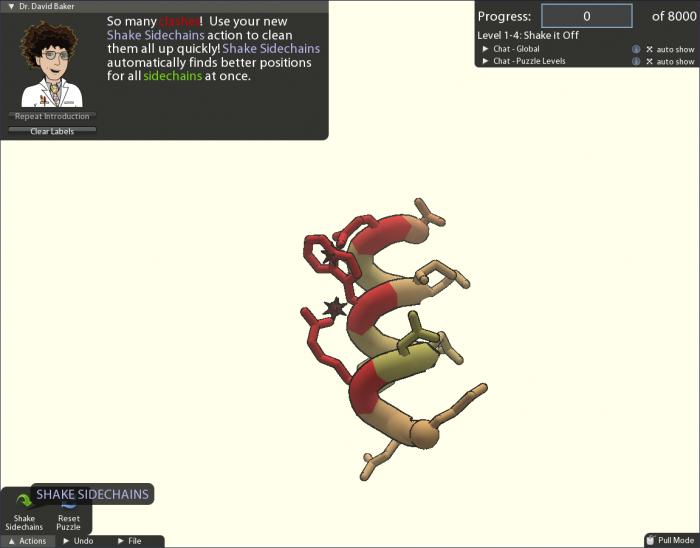
Час від часу поїзд зупинявся в місцях, де можна було придбати необхідні товари за все більш зростаючими цінами і дізнатися нові історичні факти. Місцями зупинок були скелі Чімні-Рок та в форте Холл. Крім того, в будь-якому місці на шляху між поселеннями і іншими орієнтирами гравець міг зупинитися, щоб відпочити або поторгувати. Інтенсивне полювання приводила до зменшення поголів'я тварин. Відстріл бізонів давав набагато більше м'яса, ніж міг вмістити Конестога гравця. На річці Колумбія гравець опинявся перед незвичайним вибором. Він не міг перейти її вбрід, замість цього йому потрібно було подолати її, пливучи вниз за течією в полуаркадном стилі, що в цілому було нехарактерно для гри. У цій міні-грі користувач повинен був правити фургоном на плоту, ухиляючись від каменів.

Дії гри починаються в 1848 році в місті Індепенденс, штат Міссурі, де починається Орегонська стежка. Гравець вибирає професію, дає імена членам загону і купляє припаси. Гравець може бути банкіром з Бостона, теслею з Огайо або фермером з Іллінойсу. Гра персонажами, які мали більше грошей, була легшою. Потім гравецю необхідно було купляти необхідне спорядження, худобу, продукти, боєприпаси. Рекомендувалося також купити запасні частини для фургона, наприклад, вісь або хомут.

В кінці подорожі очки нараховуються за формулою, в залежності від обраної професії, числа і стану здоров'я членів загону що вижили, того які залишилися пожитків і скільки готівки.

[https://ru.wikipedia.org/wiki/The\_Oregon\_Trail ] 10/05

# 1.2.4 Огляд освітньо-наукової гри «Foldit»

На фоні відносно великої кількості навчальних проектів яскраво виділяється гра, що носить дослідницько-науковий характер. Застосунок розроблений як частина ефективного дослідницького проекту Вашингтонського університету. Foldit — онлайн-головоломка про холдинг, інакше кажучи, згортання білків. Мета гри полягає в тому, щоб найкращим чином згорнути структуру протеїнів. Гравці отримують послідовність амінокислот або частково згорнутий білок і повинні знайти їх конформацію із найменшою вільною енергією.  Рис. 1.4 Червоний колір позначає області зі скупченнями вільної енергії

Найкращі рішення користувачів використовуються вченими. За допомогою данних отриманих від гравців вчені працюють над рішеннями реальних наукових проблем, таких як пошук вакцин та біологічних інновацій. Спершу успіхи Foldit обмежувалися тим, що були відкриті структури конкретних білків, пізніше застосунок дав змогу вирішувати і більш складні завдання, такі як розробка алгоритму для передбачення конформації білкових молекул.

Ціль головоломки полягає у пошуку тримірної структури певного білка із найнижчим рівнем вільної енергії. Кожне нове завдання публікується на сайті на певний визначений термін, протягом якого гравці змагаються між собою, намагаючись отримати найменший показник вільної енегрії. Одночасно також існує набір головоломок, що доступні постійно. Цей набір розроблений для знайомства нових користувачів із особливостями гри Foldit. В ході ігрового процесу гравці інтерактивно здійснюють перетворення над молекулою, змінюючи форму основного каркасу і положення бічних груп. Вони також можуть обертати α-спіралі навколо соєї осі, редагувати сполучення ланцюгів у β-структурах, накладати слабкі обмеження у певних ділянках («rubber bands») або фіксувати їх[2 ??]. Користувачі мають доступ до панелі інструментів для виконання автоматизованих завдань, таких як команда «wiggle». Ця команда дозволяє локально мінімізувати енергію[3 ??]. Багато технічних термінів у грі замінені на зрозуміліші для користувачів без підготовки, щоб зробити гру доступною для всіх гравців. Кращі результати аналізуються вченими для подальшого використання. Цікаво, що більшість із найкращих гравців Foldit не мають біохімічної освіти [ Cooper S, Khatib F, Treuille A, Barbero J, Lee J, Beenen M, Leaver-Fay A, Baker D, Popović Z, Players F (2010). «Predicting protein structures with a multiplayer online game». Nature 466: 756—60. DOI:10.1038/nature09304. PMID 20686574].

Користувач дізнається наскільки вдало він згортає білок, за допомогою балів, які нараховуються, за утворення нових водневих зв'язків, переміщення гідрофобних залишків всередину молекули та інше. У грі розроблені підказки, наприклад такі, що підсвічують ділянки, у яких певні групи стикаються і потребують розведення або про те, що відкритими є гідрофобні ділянки, котрі бажано «сховати» в середину молекули і порожнини, які слід заповнити[2 ?? ]. Користувачі можуть спілкуватись, обговорювати питання та ділитись один з одним варіантами рішень, а також брати участь у покращенні вікі присвяченої Foldit[3 ?? ].

Підсумовуючи цю інформацію можна виділити основні засоби гейміфікації. Кожен з цих засобів несе в собі «ігрове» забарвлення та має певний психологічний вплив, котрий може бути позитивно використаний при застосуванні з метою ігрофікації. Засоби ігрофікації є відмінними один від одного але деколи мають спільні властивості. Застосування окремо кожного з цих засобів дає змогу отримати певний ефект впливу на користувача. Цей ефект є корисним з точки зору гейміфікації адже дозволяє зацікавити користувача, мотивувати його або стимулювати в ньому суперницький настрій з метою залучення до навчального процесу за допомогою гейміфікації. Комбінуючи або використовуючи одночасно засоби гейміфікації, залежно від конкретних умов реалізації кінцевого продукту, можливо зацікавити користувача до проводження часу за використанням освітнього застосунку.

Проведено короткий огляд з еволюційної точки зору декількох застосунків з освітньої та наукової галузі. Представлені застосунки реалізовані унікально і сильно відрізняються один від одного. Відмінності застосунків зумовлені тим, що застосунки були розроблені для різної користувацької аудиторії та та програмному рівні мають різні цільові середовища для виконання. Попри це застосунки обєднує спільна проблема, котру вони вирішують. Кожен з них створений з освітньою метою і метою виховання здібностей до раціонального міркування, вирішення типових завдань відповідно до сфери застосування.

## Висновки

1. Приведено визначення гейміфікації.
2. Окреслено особливості гейміфікації, її підходи, властивості .
3. Здійснено огляд декількох за стосунків освітнього та дослідницького характеру.

# РОЗДІЛ 2

# Дослідити можливість їх застосування в навчальному процесі. Запропонувати модель навчального процесу із застосуванням принципів гейміфікації.

# 2.1 Нова освітня модель в контексті навчального процесу

Сучасний освітній процес в навчальних закладах включає наявність декількох видів занять таких як лекції, практичні та лабораторні роботи. На самостійну роботу роботу студента відводиться велика кількість часу. У тому числі і час для підготовки до занять. Зазвичай підготовка до лекцій являє собою повторення вже пройденого матеріалу або ознайомлення з новими знаннями і літературою такою як підручники, методичні посібники, конспекти лекцій. Підготовка до практичних робіт являє собою рішення прикладних практичних задач та виконання інших завдань. Підготовка до лабораторних робіт зазвичай зводиться до повторення теоретичного матеріалу. Під час лабораторних занять студентам пропонується отримати навички роботи з апаратурою і різними пристроями, установками або програмами-симуляторами на ПК. Цей досвід розширює світогляд студентів і дає можливість одержати навички роботи та побачити варіанти реального застосування знань, одержуваних на лекційних і практичних заняттях.

????. В данній роботі пропонується застосування в навчальному процесі навчальної моделі з застосуванням гейміфікації. Існує ряд переваг моделі з використанням гейміфікації перед подібною моделлю без використання гейміфікації (Рис 2.1.1 )

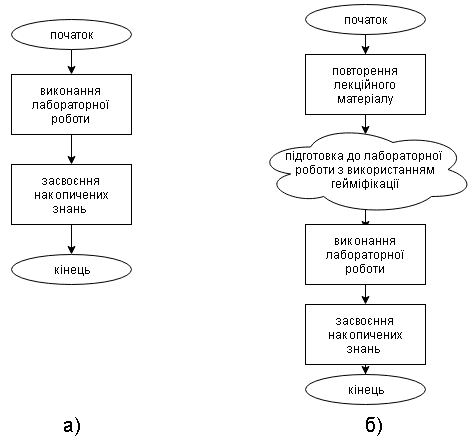


Рис. 2.1.1 Моделі підготовки до лабораторних занять а) з використанням гейміфікації б) без

. Також наявні переваги використання в навчальній моделі беспосередньо застосунка, ці переваги будуть висвітлені дещо пізніше. Стосовно переваг освітньої моделі на рисунку 2.1.2 представлений блок що включає ознайомлення з метою та змістом лабораторної роботи спираючись на методичний матеріал. Після нього слідує блок виконання завдань в програмному застосунку, що ознайомлять студента з виглядом лабораторної установки та її складовими частинами. Це дає змогу студентові напередодні виконання лабораторної роботи отримати короткі відомості про роботу та на момент безпосереднього виконання почувати себе впевнено в більш «знайомих» умовах при відпрацюванні роботи «на техніці». При відпрацюванні лабораторних робіт курсу технічна електродинаміка та поширення радіохвиль важливим чинником є знання студентом лабораторної установки та призначення окремих її «вузлів», оскільки для отримання вимірів при проведенні лабораторної роботи студентові необхідно заміняти

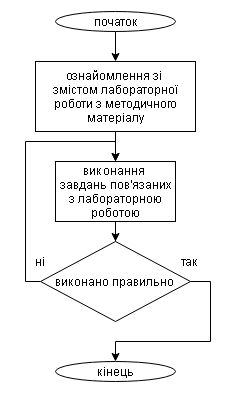


Рис. 2.1.2 Процес підготовки до лабораторної роботи.

або від’єднувати окремі частини установки. Ці маніпуляції дають змогу отримати різні результати вимірювань. В подальшому, користуючись цими результатами вимірювань студенти можуть порівняти їх, підтвердити певні відомі закономірності, або оцінити правильність отриманих вимірів.

Також необхідно звернути увагу, що при використанні данної освітньої моделі студент декілька разів звертається і до методичного матеріалу, як при підготовці до роботи з додатком так і під час виконання лабораторної роботи. Так само двічі студент проходить виконання самої роботи при роботі з додатком та в ході лабораторної роботи. Тобто збільшується кількість повторень котра позитивно впливає на якість запам’ятування і засвоєння знань.

Автор відомої кривої забування професор Еббінгауз завдяки своїм експериментам отримав наступний досвід. Він давав для запам'ятовування своїм студентам довгий список безглуздих звукосполучень типу «дейюкс», «кволі» і подібні до них. В процесі експерименту він встановив, що студенти запам'ятовували ту ж кількість звукосполучень за 38 повторень, вивчаючи їх з інтервалами протягом трьох днів, що і за 68 безперервних повторень протягом одного дня.[ <http://www.brainhack.me/2015/08/3-memory-laws/> 240517] Еббінгауз зрозумів, що в певний момент люди досягають плато - точки, в якій додаткові зусилля майже не призводять до додаткових результатів. Він почав експериментувати з часовими інтервалами між изучениями інформації і згодом довів, що ефективність може бути підвищена на 500 відсотків за допомогою впровадження найпростіших технологій. Його відкриття плато запам'ятовування, а також дослідження про те, як цього ефекту уникнути, привели до розвитку методів навчання, які допомогли прискорити процес запам'ятовування і лягли в основу однієї з найефективніших технік запам'ятовування в світі.

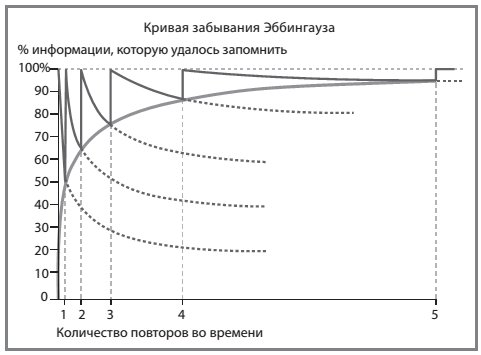


Рис. 2.1.3 Крива забування Ебінгауса

Згідно до найефективнішого методу навчання Ебінгаус визначає наступний алгоритм повторень.

* Перше повторення — відразу після закінчення читання.
* Друге повторення — через 20 хвилин після першого повторення.
* Третє повторення — через 24 години після другого.

Можна помітити, що запропонована модель навчання за графіком повторень дуже наближується до встановлених Ебінгаусом висновків, що свідчить про її високу ефективність.

# Відмінності між роботою(освіта) та грою

[presentation ENTERPRISE GAMIFICATION Engaging people by letting them have fun Mario Herger]

У своїй доповіді: «Зацікавлення людей, дозволяючи їм отримувати задоволення» - Маріо Хергер, доповідаючи про гейміфікацію, приводить порівняльну характеристику гри та роботи(Табл. 2.1). У таблиці в стовпці «робота» представлено сукупні характеристики, що є справедливими для більшості професій та багатьох видів періодичної монотонної діяльності. Оскільки освіта або точніше навчання — важка розумова діяльність, праця, в такому випадку справедливо говорити про тотожність роботи та навчання (провести аналогію з роботою). В буквальному сенсі слово «гейміфікація» можна зрозуміти як процес перетворення чогось в гру. В данній роботі мова йде про перетворення певного навчального матеріалу в гру. З таблиці можна побачити різку перевагу гри перед навчанням у чистому вигляді. В дещо меншій мірі гейміфікація дозволяє досягти цієї різниці, але відмінність все одно відчутна.

Табл. 2.1 Порівняльна характеристика між грою та роботою.

|  |  |  |
| --- | --- | --- |
|  | Гра | Робота |
| Завдання | Повторюється, веселе | Повторюється, одноманітне |
| Зворотній звязок | Постійно | Рідко |
| Мета | Зрозуміла | суперечливий, розпливчастий |
| Шлях до досконалості | Зрозумілий | Не зрозумілий |
| Правила | Зрозумілі | Не зрозумілі |
| Інформація | Потрібна кількість в потрібний час | Забагато і недостатньо |
| Помилки | Очікується, підбадьорений, ефектний, хвалитися про це | заборонено, карали, не говорити про це |
| Статус користувачів | прозорий, своєчасний | Прихований |
| Співробітництво | Присутнє | Присутнє |
| Швидкість, ризик | Вісокі | Низькі |
| Автономність | Висока | Середня-низька |
| Інформативність | Висока | Іноді |
| Перешкоди | Залежно від мети | Випадкові |

В те дни, когда я даже думать не хотел о рисовании, я исполь- зовал трюк, который узнал от Рика Рубина, который состоит в том, чтобы согласиться на минимальный объем работы. В моем случае я говорил себе, что могу прекратить рисовать, как только нарисую одну линию

[Mihaly Csikszentmihalyi: *Flow: The Psychology of Optimal Experience*, 1991 *Beyond Boredom and Anxiety: Experiencing Flow in Work and Play*, 1975]

Зворотній зв’язок, між користувачем і застосуванням є важливим засобом гейміфікації, він дозволяє найефективнішим шляхом тримати користувача в стані найбільш підходящому для ефективного освітнього процесу. Застосунок з використанням гейміфікації певним чином динамічно пристосовується до поведінки користувача. Наприклад за стосунок може змінювати складність завдань. Понижувати або збільшувати рівень складності завдань відповідно до поведінки користувача. Такий підхід може бути реалізований наприклад наступним чином. Якщо користувач починає робити помилки частіше, застосування динамічно знижує рівень складності, щоб користувач отримував адекватні за складністю завдання відповідно до свого рівня підготовки. Такий підхід є дуже ефективним, адже дозволяє підтримувати інтерес користувача не перевантажуючи його надто складними завданнями. Таким чином будь-який користувач знаходиться у стані сприятливому для навчання. Подібний процес описав психолог Михай Чиксентмихайи. Свою теорію він назвав теорією оптимального переживання. Теорія оптимального переживання ґрунтується на понятті потоку-стану. Потік-стану означає повне поглинання діяльністю настільки, що інші фактори вже не чинять істотного впливу. Описуючи величину отримуваного задоволення від процесу діяльності вчений зазначає, що задоволення настільки велике, що люди будуть готові навіть платити гроші щоб займатися цією діяльністю.

Також Михай вводить поняття потокової діяльності, зміст якої полягає в набутті радості. Прикладами таких занять є ігри, мистецтво, спорт, а також різні активності. Потокової діяльності змінюють особистість, роблячи її більш складною, завдяки її розвитку. Розвиток тісно пов’язаний зі змістом потокової діяльності (рис. 1).

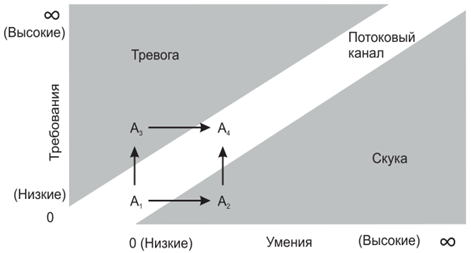
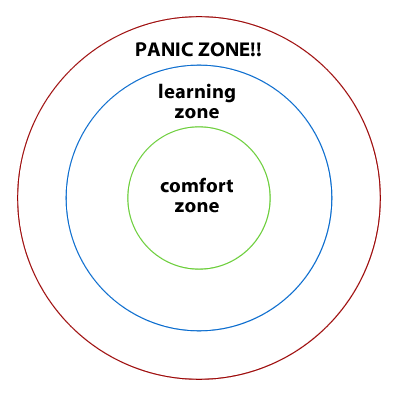


Рис. 2.4 Потоковий канал

Припустимо, що наведений нижче графік описує якусь діяльність, наприклад, гру в теніс. На його осях відкладені рівні розвитку умінь і складності завдань. буквою А позначений Алекс - хлопчик, який вчиться грати в теніс. На діаграмі він зображений в чотирьох тимчасових точках. На момент початку тренувань у нього практично відсутні навички гри в теніс ( точка А1 ). Від нього вимагається просто влучити по м'ячу. Це не складно, проте Алекс тренується з величезним задоволенням. Адже рівень складності завдання як раз відповідає його елементарних навичок. Тому на цьому рівні він, ймовірно, буде відчувати відчуття потоку. ( Рис. 2.4 ). Згодом, якщо він продовжить тренуватися, його спортивна форма стане краще. йому буде нудно просто відбивати м'яч через сітку ( точка А2 ). Можливо, він знайде собі більш досвідченого партнера і усвідомить, що існують більш складні завдання, ніж просто бити по м'ячу. У цій точці A3 він відчує занепокоєння через те, що погано справляється з ситуацією. Оскільки ні нудьга, ні тривога не відносяться до приємних станів, Алекс знову хоче випробувати стан потоку. З малюнка видно, що, якщо він нудьгує ( точка А2 ) і прагне знову потрапити в потік, у нього є тільки один вихід - підвищити складність виконуваних завдань. Поставивши перед собою нову, більш складну мету, що відповідає рівню його навичок, — наприклад, перемогти сильнішого супротивника, — Алекс знову опиниться в потоці ( точка А4 ). Якщо він відчуває занепокоєння ( точка A3 ), для повернення до стану потоку йому необхідно далі вдосконалювати свої навички. Стан в А4 можна вважати більш «просунутим» у порівнянні з А1, тому що він передбачає більш складні завдання і вимагає від гравця більшої майстерності. Однак ситуація А4, незважаючи на складність і принесену нею радість, також не є стабільною. якщо Алекс продовжить грати, він зіткнеться або з нудьгою через які втратили новизну завдань, або з тривогою і фрустрацією через усвідомлення своїх відносно слабких можливостей. прагнення отримувати задоволення знову штовхне його в потік, але вже на рівні складності, що перевищує А4. Така динаміка пояснює, чому потік сприяє особистісному розвитку. Людина не може довго отримувати насолоду, роблячи одне і те ж. Щоб уникнути нудьги, він повинен ставити собі нові цілі і, досягаючи їх, удосконалювати свої навички.



Comfort Zone:

Зона комфорту дійсно відмінне місце, щоб бути в. Це зона, де не приймають ніяких проблем. Це місце, де багато хто з нас живуть просто тому, що це зручно, безпечно, не займають думка або розум, і ніяких турбот не беруть участь. Отже, не інтелектуальні зусилля не виплачений, так як це зона нормального життя, яка не включає в себе ніяких ризиків взагалі. Звичайно ж, це не обов'язково означає, що ми щасливі в той час як ми знаходимося в Comfort Zone, тому що ми могли б бути в болю поки ми не вживати ніяких дій, щоб вийти з цієї ситуації, а ми віддатися самовдоволення, і наше життя перетворити в рутину, людина, яку ми сьогодні та сама людина, ми були вчора. У зв'язку з цим, життя не захоплюючий, і ми ні рости, ні розвиватися; навіть якби ми зробили, це нереально зростання або сприяє відсталості - так як недолік розвитку самої відсталість.

Паніка зона:

Зона паніки характеризуються сильною тривогою і напругою. Людина в цій зоні постійно перебуває в стані як безмисленной активності і стресу. Навіть коли нічого не відбувається, він знаходиться в постійній тривозі і стресі, тому що він знайде щось панікувати і бути стурбовані. Він може навіть створювати речі, щоб турбуватися про.

В основному, всі питання, великі і важливі, і вимагають уваги, яке супроводжується певною кількістю страху або тривоги, що перешкоджає нашій здатності впоратися з ситуацією належним чином. Життя сповнене проблем і труднощів. Ми часто бажаємо на краще, але ми насправді не робити жодних кроків, щоб зробити його краще. Ми знаходимося в стані постійної реакції на речі навколо нас. Як результат - такий же, як вона знаходиться в зоні комфорту - ми не будемо розвивати.

Навчання зони:

Зона навчання характеризується різним способом обробки все. Для кожної ситуації є чому повчитися, чи було це добре чи погано, правильно чи неправильно, легко чи важко, він завжди буде виглядати як досвід. Для того, щоб бути в навчальній зоні, щоб мати можливість сказати: «незалежно від того, що відбувається, я не буду користуватися і вчитися». У цій зоні, проблеми розглядаються як спосіб рости, помилки розглядаються як спосіб отримати досвід, і невдача розглядається як можливість створення персонажа. Людина, ми сьогодні були мудрішими, ніж людина, ми були вчора. Життя повне можливостей і сюрпризів, і наше хвилювання на-віч день слід збільшити день за днем.

Перейти до Вивченню Зоні:

Люди часто виявляються або в комфортній зоні або панічної зоні великою частиною свого життя. Там немає ніякого зростання, і завжди є тривога і стрес. Життя перетворюється в фатальну рутину, без зору або цілі, щоб дивитися вперед. Причина, чому людина повинна двічі подумати, як перейти в стан або зони навчання. Для цього потрібно лише воля, рішучість і дії.

Це зовсім не означає, що ми не знаходимо себе, перебуваючи в будь-якому панічної зоні або зоні комфорту; це просто означає, що, коли ми опиняємося в них, ми повинні бути в змозі прийняти рішення вийти з них і крок в навчальній зоні, і що ми будемо реагувати і реагувати на ситуацію по-іншому. Перебуваючи в зоні навчання змінить спосіб, яким ми живемо на краще. Саме тому ми повинні вирішити, щоб бути в навчальній зоні тепер.

# РОЗДІЛ 3

# Розглянути особливості застосування запропонованої навчальної моделі для курсу технічної електродинаміки.

# РОЗДІЛ 4 Розробити макети лабораторних робіт з курсу технічної електродинаміки, які базуються на принципах гейміфікації.

## Созданная компьютерная программа позволяет ознакомиться с устройством и порядком сборки лабораторной установки. В этом приложении создано макеты лабораторных установок предмета техническая электродинамика и распространение радиоволн. Игровая логика позволяет получить интерактивное взаимодействие с установками. При проведении лабораторных работ важной частью является порядок сборки установки. При снятии показаний с измерительных устройств установки в ходе проведения лабораторной работы производится замена некоторых её составляющих. Поочередная установка различных сменных деталей позволяет получить разные выходные данные. С помощью отличающихся данных студенты могут пронаблюдать физические зависимости, проверить практически известные теоретические знания. В приложении предоставляется возможность сборки установки.

## Для создания приложения была выбрана игровая платформа разработки Unity 3D так как она предоставляет возможности для создания интерактивной программы с использованием трехмерных моделей. Их использование позволяет сделать процесс работы в приложении более наглядным. Трехмерные модели составных частей установки были смоделированы в пакете создания трехмерной графики Blender 3D. Оба средства разработки являются бесплатными программными продуктами.

## Висновки

1. Сформовано необхідні завдання та вимоги до розробки інформаційної системи.
2. Запропоновано та впроваджено програмну реалізацію інформаційної системи оцінки психофізіологічного стану людини.
3. Розроблено додаткові модулі та підсистеми для поглибленого аналізу та представлення інформації в наглядному вигляді.
4. Впроваджено для використання в НАНЦ України.

# ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Проведено детальний аналіз методів оцінки стану людини за психологічними показниками.
2. Визначена необхідність розробки інтелектуальної технології оцінки психофізіологічного стану людини, який може використовувати існуючі методи за відсутності експерта.
3. Поставлене завдання розробки інтелектуальної технології. Визначено критерії та правила, за якими отримана оцінка вважається достовірною.
4. Запропоновано інтелектуальну технології оцінки психофізіологічного стану людини.
5. В рамках системи оцінки психофізіологічного стану людини, запропоновано алгоритм визначення кольорової преференції.
6. Розроблено апаратно-програмну інформаційну технологію оцінки психофізіологічного стану людини з можливостями налаштування та перегляду параметрів, введення інформації, її зберігання та використання, представлення інформації в декількох можливих варіантах.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мадяр А. Й. Спосіб корекції психофізіологічного стану людини. / А. Й. Мадяр, О. Е. Ковалевська, В. В. Арбатов, В. Н. Бержанський, М. В. Луцюк, Є. В. Моісеєнко, В.Б. Павленко, С.В. Чорний // Патент на корисну модель № 28058 від 26 листопада 2007 року.
2. Готовский Ю. В. Цветовая светотерапия. / Ю. В. Готовский, А. П. Вышеславцев, Л. Б. Косарева, Ю. Ф. Перов, М. М. Шрайбман – М. : ИМЕДИС, 2001. – 432 с
3. Петухова Н. В. Оценка психофизиологического состояния до и после формирования и активаций артифициальных стабильных функциональных связей мозга человека : автореф. дис. к.б.н. / Н. В. Петухова. – Санкт - Петербург: ИМЧ РАН, 2000 – 118 с.
4. Мадяр А. Й. Инновационные методы исследований в психофизиологии цветового восприятия : методическое пособие / А. Й. Мадяр, Е. В. Моисеенко, Е. Э. Ковалевская – М. : ИМЕДИС, 2015. – 120 с.
5. Guth L. S.: The effect of wavelength of visual perception latency. Vis. Res. 4, 567 , 1964.
6. Свідоцтво про реєстрацію авторського права на твір №7043 від 30.01.2003. Наукова розробка «Біоколор – спосіб корекції психофізіологічного стану людини».
7. Яньшин П. В. Клиническая психодиагностика личности. Учебно-методическое пособие / П.В. Янышин — СПб.: Речь, 2007. — 320 с.
8. Нейронные сети [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.statsoft.ru/home/textbook/modules/stneunet.html. Дата доступу: 10 березня 2016.
9. [Розенблатт Ф.](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%BD%D0%BA_%D0%A0%D0%BE%D0%B7%D0%B5%D0%BD%D0%B1%D0%BB%D0%B0%D1%82) [Принципы нейродинамики: Перцептроны и теория механизмов мозга](http://vlabdownload.googlecode.com/files/ros_principles2.djvu) / Ф. Розенблатт — М.: Мир, 1965. — 480 с.
10. Уоссермен Ф. [Нейрокомпьютерная техника: Теория и практика](http://evrika.tsi.lv/index.php?name=texts&file=show&f=410) / Ф. Уоссермен — Мир, 1992. — 240 с. — [ISBN 5-03-002115-9](https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0:%D0%94%D0%B6%D0%B5%D1%80%D0%B5%D0%BB%D0%B0_%D0%BA%D0%BD%D0%B8%D0%B3/5030021159).
11. Нейронные сети. [Електронний ресурс]. Електрон. текстові дані. – Режим доступу: http://www.neuroproject.ru/neuro.php. Дата доступу: 01 квітня 2016.
12. Джарратано Дж. Экспертные системы: принципы разработки и программирование / Дж. Джарратано — М.: Издательский дом «Вильямс», 2006. — 1152 с.
13. Джексон П. Введение в экспертные системы / П. Джексон — М. : Издательский дом «Вильямс, 2001. — С. 624. — [ISBN 0-201-87686-8](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0201876868).
14. Таунсенд К. [Проектирование и программная реализация экспертных систем на персональных ЭВМ / К. Таунсенд, В. А. Кондратенко, С. В. Трубицына.](http://web.archive.org/web/20060525230925/http:/www.forth.org.ru/~kp/taunsend.zip) — М.: Финансы и статистика, 1990. — 320 с.
15. Гаврилова Т. А. Базы знаний интеллектуальных систем. Учебник. / Т. А. Гаврилова, В. Ф.  Хорошевский — СПб.: Питер, 2000.

# Додаток А Блок-схема методу визначення кольорової преференції



