Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Арнаутова Олена Миколаївна,
студент групи АС-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмний засіб для кількісної оцінки характеристик інтерфейсу користувача програмних систем

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня професійна програма:
Інженерія програмного забезпечення

Керівник:
Крісілов В. А., д.т.н., професор

Одеса – 2022

Міністерство освіти і науки України

Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____Комлева Н. О.

"_____"_____2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Арнаутової Олени Миколаївни, група АС-171

1. Тема роботи: «Програмний засіб для кількісної оцінки характеристик інтерфейсу користувача програмних систем»
   Керівник роботи: Крісілов В. А, професор кафедри ІПЗ
   затверджені наказом ректора від «20 » жовтня 2022 року № №399-в
2. Зміст роботи: огляд та характеристика сучасних методик оцінки характеристик інтерфейсу, розробка кількісної оцінки характеристик інтерфейсу користувача програмних систем, розробка програмного продукту для кількісної оцінки, керівництво користувача та тестовий приклад.
3. Перелік ілюстративного матеріалу: Згідно до слайдів презентації

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

5. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1.	Аналіз предметної області, збір вимог замовника	20.09.2022	
2.	Розгляд аналогів, прийняття рішення щодо необхідності розробки	30.09. 2022	
3.	Визначення вимог до програмного, апаратного забезпечення	10.10. 2022	
4.	Визначення технологій розробки	20.10.2022	
5.	Проектування програмної системи	28.10.2022	
6.	Програмна реалізація системи	17.11.2022	
7.	Тестування програмної системи	30.11.2022	
8.	Оформлення пояснювальної записки та графічного матеріалу	04.12.2022	

Здобувач вищої освіти **О. М. Арнаутова**

Керівник роботи **В. А. Крісілов**

АНОТАЦІЯ

У дипломній роботі поставлено та вирішено завдання розробки методики кількісної оцінки характеристик призначеного для інтерфейсу програмних систем, яка не вимагає великої кількості витрат для залучення користувачів або експертів і володіє досить низьким рівнем суб'єктивності. Розроблена методика істотно полегшить оцінювання інтерфейсів програмних систем.

Метою даної роботи є зменшення часу оцінки характеристик інтерфейсу користувача програмних систем шляхом формалізації кількісної оцінки характеристик інтерфейсу.

Програма для методики кількісної оцінки юзабіліті інтерфейсів програмних систем розроблена на мові програмування C#.

Ключові слова: юзабіліті, інтерфейс, складність, експертна оцінка, кількісна оцінка, анкетування користувачів.

ABSTRACT

In the thesis the task of developing the examined methodology of quantitative assessment of user interface, which does not require a large number of costs for the involvement of users or experts and has a very low level of sub-activity, is set and solved. The developed methodology will significantly facilitate evaluation of program system interfaces.

The purpose of this work is to reduce the time of evaluation of the characteristics of the user interface of software systems by formalizing the quantitative evaluation of the characteristics of the interface.

The program for the methodology of quantitative assessment of the usability of software system interfaces is developed in C# programming language.

Keywords: usability, interface, complexity, expert evolution, quantitative assessment, user survey.

ЗМІСТ

ВСТУП

Одним з найбільш актуальних питань при створенні програмногозабезпечення і зокрема web-додатків, залишається розробка зручного механізму взаємодії з користувачем. Інтерфейс є найважливішою частиною інформаційної системи, і його якість багато в чому гарантує успішність продукту і його конкурентоспроможність на ринку.

Інтерфейс користувача часто розглядається як зовнішній вигляд програмного забезпечення. Однак користувачі сприймають через інтерфейс систему в цілому, і тоді така концепція є занадто вузькою. Розвиток інформаційних систем показує, що конкуренція програмних продуктів переміщується з області функціональності в область зручності використання

Користувальницький інтерфейс (user interface або скорочено UI) - це інтерфейс, за допомогою якого людина може управляти програмним забезпеченням або апаратним обладнанням. UI повинні бути зручними у використанні, щоб взаємодія з ними відбувалося на максимально інтуїтивному рівні. [3]

Проблематикою проектування і аналізу призначеного для користувача інтерфейсу займається така галузь програмної інженерії, як юзабіліті-інженерія (usability engineering). [1]

Юзабіліті - поняття в мікроергономіці, що відображає ступінь зручності предмета для застосування користувачами при досягненні певних цілей в певному контексті. Базисом для виникнення юзабіліті послужила робота в інтерактивних середовищах. У загальному плані юзабіліті - це науково-прикладна дисципліна, що служить по підвищенню ефективності, продуктивності і зручності користування земельними інструментами діяльності. Вона вивчає і реалізує про процеси створення сукупності властивостей інструмента, які впливають на ефективність його використання в конкретній предметній діяльності. Виражається в застосовності даного інструменту, легкості, природності його використання, безпомилковості, супроводжуваних задоволенням користувача. Можна

стверджувати, що юзабіліті займається споживчими якостями продукту. На відміну від ергономіки, яка спрямована на підвищення ефективності людино - машинної системи в цілому, юзабіліті цікавить тільки ефективність системи щодо споживача (користувача). Їй важливо, щоб система була зручною для людини.

Особливо широке застосування юзабіліті отримала в сфері створення і експлуатації комп'ютерних інтерфейсів. Саме тут відзначені основні успіхи цього напрямку.

Залежно від інструменту і сфери діяльності виділяють software usability - розробка програмних продуктів і web-usability - розробка і вдосконалення веб-сайтів. Термін має зв'язок з поняттям «ергономічність», але на відміну від останнього менше асоціюється з технічної естетикою, з зовнішнім виглядом і більш прив'язаний до утилітарності об'єкта. Українськомовний аналог - зручність використання.

Розробники програмного забезпечення стикаються з проблемою дизайну інтерфейсу, який дозволяє ефективно та легко використовувати програмне забезпечення. У цих умовах юзабіліті методи дизайну стали технологією, яка забезпечує успіх проекту. Проблема в тому, що часто неможливо визначити, який варіант оформлення інтерфейсу користувача краще. Емпіричні суб’єктивні оцінки не можуть бути найкращим критерієм вибору інтерфейсу. Тому необхідні кількісні методи оцінки інтерфейсу користувача. Різні варіанти оформлення інтерфейсу можна оцінити за різними критеріями за допомогою кількісних методів і вибрати найкращий за критерієм пріоритету або сукупністю критеріїв пріоритету.

Основними характеристиками юзабіліті є доступність, ефективність, корисність і продуктивність.

Існує кілька методик оцінки юзабіліті користувальницьких інтерфейсів. Із залученням експертів, із залученням користувачів і кількісні методи.

Метою даної роботи є зменшення часу оцінки характеристик інтерфейсу користувача програмних систем шляхом формалізації кількісної оцінки характеристик інтерфейсу.

Для досягнення даної мети необхідно вирішити такі завдання:

- Проаналізувати існуючі методики кількісної оцінки характеристик інтерфейсу користувача програмних систем .
- Розробити методику кількісної оцінки характеристик інтерфейсу користувача програмних систем.
- Розробити програмний продукт для методики кількісної оцінки характеристик інтерфейсу користувача програмних систем.
- Провести дослідження розробленої кількісної оцінки характеристик інтерфейсу користувача програмних систем.
- Зробити висновки.

Переваги використання методу із залученням експертів: досить швидко; не сильно ресурсовитратності; якісна оцінка юзабіліті-характеристик; опис конкретних проблем і чітких інструкції щодо їх усунення.

Основним недоліком методів оцінки юзабіліті із залученням користувачів є його висока ресурсовитратність. Що стосується методів експертної оцінки, то тут можна виділити високу суб'єктивність.

У першому розділі були розглянуті існуючі аналоги та визначено основний функціонал системи та не функціональні вимоги. У другому розділі було описано розробку кількісної оцінки для інтерфейсів програмних систем. У третьому розділі було визначено програмні класи, описано вимоги до розробленої системи.. У четвертому розділі описано, як саме виглядає система для користувача та на прикладі двох подібних програм було розраховано кількісну оцінку.

1 ОГЛЯД ТА ХАРАКТЕРИСТИКА СУЧАСНИХ МЕТОДИК ОЦІНКИ ХАРАКТЕРИСТИК ІНТЕРФЕЙСУ

У даному розділі розглянуті існуючі методи оцінки юзабіліті інтерфейсів програмних систем, які використовуються часними фірмами та лабораторіями, які займаються експертною оцінкою та проводять дослідження у цій області.

Існує кілька методик оцінки юзабіліті користувальницьких інтерфейсів. Із залученням експертів, із залученням користувачів, кількісні та формальні методи.

Кількісні дослідження містять здійснення різних опитувань, заснованих на використанні структурованих питань закритого типу, на які відповідає велика кількість респондентів. Основним завданням кількісних досліджень є отримання чисельної оцінки стану ринку або відповідної реакції респондентів на певну подію. Такі дослідження застосовуються, коли необхідні точні, статистично надійні чисельні дані. [4]

Частина програмної системи, яка забезпечує інтерфейс користувача, - один з найбільш нетривіальних об'єктів для верифікації. Не тривіальність полягає в дуальному сприйнятті терміну "інтерфейс користувача". З одного боку, інтерфейс користувача (ІК) - це частина програмної системи. Відповідно до ІК пишуться функціональні та низько рівневі вимоги, за якими складаються тест-вимоги та тест-плани. Вимоги визначають реакцію системи на кожне введення користувача та вигляд інформаційних повідомлень системи, які виводяться на екран, пристрій друку або інший пристрій виведення. При верифікації таких вимог мова йде про перевірку функціональної повноти ІК - наскільки реалізовані функції відповідають вимогам, чи коректна виводиться інформація на екран. З іншої сторони, ІК - це "обличчя" системи, і від його продуманості залежить ефективність роботи з системою. Фактори, які впливають на ефективність роботи, слабко підлягають формалізації у вигляді конкретних вимог до окремих елементів, однак повинні бути враховані у вигляді загальних рекомендацій та принципів побудови інтерфейсу. Перевірка інтерфейсу на ефективність людино-машинної взаємодії одержала назву тестування зручності використання (usability verification). В останні роки було проведено ряд порівняльних аналізів,

спрямованих на визначення переваг графічних ІК над традиційними текстовими інтерфейсами для виконання стандартних задач. Одним з найбільш відомих досліджень є звіт Wharton Report під назвою "The Value of GUIs", який пропонує 7 переваг ГІК на основі одержаних результатів тестування. Вони полягають в тому, що користувачі ГІК:

- працюють швидше;
- виконують більше задач;
- мають більш високу продуктивність;
- зазнають меншої розгубленості;
- менше втомлюються;
- мають більше можливостей для самонавчання та дослідження додатків;
- здатні до вивчення більшої кількості можливостей додатку.

Порівняльне тестування - випробовування товарів чи послуг, яке проводиться незалежними організаціями користувачів з метою вивчення властивостей продукції в даному сегменті ринку за допомогою їх порівняння між собою та інформування спільноти про одержані результати. При порівняльному тестуванні операційних систем основну увагу приділяють наступним факторам:

- час виконання задачі в системі;
- показник успішності виконання задачі;
- продуктивність системи;
- ступінь задоволеності користувачів;
- переваги користувачів.

Висновки про порівняльні тести: порівняльні тести можуть надати корисну інформацію при дослідженні зручності застосування нових версій програмного продукту; слід використовувати задачі загального характеру і відомі проблеми у якості базових задач, щоб провести точні порівняльні вимірювання між версіями продуктів; слід застосовувати вимірювання зручності застосування; для зменшення суб'єктивності сприйняття слід використовувати зовнішніх, незалежних виконавців для планування та проведення тестування на зручність

використання.

Тестування зручності використання ІК

Зручність використання є "клеєм", що скріплює всі частини, які повинні об'єднатись разом, щоб скласти програмний продукт. Тобто, поняття "зручність використання" об'єднує в єдине ціле бізнес-процес, технологію, інтерфейс користувача, електронну підтримку виконання задачі. Зручність використання повинна бути частиною проекту і проходити тестування в процесі проектування та розроблення. Вона повинна мати операційне визначення (щоб її можна було виміряти) та забезпечувати тестування. Головне при тестуванні зручності використання інтерфейсу - правильно обрати методи. Проблема розробки якісного ІК та проведення тестування на зручність використання надзвичайно важливі та взаємозв'язані. Вдалий проект інтерфейсу ще не гарантує, що продукт буде зручно використовувати, і в той же час тестування за участю користувачів в жодній мірі не заміняє якісної розробки. Обидва питання складають частину процесу розроблення інтерфейсу, яка називається проектуванням зручності використання. Раніше проектувальники вдосконалювали програмні продукти, додаючи до них додаткові функції. Для вдалого продажу основний акцент робився на кількість функцій продукту, при цьому мало кого цікавило, як користувачі їх використовуватимуть. Сучасні проектувальники ПЗ орієнтуються на задачі, які стоять перед користувачами. Зручність використання ІК (usability) - показник його якості, який визначає кількість зусиль, необхідних для вивчення принципів роботи з ПЗ з використанням пропонованого інтерфейсу. Отже, зручність використання визначає ступінь простоти доступу користувача до функцій системи, наданих посередництвом інтерфейсу. У комп'ютерній галузі термін "зручність застосування" трактується надто вільно. Тому, на сьогодні, пропонують декілька категорій загального характеру, за допомогою яких можна дати чітке визначення терміну "зручність використання" з точки зору комп'ютерного ПЗ в різних системних середовищах та середовищах користувача. До цих категорій слід звернутись під час розроблення анкет, контрольних списків або керівних принципів для оцінювання програмних продуктів. Категорії

зручності використання ІК представлені на рис.1.1.

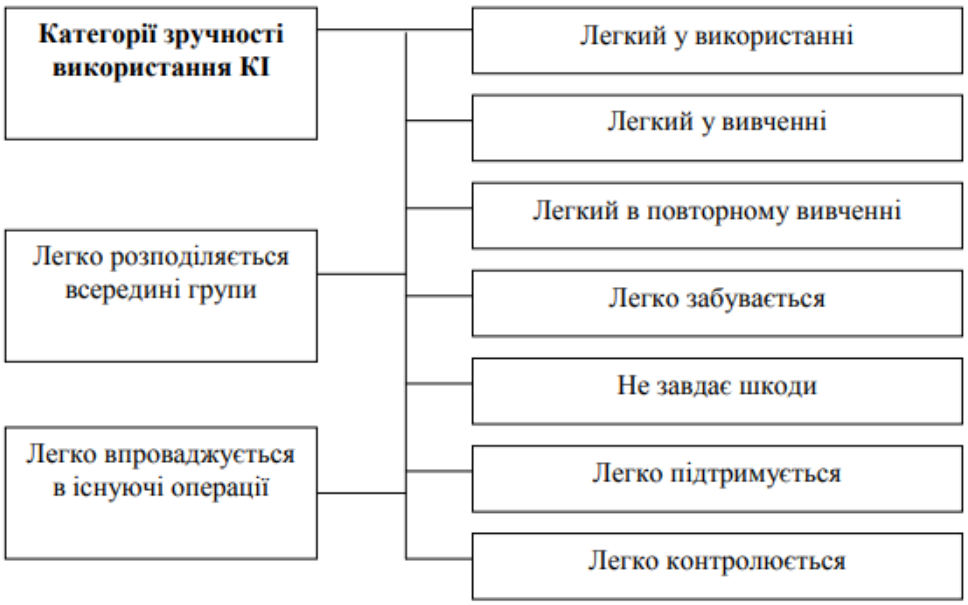

Рисунок 1.1 - Категорії зручності використання ІК

Вже на етапі проектування ПЗ тестування його на зручність використання дозволить зрозуміти, чи зможуть користувачі використовувати цей продукт. Основні причини важливості тестування на зручність використання: 1) інтуїція розробників та проектувальників не завжди забезпечує кращі рішення; 2) термінологія розробників і проектувальників не завжди співпадає з термінологією, до якої звикли користувачі; 3) у природі не існує "середньостатистичного" користувача; 4) інструкції та керівні принципи щодо розроблення зручності використання не є повними та досконалими; 5) для проведення оцінки якості продукту недостатньо інформації від користувачів, одержаної по телефону чи електронною поштою; 6) витрачені на проведення тестування час, гроші та ресурси завжди окупаються; 7) продукти, створені частинами, часто є несумісними на системному рівні; 8) проблеми, виявлені на завершуючих стадіях розробки, складніше і найладніше виправляти; 9) усунення помилок під час проектування дозволить скоротити витрати на подальшу підтримку ПЗ; 10)оцінка зручності використання надає переваги перед конкуруючими продуктами. Міжнародна організація стандартизації (ISO) дає наступне визначення: "Зручність використання - це ефективність, рентабельність

та задоволення, з яким користувачі можуть виконати ті чи інші задачі в заданому середовищі". Тестування на зручність використання проводиться для того, щоб оцінити якість продукту і вияснити, наскільки він ефективний, рентабельний та чи задоволені ним користувачі. Тестування на зручність використання здійснюється на різних етапах розробки продукту, щоб забезпечити зворотній зв'язок з користувачами. Це допомагає вдосконалювати весь проект, скорочує кількість помилок, а також підтверджує відповідність продукту вимогам, які до нього висуваються.

Методи оцінки зручності використання:

1) кількісні - методи оцінки функцій, які передбачають підрахунок дій, визначення повноти виконання задачі, підрахунок часу, помилок та звернень по допомогу;

2) якісні - суб'єктивні методи, які передбачають збирання усних та письмових повідомлень користувачів про їх сприйняття, думки, судження, переваги, а також ступінь задоволеності системою.

Перш ніж планувати і проводити тестування на зручніть використання продукту, слід чітко визначити цілі і задачі, які стоять перед ним. Бут (Booth) виявив 4 фактори, що складають зручність використання: корисність, ефективність, простота вивчення та відношення користувача. Шекель (Shackel) теж розбиває зручність використання на 4 схожих категорії: простота вивчення, ефективність, гнучкість та відношення користувача. Цілі і задачі повинні визначатись для всіх програмних продуктів. Цілі - це забезпечення переваг продукту перед конкуруючими в простоті вивчення, ефективності, гнучкості і т.і. Цілі пов'язані з чотирма факторами зручності використання. Самі по собі вони не підлягають безпосередній оцінці, а повинні бути розділені на задачі, які є уточненням цілей. Вони більш конкретні і детальні, їх можна оцінити і виміряти. Досягнення однієї цілі може вимагати розв'язку багатьох задач. Задачі вибудовуються таким чином, щоб містити інформацію по окремих діях чи операціях. Тестування зручності використання ІК не належить до класичних методів тестування ПЗ. Спеціаліст по тестуванню ІК повинен мати знання як в

галузі програмної інженерії, так і в галузях фізіології, психології та ергономіки. На зручність використання ІК впливають фактори, представлені на рис.1.2. Легкість навчання показує, чи швидко користувач вчиться використовувати систему. Ефективність навчання показує, як швидко користувач працює після навчання. Запам'ятовуваність навчання показує, чи легко запам'ятовується все, чому користувач навчився. Частота помилок показує частоту появи помилок під час роботи користувача. Загальна задоволеність показує, чи є загальне враження від роботи із системою позитивним.

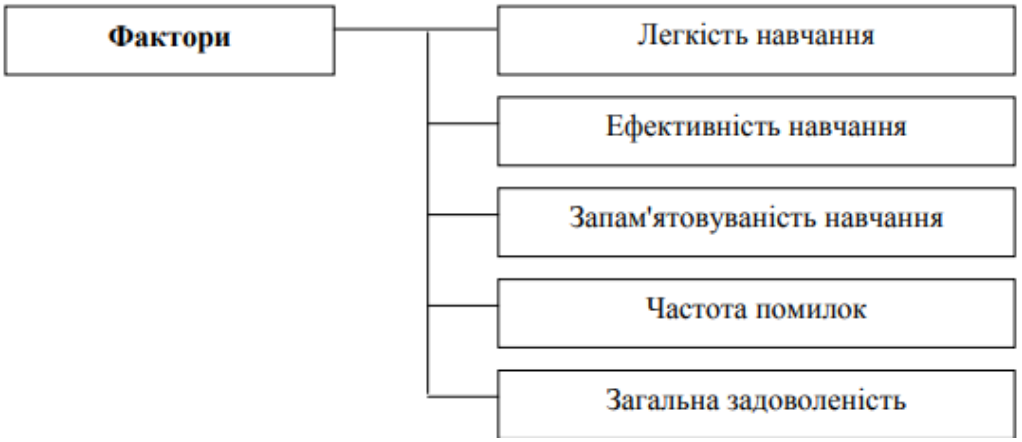

Рисунок 1.2 - Фактори, які впливають на зручність використання ІК

1.1 Сучасні методи оцінки юзабіліті інтерфейса програмних систем

Для аналізу інтерфейсів повинні бути вироблені критерії оцінки інтерфейсів користувача і відповідні їм кількісні шкали, що спираються, в першу чергу, на загальновизнані стандарти в галузі проектування. Існує безліч методів кількісного аналізу інтерфейсів користувача. Умовно їх можна розділити такі основні групи.

1. Аналітичні методи, що дозволяють змоделювати та провести аналіз виконання того чи іншого завдання оператором та на підставі отриманих даних провести оцінку якості ПІ.

2. Експериментальні, спрямовані на отримання якісних та кількісних оцінок під час виконання професійних завдань оператором за допомогою ПІ. Експериментальні методи можуть проводитися на тренажерах (лабораторні) або реальних умовах на реальних робочих місцях (натурні). Експериментальні методи

дають хороші результати на етапах технічного проекту та дослідної експлуатації, за рахунок виявлення та усунення недоліків створюваних спеціалізованих систем.

3. Анкетування, що дозволяють оцінити ставлення операторів до ПІ та (або) його окремим складовим, а також виявити основні проблеми, що виникають у операторів, при роботі з системою.

4. Експертні, що дозволяють при дослідженні ПІ експертами виявляти переваги та недоліки, які, надалі, необхідно усунути. У цьому випадку розглядаються аналітичні методи: GOMS, KLM, а також методика оцінки складності системи Тіма Комбера та Джона Мелтбі. GOMS – це сімейство методів, що дозволяють оцінити час виконання завдання як основний критерій якості. KLM є одним з різновидів методу GOMS, в якому використовуються тільки оператори. Методика оцінки складності системи Тіма Комбера та Джона Мелтбі базується на типах елементів інтерфейсу та їх характеристик.

1.1.1 Кількісні оцінки, що базуються на експериментальних даних

Кількість помилок

Запис будь-яких ненавмисних дій, помилок, промахів, помилок або прорахунків які користувач робить при спробі вирішити задачу. Надалі можна помилки або класифікуються за категоріями. Середня кількість помилок при виконанні завдання становить 0,7. При дослідженні на вибірці з 719 завдань, Джефф Савур встановив, що при підрахунку кількості промахів і помилок, у двох з кожних трьох користувачів (2/3) була помилка. Тільки 10% всіх спостережуваних завдань були виконані безпомилково.

Середній рівень виконання задач

Базова юзабіліті-метрика виконання завдань - найпростіший спосіб виміряти юзабіліті. Зазвичай записується у вигляді довічних показників (1 завдання виконане, 0 - завдання провалена). Якщо користувачі не можуть досягти своїх цілей, то інше вже не важливо. Середній рівень виконання завдань становить 78% (після дослідження понад 1100 випадків Джеффом Савур і Measuring Usability).

Об'єднана метрика (Single Usability Metric, SUM)

SUM є середнім значенням суми кількох метрик: метрики завершеності завдань, тимчасового показника і метрики складності завдання. Середній бал SUM складає 65%. Це для 3-метричної оцінки SUM. Він буде вище для 4-метричної оцінки, яка включає в себе помилки.

1.1.2 Оцінки що базуються на порівнянні і відповідності середовища

Стандартизованість і відповідність робочому середовищі

Правила середовища (операційної системи) створюються і розповсюджуються розробниками операційної системи. Вибір розробником стандарту типу призначеного для користувача інтерфейсу, адекватної предметної області і використовуваної ОС, потенційно має забезпечити, хоча б частково, виконання таких принципів якості призначеного для користувача інтерфейсу, як природність і узгодженість в межах робочого середовища. Явний облік синтактіки інтерфейсу полегшує створення однорідного за стилем і передбачуваного для користувача інтерфейсу. Крім того, при розробці самого стандарту вже враховувалися базові принципи проектування користувальницького інтерфейсу.

Великі фірми, такі як Microsoft, Apple або Google, мають спеціальні гайдлайни для розробників, які декларують основні закони побудови інтерфейсів для даної операційної системи.

Гайдлайни носять рекомендаційний характер, дотримання ним гарантує тільки мінімальний рівень якості. Для складних і піонерських додатків вимога забезпечення функціональної повноти може вступити в протиріччя з обмеженими можливостями, наданими стандартом керуючих засобів призначеного для користувача інтерфейсу.

Бенчмаркінг

Однією з найпоширеніших практик є порівняння продукту з прямими конкурентами. Бенчмаркінг - важливий етап розробки відповідно до філософією постійного управління якістю. Але він не може бути використаний як джерело правдоподібних і значущих для покупця метрик. Бенчмаркінг забезпечує більшою мірою отримання інформації про функціональні недоліки, а не про кількісні

метриках.

Експертна оцінка

Найяскравіше гідності експертної оцінки (ЕО) виявляються в порівнянні її з юзабіліті-тестуванням. Ключове питання - що знаходить тестування? Відповідь - великі проблеми інтерфейсу. Якщо використовується величезна вибірка, вдається виявити і частина дрібних. Але якщо тестування показує тільки погані фрагменти інтерфейсу, як можна виявити фрагменти недостатньо хороші? Як тестування може показати, наприклад, некоректні фрагменти інтерфейсу, недостатньо серйозні, щоб викликати людські помилки, але знижують швидкість роботи?

Досвід експерта дозволяє виявити:

1. Експерт добре знає стандарти на інтерфейс, як писані, так і неписані. Самі по собі порушення стандартів не погані і не хороші, але якщо вони не виправдані, тобто з'явилися випадково, їх краще усунути.
2. Експерт свідомо уважний і, що набагато важливіше, дивиться на інтерфейс свіжим поглядом. Це дозволяє йому знаходити безліч дрібниць, що балансують між багом і проблемою інтерфейсу (наприклад, неузгодженість термінології), які самим розробникам вже не видно.
3. Експерт має досвід, що дозволяє йому безпомилково визначати проблеми, які проявлялися в його професійній діяльності раніше. Якщо якесь рішення не спрацювало в минулому, воно, по видимості, не спрацює і в майбутньому.

Якість експертної оцінки може бути сильно покращено, якщо збільшити кількість експертів.

У той же час експертна оцінка має і низку недоліків:

1. Якщо погано проведене тестування не показує всіх проблем, то погано проведена оцінка не тільки не показує частину проблем, але, ще гірше, показує проблеми неіснуючі. Виправлення цих проблем коштує грошей, але уникнути появи їх у звіті по ЕО неможливо.

2. ЕО нездатна виявити унікальні проблеми конкретного інтерфейсу (вони унікальні, а значить, експерт з ними ще не стикався).

3. Застосування формальних методів аналізу інтерфейсу покращує результат, але також значно збільшує вартість ЕО.

Кількість виявлених проблем при ЕО залежить не тільки від досвіду експертів, а й від їх кількості, через що замовляти роботу доводиться у кількох різних виконавців. В результаті для замовника ЕО організаційно складніше юзабіліті-тестування.

Тривалість експертної оцінки в годинах можна приблизно передбачити наступною формулою:

$$X \times 0{,}3 + X + Y \times 0.6\,, \qquad (1.1)$$

де X - число екранів і вікон системи (на більшу частину екрана новий системи буде потрібно хвилин десять, але приблизно п'ята частина потребує більше часу), а Y - кількість призначених для користувача задач (вони вимагають окремого проходу по інтерфейсу).

Евристична оцінка

Евристична оцінка була розроблена Якобом Нільсеном і Рольфом Молічем, які сподівалися з її допомогою скоротити тривалість проведення перевірки по контрольному списку. При евристичної оцінкою замість десятків і сотень конкретних вимог інтерфейс перевіряється на відповідність всього декільком загальним принципам.

Сам Нільсен рекомендував наступні принципи:

- У будь-який момент часу система показує, що з нею відбувається.
- Система використовує терміни, поняття та метафори, присутні в реальномусвіті, а не обумовлені комп'ютером.
- У будь-який момент користувач контролює систему, а не навпаки. Будь-яку команду можна скасувати або повторити.
- У будь-який момент часу система виглядає і функціонує однаковим і

стандартним способом.

- Інтерфейс системи перешкоджає появі людських помилок.
- У будь-який момент часу інтерфейс показує об'єкти і команди сам, не вимагаючи від користувача згадувати їх.
- В інтерфейсі є методи прискорення роботи, призначені для досвідчених користувачів і не заважають користувачам недосвідченим; завдяки таким прискорювачів досвідчені користувачі отримують резерв для підвищення власної продуктивності.
- Інтерфейс естетичний і в будь-який момент часу не містить непотрібної зараз інформації.
- Інтерфейс допомагає користувачам виявляти і виправляти проблеми, включаючи людські помилки.
- Довідка доступна в будь-який момент часу. Вона достатня, але не надлишкова; до неї легко звертатися; вона не абстрактна, а націлена на вирішення конкретних завдань користувача; в ній описуються конкретні кроки щодо вирішення проблем.

Евристична оцінка ніяк не враховує діяльності користувачів, в кращому випадку вона покриває виконуються користувачами операції. В результаті евристична оцінка не виявляє структурні проблеми інтерфейсу, які, як правило, помітно важливіше локальних проблем.

Формальні методи оцінки

Інформаційний пошук

До інформаційного пошуку, тобто знаходженню на моніторі об'єкта із заданим ознакою, зводиться багато видів діяльності користувача.

Математичне очікування часу пошуку обчислюється за формулою:

$$r = \frac{\frac{N}{y} - t}{M+1} \qquad (1.2)$$

де N - загальний обсяг інформації (кіль кість елементів на екранікористувача);

M - кількість елементів, що володіють за даними для пошуку ознакою;

η - оперативний обсяг зорового сприйняття;

t - тривалість зорової фіксації.

Величина η обмежена обсягом оперативної пам'яті (5 ± 2 елементи) і оперативним полем зору 10 градусів. Тривалість фіксації t залежить від способу кодування інформації та складності виконання завдання. Для визначення t в кожному конкретному випадку використовують методику, засновану на поєднанні експериментального і аналітичного методів.

При цьому передбачається проведення невеликого за обсягом експерименту. Суть методу полягає в наступному. Вибирають m-різних завдань, що вирішуються користувачем. Кожне завдання розбивають на n-різних, але однакових для кожного завдання типів елементарних дій. Число дій j-го виду в i-тій задачі позначимо a_{ij}. Тоді середнє значення часу рішення і-того завдання одно:

$$a_{i_1} t_1 + a_{i_2} t_2 + \cdots + a_{i_n} t_n = T_i \qquad (1.3)$$

де t_j- середнє значення часу виконання j-го елементарної дії (j = 1 ... n).

Якщо є m завдань, що розрізняються значеннями a_{ij}, то можна отримати m лінійних алгебраїчних рівнянь з n невідомими t_j.

На підставі теореми про складання дисперсій незалежних величин маємо:

$$a_{i1}\sigma_1^2 + a_{i2}\sigma_2^2 + \cdots + a_{in}\sigma_n^2 = S_i^2 \qquad (1.4)$$

де σ^2 - дисперсія часу виконання j-го дії;

S^2- дисперсія часу рішення і-того завдання.

Дисперсія S_i^2 визначаються в результаті рішення системи з m рівнянь. Величини T і S беруться з експерименту. Виходячи з розрахованого часу інформаційного пошуку можна визначити складність інтерфейсу.

Інформаційна продуктивність

E - інформаційна продуктивність інтерфейсу. Визначається як відношення

мінімальної кількості інформації, необхідної для виконання завдання, до кількості інформації, яке має ввести користувач. Параметр E може змінюватися в межах від 0 до 1.

Якщо ніякої роботи для виконання завдання не потрібно або робота просто не проводиться, то продуктивність становить 1. (Це формальне положення вводиться для того, щоб уникнути поділу на 0). Продуктивність E може дорівнювати і 0 у випадках, коли користувач повинен ввести інформацію, яка абсолютно марна. Слід зазначити, що в інтерфейсах можна зустріти чимало деталей, які мають сумнівну цінність через параметр E = 0. Прикладом такого марного елемента може бути діалогове вікно, в якому є тільки одна-єдина можливість для дії користувача, наприклад кнопка OK.

Інформація вимірюється в бітах. Один біт, який являє собою один з двох альтернативних варіантів (0 або 1, так чи ні), є одиницею інформації.

Наприклад, щоб вибрати один з будь-яких чотирьох об'єктів, буде потрібно 2 біти інформації. Якщо об'єкти позначити як A, B, C і D, перший біт інформації визначить вибір між A і B або C і D. Коли перший вибір зроблений (наприклад, C і D), другий біт визначить вибір між наступними двома елементами (або C, або D). Двох довічних виборів, або двох бітів, досить для вибору одного елемента з чотирьох. Щоб зробити вибір з групи восьми елементів, буде потрібно 3 біта. З шістнадцяти елементів - 4 біта, і т.д.

У загальному випадку при кількості n рівно можливих варіантів сумарна кількість переданої інформації (I):

$$I = \log_2 n \qquad (1.5)$$

Кількість інформації для кожного варіанта визначається за формулою:

$$I = \log_2 n/n \qquad (1.6)$$

Якщо ймовірності для кожної альтернативи не є рівними 0 і i-я альтернатива має ймовірність p (i), то інформація, що передається цією альтернативою, визначається за формулою:

$$I = p(\text{i}) \times \log_2(1/p(\text{i})) \quad (1.7)$$

Кількість інформації є сумою по всіх варіантах. Звідси випливає, що інформаційний зміст інтерфейсу, в якому можливо зробити тільки натискання єдиної клавіші (а ненажатом клавіші не допускається), становить 0 біт.

Інформаційна продуктивність інтерфейсу і швидкість виконання - слабо корелюють величини. Рішення, що дозволяє запитувати у користувача менше інформації може займати більше часу, і навпаки. Час, необхідний користувачу для вибору з різноманіття варіантів описує закон Хіка. Закон Хіка- твердження, що час реакції при виборі з деякими го числа альтернативних сигналів залежить від їх числа.

$$T = b * \log_2(n + 1) \quad (1.8)$$

де n - кількість можливих альтернатив,

T - середній час вчинення дії,

b - константа, яка повинна бути визначена емпірічно виходячи зхарактеристик користувача.

Моделі KLM-GOMS

GOMS - модель, заснована на оцінці швидкості друку. Розробники моделі GOMS під час її створення помітили, що час, потрібний для виконання якогось завдання системою «користувач-комп'ютер», є сумою всіх тимчасових інтервалів, які потрібні були системі на виконання послідовності елементарних жестів, що складають дану задачу. Хоча для різних користувачів час виконання того чи іншого жесту може сильно відрізнятися, дослідники виявили, що для більшої частини порівняльного аналізу завдань, що включають використання клавіатури і графічного пристрою введення, замість проведення вимірювань для кожного окремого користувача можна застосувати набір стандартних інтервалів. За допомогоюретельних лабораторних досліджень було отримано набір тимчасових інтервалів, необхідних для виконання різних жестів.

- f K (0,28с.) - одне натискання клавіші. Вводяться окремі значення для різних категорій користувачів - експерт (90 ударів в хвилину, 0,12с. На удар), досвідчений користувач (55 ударів в хвилину, 0,2 с. На удар), звичайний користувач (40 ударів в хвилину, 0,28с . на удар), новачок (швидкість друку нерівномірна, 1,2с. на удар). Рекомендується використовувати значення 0,28 с .;
- f T (n) - логічно пов'язана послідовність натискань. (Наприклад, набір слова на клавіатурі), еквівалентно операторам K, узятим кількості n. Обчислюється як T (n) = n × K;
- f P (1,1с.) - вказівка за допомогою маніпулятора типу "миша" на необхідну ділянку екрану;
- f B (0,1 с.) - натискання або відпускання кнопки на маніпуляторі типу "миша";
- f BB (0,2 с.) - натискання, а потім відпускання кнопки на маніпуляторі типу "миша" (одиночний "клік");
- f H (0,4 с.) - переміщення рук з клавіатури на мишу і з миші на клавіатуру;
- f M (0,6-1,35с, рекомендується використовувати значення 1,2 с.) час, необхідне користувачеві для пошуку рішення;
- f W (t) - допустимий час очікування користувачем відповіді системи.

Значення має бути визначено для кожного конкретного випадку.

Для уточнення моделі KLM-GOMS можна використовувати закон Фіттса. Закон Фіттса (або Фітс) визначає швидкість взаємодії користувача з елементами інтерфейсу. Очевидно, час взаємодії залежить від розмірів елемента управління і дистанції, яку належить виконати курсору миші. Одна з математичних формулювань закону Фіттса - формулювання Шеннона - виглядає наступним чином:

$$T = a + b \log_2 \left(1 + \frac{D}{W}\right) \tag{1.9}$$

де T - середній час, необхідний для переміщення курсора,

a і b - константи, що встановлюються дослідним шляхом і представляють час відгуку і швидкість пристрої введення,

D - дистанція,

W - ширина елемента управління, виміряна по лінії руху курсора.

Оцінка складності системи Тіма Комбер і Джона Мелтбі

Позначимо складність системи C. Відповідно до теорії інформаційної ентропії Клода Шеннона, доопрацьованій Джі Бонсіпе складність буде визначатися за формулою:

$$C = -N \sum_{i=1}^{n} p_i \log_2 p_i \qquad (1.10)$$

де N - кількість усіх об'єктів;

p_i - відносини об'єктів в i-тому класі до всіх об'єктів ($p_i = n_i /n$); n - кількість класів об'єктів;

n_i - кількість об'єктів i-го класу.

XAOS — Actions, Organizational elements, Summed entropy of RGBvalues

Метрика розроблена Крістіаном Штікелем для визначення складності інтерфейсу виходячи з аналізу візуальної складової.

Очевидним і простим способом для визначення візуальної складності інтерфейсу є вимір розміру графічного файлу (JPEG, TIFF, GIF) з деяким зразком. Стиснення графічних зображень розглядається не з точки зору зорового сприйняття, а зточки зору теорії інформації.

Візуальноая складність X залежить від трьох факторів:

$$X = A \times O \times S \qquad (1.11)$$

де A - кількість можливих взаємодій, який можна розглядати як функціональні елементи;

O - кількість груп, в яких можна організувати окремі елементи; S - сумарна ентропія RGB.

Усі види посилань або елементів графічного інтерфейсу (такі як кнопки,чек бокси і т.д.) розглядаються як функціональні елементи.

1.2 Набір інструментальних засобів розробки

Мова програмування C# - об'єктно-орієнтована мова програмування, основна мова розробки додатків для платформи Microsoft. NET. Перейнявши багато від своїх попередників - мов C++, Java - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем. Оскільки мова C# є об'єктно-орієнтованою мовою. [9]

C# - об'єктно-орієнтована мова програмування. Розроблений в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберг в компанії Microsoft як мова розробки додатків для платформи Microsoft .NET Framework і згодом був стандартизований як ECMA -334 iISO / IEC 23270.

C# відноситься до сім'ї мов з C - подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (у тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML. [11]Перейнявши багато що від своїх попередників - мов C++, Delphi, Модула, Smalltalk і особливо Java - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинні спадкування класів (між тим допускається множинне спадкування інтерфейсів).

1.3 Висновок

У першому розділі дипломної роботи були розглянуті методи оцінки юзабіліті інтерфейсів програмних систем та переваги і недоліки цих методів.

Проаналізувавши сучасні методи оцінки юзабіліті інтерфейсів можна

зробити висновок, що на сьогоднішній день актуальним є питання розробки такого підходу до аналізу юзабіліті, який би мав досить низький рівень суб'єктивності і не вимагав великої кількості витрат для залучення користувачів або експертів. [2]

Оцінювання юзабіліті користувальницького інтерфейсу – ітераційний процес, що фокусується на використанні ПЗ, а не на його можливостях і функціях. У юзабіліті-інженерії на всіх стадіях розроблення ПЗ для оцінювання якості юзабіліті користувальницького інтерфейсу застосовують різні методи, що дають змогу ідентифікувати проблеми зручності використання користувацького інтерфейсу. Оцінювання якості юзабіліті на всіх етапах розроблення ПЗ дає змогу знизити витрати на розробку й зменшити час розробки; з погляду бізнесу – збільшити доходи від продажів ПЗ; з погляду користувача – підвищити ефективність, продуктивність, задоволеність користувачів під час взаємодії з ПЗ.

Широкий спектр методів може застосовуватися на різних стадіях розроблення КІ. Кожний з розглянутих методів має певні особливості й розкриває різні проблеми usability. Вибір методу зумовлений застосовуваною методологією розроблення; масштабом проекту; типом програмного забезпечення та його цілями, задачами; контекстом використання; апаратною платформою, рівнем кваліфікації експертів тощо.

Було розроблено програмні класи на мові програмування C#.

2 РОЗРОБКА КІЛЬКІСНОЇ ОЦІНКИ ХАРАКТЕРИСТИК ІНТЕРФЕЙСУ КОРИСТУВАЧА ПРОГРАМНИХ СИСТЕМ

Існує кілька методик оцінки юзабіліті користувальницьких інтерфейсів. Із залученням експертів, із залученням користувачів і кількісні методи.

Кількісні дослідження містять здійснення різних опитувань, заснованих на використанні структурованих питань закритого типу, на які відповідає велика кількість респондентів. Основним завданням кількісних досліджень є отримання чисельної оцінки стану ринку або відповідної реакції респондентів на певну подію. Такі дослідження застосовуються, коли необхідні точні, статистично надійні чисельні дані. [4]

Основними характеристиками інтерфейсу є:

1. Доступність;
2. Ефективність;
3. Корисність;
4. Продуктивність.

Потенційний недолік підходу кількісних формальних метрик полягає в неетичність зведення оцінки роботи людини до кількох числовим параметрам і по ним судити про продуктивність і зручність. Повна оцінка додатки або системи можлива тільки експертною групою після отримання експериментальних результатів від користувачів.

Але для невеликих систем доцільно використовувати методику формальних оцінок, яка здебільшого заснована на кількісних висновках. Формальні оцінки виходять недостатньо повними без використання експертної допомоги, але участь експертів зведено до мінімуму, що значно знижує витрати. Через порівняно низьку ціну способу, оцінити рівень ергономічності свого продукту зможе велика кількість невеликих команд розробників.

Не підходять для формальної колічетвенной оцінки будь-які додатки, призначені для творчої діяльності використовують суперконтрол-робочу область (Photoshop, Illustrator, AutoCAD), і мають неформалізовані критерії результату. Придатність для чисельної оцінки визначається експертом.

Зручність використання (Є) програми є деякою функцією:

$$G = f(C, R, I, Q, B), \quad (2.1)$$

де C (complexity) - складність, далі розглядається як функція залежить від кількості класів об'єктів (різноманітність класів) і кількості об'єктів в кожному класі;

R (readability) - легкочітаемость, візуальна гармонійність;

I (information) - інформаційна ефективність (передача тільки необхідної інформації програмою). Можливо, може бути оцінена як довжина послідовності контролів, з якими потрібно взаємодіяти користувачеві, що виконати завдання. У термінах дерева GUI XML - це самий довгий або найглибший шлях чи інакше tree depth;

Q (quickness) - швидкість взаємодії (залежить від взаємного розташування і величини об'єктів, з якими взаємодіє користувач);

B (branching) - максимальна кількість шляхів, за якими користувач може піти, після взаємодії з якимсь контролом. Так само можна розглядати цю метрику як максимальне число дітей контрола в дереві XML.

Так як функція містить занадто багато параметрів, які в даний момент неможливо правильно оцінити, то оцінка буде вестися по параметру C.

2.1 Попередні дослідження. Участь експертів

Розроблювальна методика оцінки базується на ідеї ергономічності як виконання завдань користувача самим швидким і оптимальним шляхом. Тому для оцінки продукту необхідна інформація про користувачів.

Джерело даних. Вимоги до респондентів

Якщо система вже реалізується на ринку, рекомендується виходити з актуальної інформації про користувачів, якщо немає, то використовувати дані попереднього аналізу, або дані про користувачів схожого ПЗ. При розробці ПЗ на основі принципів постійного управління якістю, дані про користувачів повинні бути зібрані ще до початку розробки, як фундамент для проектування. Тому вимога надання статистики про користувачів не повинно здорожувати отримання

оцінки. Якщо ж цих даних немає і отримати їх не представляється можливим, то коефіцієнт значущості для кожного типу користувачів визначається експертом. Актуальна інформація про користувачів системи повинна ґрунтуватися на вибірці не менше ніж з 50 людей. Вибірка може бути отримана за допомогою інструментів, які збирають статистику з уже працюючого продукту, або із спеціально проведеного спостереження за користувачами.

Визначення типів користувачів

Спочатку необхідно зрозуміти для виконання яких завдань користувачі використовують оцінку ПЗ. Завдання - це мета, яка повинна бути досягнута за допомогою програмного забезпечення.

Важливо розуміти, що завданням не може бути, наприклад, встановлення таймеру мікрохвильової печі на 6 хв. 30 сек., Завданням є отримати їжу, підігріту до певної температури або правильно її розморозити. Відповідно, завдання може виконуватися завдяки розумному сканеру, який буде самостійно розпізнавати тип страви і необхідне для нього час розморожування.

Користувачі розподіляються на групи з відповідно з їх завданнями. Такі параметри як вік, стать, знайомство з системою на даному етапі не враховуються. Групи користувачів позначаються відповідно U_1, U_2, U_3 ... $U_к$.

Оптимальна кількість розглянутих завдань повинна бути не занадто великим і не дуже маленьким.

Занадто велика кількість розглянутих завдань призведе до подорожчання і затягування процесу оцінки. Занадто ж невелика кількість завдань дасть не релевантну оцінку. Доцільно виділяти не більше 10 типів користувачів, у кожного з яких буде не більше 3 пріоритетних завдань.

Визначення пропорцій кількості користувачів кожного типу

На цій стадії визначаються коефіцієнти співвідношень розмірів категорій користувачів до кількості всіх користувачів.

$K_n(U_k)$ - (number of users) – коефіцієнт співвідношення кількості користувачів U_k до числа всіх користувачів.

$$\text{Kn (Uk)} = \frac{N(U_k)}{N\sum_{i=1}^{n} U_i}, \quad (2.2)$$

Відповідно, сума кількісних коефіцієнтів буде дорівнювати 100% для всіх користувачів.

Оцінка частоти використання

Кожен з користувачів працює над продуктом протягом якогось проміжку часу. Тривалість проміжку може бути встановлена за допомогою інтерв'ю чи анкетування, але більш надійним способом є спостереження і складання середнього статистичного значення.

$K_s(U_k)$ - (system usage) - відносний коефіцієнт використання системи користувачами U_k.

$$\text{Ks (Uk)} = \frac{\nu(U_k)}{\nu_{max}}, \quad (2.3)$$

де ν_{max} - максимальна частота використання для всіх типів користувачів.

Частота використання визначається з дослідження, якщо продукт або близькі за функціями системи, вже присутні на ринку, або з попереднього аналізу. Частота використання визначається виходячи з часу, яке користувач проводить, взаємодіючи з системою. Так як коефіцієнт відносний, тобто безрозмірний, одиниці можуть варіюватися в залежності від даного дослідження.

Визначення вартості часу користувачів

K_w (U_k) - (user wage) відносний коефіцієнт вартості часу користувачів.

$$K_W(U_k) = \frac{W(U_k)}{W_{max}} \quad (2.4)$$

де W_{max} – максимальна середня зарплата серед користувачів.

Калькуляція коефіцієнта значущості користувача з отриманих даних, або привласнення коефіцієнта експертом

K_u (U_n) - підсумковий коефіцієнт для типу користувачів.

$$Ku\ (Un) = Kw\ (Uk) \times Kn\ (Uk) \times Ks\ (Uk)\ , \qquad (2.5)$$

Ранжування завдань для кожного типу користувачів

Кожній із завдань користувачів, виявлених на першій стадії, присвоюється коефіцієнт важливості у відсотках, виходячи з того, що загальна сума коефіцієнтів повинна дорівнювати 100%. Завдання (tasks) позначаються t_1, t_2, t_3. Коефіцієнти відповідно: C_{t1} (U_1) - коефіцієнт важливості t_1U_1.

Виділення послідовності екранів, необхідних для вирішення завдання

Після виділення завдань, складаються юзкейси, що складаються з черги екранів, які повинен пройти користувач для досягнення своєї мети.

Наприклад, якщо при оцінці виявлено 2 типу користувачів, у першого з яких 2 пріоритетні завдання, а у другого - одна, юзкейси будуть представляти із себе три ланцюжки.

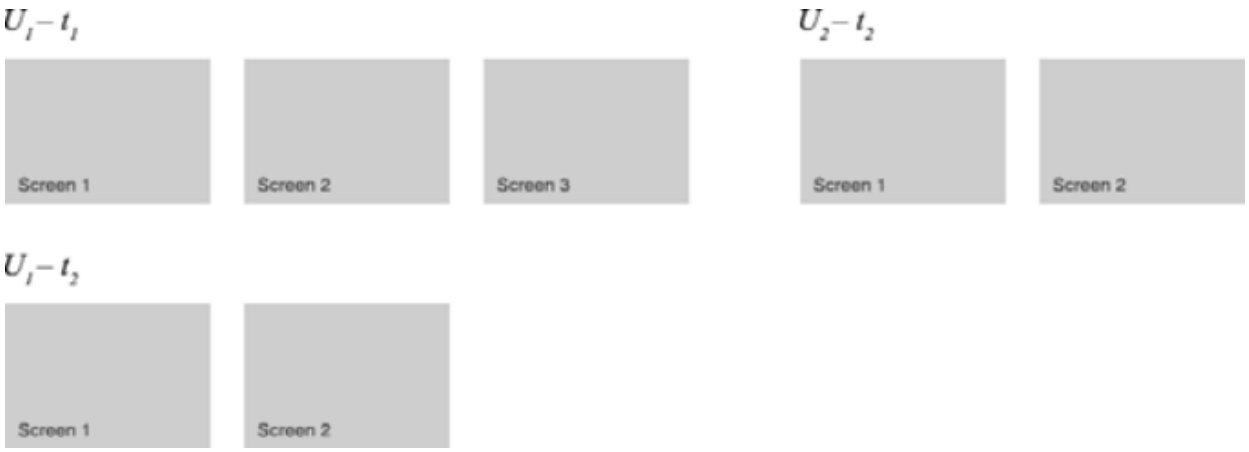

Рисунок 2.1 - Приклад юзкейсов для системи з 2-ма типами користувачів

2.2 Формальний розрахунок

Спочатку оцінюється складність за методикою Тіма Комбер і Джона Мелтбіза формолою 1.10:

$$C = -N \sum_{i=1}^{n} p_i \log_2 p_i$$

Вихідний алгоритм передбачає оцінку тільки одного, головного екрану програми, не враховуючи оцінки декількох послідовних екранів ПЗ.

Для визначення підсумкової оцінки, складність кожного екрану (Csm (Uk)) додатку визначається за формулою 1.10, після чого отримані показники сумуються.

Підсумковий коефіцієнт складності для завдання:

$$C_{tn}(U_k) = \sum_{i=1}^{m} C_{si} \qquad (2.6)$$

Підсумковий коефіцієнт важливості користувача обчислюється за допомогою підсумовування складнощів всіх його завдань з відповідними коефіцієнтами важливості:

$$C_{uk} = \sum_{i=1}^{n} (C_{tn} \times K_{tn}) \qquad (2.7)$$

Підсумкова складність програми обчислюється підсумовуванням складнощів кожного автоматизоване помножених на коефіцієнт важливості конкретного користувача:

$$\epsilon_{\mathrm{uk}} = \sum_{\mathrm{i=1}}^{\mathrm{k}} (\mathrm{C_{uk}} \times \mathrm{K_u}), \qquad (2.8)$$

На заключному етапі до отриманої оцінки за модифікованою методикою Тіма Комбера і Джона Мелтбі підсумовується складність, отримана в ході аналізу графічного інтерфейсу користувача за методикою KLM-GOMS для оцінки часу виконання завдання. [14]

Виділені завдання оцінюються за методикою KLM-GOMS, за участю групи експертів, яка виділяє найбільш оптимальний шлях вирішення поставленої задачі, фіксує еталонні результати та порівнює з ними дані отримані формального розрахунку [15].

Для методики KLM-GOMS існують достовірні статистичні дані щодо часу виконання елементарних, «атомних» операцій, таких як переміщення вказівника миші, набір аргументів на клавіатурі, переміщення руки з миші на клавіатуру і т.д. Виділена оптимальна послідовність передбачає два варіанти використання діалогу як за допомогою клавіатури, так і за допомогою миші. Тому фактичні результати порівнюються із середнім значенням швидкості виконання клавіатури та миші. Таким чином, формула для розрахунку швидкості виконання матиме вигляд:

$$\frac{K_t + M_t}{2}, \tag{2.9}$$

де Kt – час виконання завдання за методикою KLM-GOMS за допомогою клавіатури;

Mt – час виконання завдання за методикою KLM-GOMS за допомогою миші.

2.3 Розбиття на класи

Угруповання (сортування, класифікація) полягає в тому, що досліджувана безліч розбивають на n класів і об'єкт відноситься до одного з них. Усередині кожного класу об'єкти вважаються равно переважними. Чисельні кордону класів будуть визначені експертами після накопичення більшого обсягу статистичної інформації.

2.4 Вимоги до програмного продукту

Функціональні вимоги:

Для дослідження складності інтерфейсу:

1. обчислення складності за методикою Тіма Комбер і Джона Мелтбі;
2. обчислення підсумкового коефіцієнту складності для завдання;
3. обчислення коефіцієнтів важливості користувачів;
4. розбиття на класи;
5. обчислення підсумкової складності інтерфейсу програми.

Нефункціональні вимоги:

− зручність використання – система розроблена таким чином, що вся робота узгоджена у межах робочого середовища і основана на навичках користувачів, отриманих раніше при роботі в середовищі ОС, а також узгодженість у використанні;

− мобільність – програмний засіб легко повинен переноситись з процесора одного типу на процесор іншого типу;

− супроводження – дозволяють мінімізувати зусилля по внесенню змін.

2.5 Висновок

У другому розділі описано запропоновану методику для оцінки якості інтерфейсу користувача. Створення власної методики зумовлювалося тим, що розрізнене використання існуючих методик оцінки якості неможливо повною мірою відчути всі переваги від автомазованого тестування інтерфейсу. Для оцінки були обрані базові методики, а саме: експертна оцінка, розрахунок за методикою Корбера та Мелтбі, а також методика GOMS. Сукупність даних методик спільно дозволить провести оцінку якості найбільш точно, грамотно та обґрунтовано.

В результаті розробки програмного продукту для методики кількісної оцінки було проведено аналіз вимог та формування специфікації вимог до системи.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ КІЛЬКІСТНОЇ ОЦІНКИ ХАРАКТЕРИСТИК ІНТЕРФЕЙСУ КОРИСТУВАЧА ПРОГРАМНИХ СИСТЕМ

3.1 Постановка завдання на розробку

У даному розділі пояснювальної записки описуються технології створення програмного продукту для методики кількісної оцінки характеристик інтерфейсу.

3.2 Вибір та обґрунтування інструментальних засобів

3.2.1 Мова C#

Мова програмування C# - об'єктно-орієнтована мова програмування, основна мова розробки додатків для платформи Microsoft. NET. Перейнявши багато від своїх попередників - мов C++, Java - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем. Оскільки мова C# є об'єктно-орієнтованою мовою [16].

C# - об'єктно -орієнтована мова програмування. Розроблений в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберг в компанії Microsoft як мова розробки додатків для платформиMicrosoft .NET Framework і згодом був стандартизований як ECMA -334 iISO / IEC 23270.

C# відноситься до сім'ї мов з C - подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (у тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [18].

Перейнявши багато що від своїх попередників - мов C++, Delphi, Модула, Smalltalk і особливо Java - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці

програмних систем, наприклад, C# на відміну від C++ не підтримує множинне спадкування класів (між тим допускається множинне спадкування інтерфейсів) .

C# розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, насамперед, від можливостей самої CLR. Це стосується, перш за все, системи типів C#, яка відображає BCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність була порушена з виходом C # 3.0, що представляє собою розширення мови, що не спираються на розширення платформи. NET) CLR надає C#, як і всім іншим. NET- орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування.

Common Language Runtime (CLR - загальномовне виконуюча середа) - віртуальна машина , що інтерпретує та виконуюча код мовою CIL, в який компілюються програми, написані, зокрема, на .NET- сумісних мовах програмування (C#, Managed C ++, Visual Basic. NET, Visual J # і т. п.); компонент пакету Microsoft .NET Framework.

Середа CLR є реалізацією специфікації CLI (англ. Common Language Infrastructure), специфікації загальномовний інфраструктури компанії Microsoft.

CLR інтерпретує і виконує код мовою CIL (реалізація компіляції якого компанією Microsoft називається MSIL), а також надає MSIL - програмам (а отже, і програмами, написаним на мовах високого рівня, що підтримують. NET Framework) доступ до бібліотек класів . NET Framework, або так званої . NET FCL (англ. Framework Class Library) [17].

3.2.2 Платформа розробки

Система створена в інтегрованому середовищі розробки Microsoft Visual Studio 2010 на мові програмування C#.

Середа Microsoft Visual Studio 2010 є сучасним середовищем розробки програмного забезпечення, на базі операційної системи Windows. Дане

середовище надає розробнику великий вибір інструментів для швидкого і якісного створення desktop - систем. Інтегрована підтримка розробки через тестування і нові інструменти налагодження дозволяють швидко і без зусиль знаходити і усувати помилки, забезпечуючи високу якість рішень. Оскільки частка операційних систем Windows становить 90,29% на грудень 2010 року[7], то була обрана саме ця платформа для розробки.

Microsoft Visual Studio - лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки , так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms , а також веб - сайти, веб -додатки, веб-служби як у рідному, так і в керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone. NET Compact Framework і Microsoft Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб -редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як наприклад,Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно - орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

3.3 Опис вхідних та вихідних даних при роботі з програмним продуктом для методики кількісної оцінки характеристик інтерфейсу

Вхідними даними для даної методики є:

Для оцінки характеристик інтерфейсу користувача ПС потрібно ввести наступні дані:

- кількість користувачів;
- присвоїти кожному користувачу коефіцієнт важливості;
- кількість сценаріїв;
- кількість класів об’єктів для усіх сценаріїв;

Вихідними даними для даної методики є:

- таблиця, яка містить важливість кожного завдання та діаграми важливості кожного завдання;
- файли з необхідною інформацією для аналізу складності;
- обчислення коефіцієнтів важливості користувачів;
- обчислення підсумкової складності інтерфейсу програми.

3.4 Специфікація вимог до програмного продукту

Необхідно розробити програмний продукт для методики оцінки юзабіліті інтерфейсів програмних систем, яка дозволить, при залученні експертів, стежити за складністю.

Варіанти використання

Діаграми варіантів використання показують взаємодії між варіантами використання і діючими особами, відображаючи функціональні вимоги до системи з точки зору користувача.

Побудуємо діаграму варіантів використання для розроблюваної програмної системи (див. рис. 3.1), опишемо акторів та прецеденти.

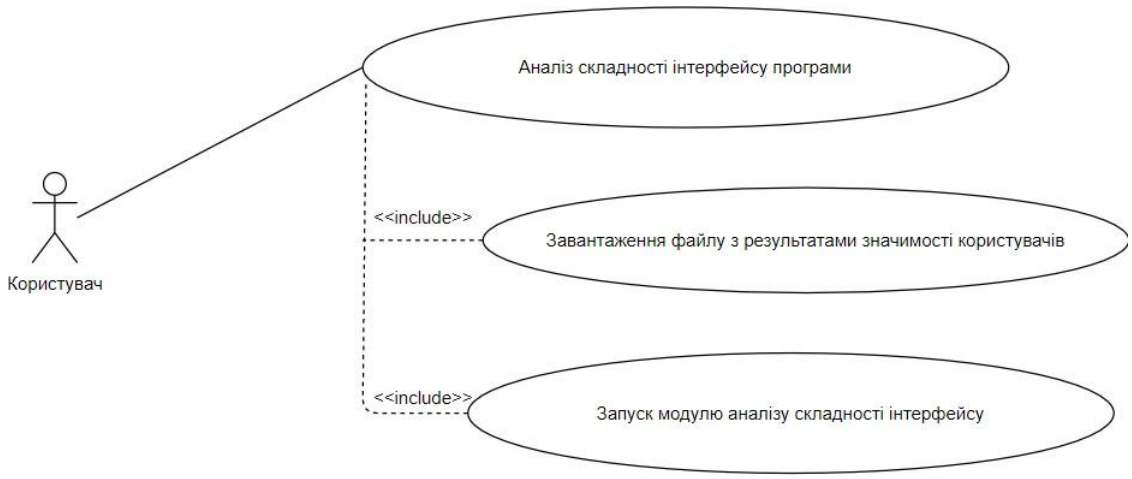

Рисунок 3.1 – Діаграма варіантів використання

Опишемо сценарії для розроблених варіантів використання.

Таблиця 3.1 – Прецедент «Аналіз складності інтерфейсів програмних систем»

Основне діюче лице	Користувач
Область дії	Система
Рівень	Мета користувача
Користувачі та Інтереси	Користувач запускає програму для аналізу інтерфейсу на складність
Предумови	Вхід до програми
Гарантії успіху	Інтерфейс досліджений за крітерієм складності.
Основний сценарій	1. Програма завантажує вибраний файл 2. Програма запускає модуль аналізу складності 3. Запис файлу тестування
Розширення	1.А. Програма не може знайти файл
	1.А.1. Повідомлення про помилку
	2.А. Програма не може знайти даний модуль
	2.А.1. Повідомлення про помилку

Таблиця 3.2 – Прецедент «Завантаження файлу з результатами важливості завдання»

Основне діюче лице	Користувач
Область дії	Система
Рівень	Мета користувача
Користувачі та інтереси	Користувач завантажує файл
Предумови	Вхід до програми
Гарантії успіху	Файл завантажен
Основний сценарій	1. Користувач вводить шлях за яким слід шукати файл з результатами аналізу складності
Розширення	1.А. За введеним користувачем шляху файлу не існує.
	1.А.1. Повідомлення про помилку 1.А.2. Повернення до пункту 1

Таблиця 3.3 – Прецедент «Запуск модуля аналізу складності інтерфейсів»

Основне діюче лице	Користувач
Область дії	Система
Рівень	Мета користувача
Користувачі та інтереси	Користувач натискає на кнопку «Проаналізувати»
Предумови	Файл з результатами аналізу завантажен
Гарантії успіху	Інтерфейс досліджений за критерієм складності.
Основний сценарій	1 . Програма відкриває вибраний файл 2 . Зчитування даних з файлу 3 . Програма досліджує інтерфейс на складність 3.1.Обчислення підсумкового коефіцієнту складності для завдання 3.2. Обчислення коефіцієнтів важливості

	користувачів 3.3. Обчислення підсумкової складності програми 4. Дослідження складності інтерфейсу 5. Рекомендації щодо зменшення складності 6 . Запис файлу аналізу
Розширення	1.А. Програма не може відкрити файл 1.А.1. Повідомлення про помилку

3.5 Проектування структури та організації класів

Розроблена структура програмного продукту описується за допомогою діаграми класів (див. рис. 3.2). Діаграми класів використовуються для моделювання статичного вигляду системи з точки зору проектування. [20]

Діаграма класів може відбивати різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їхню внутрішню структуру і типи відносин. Діаграми класів складають основу ще двох діаграм - компонентів і розгортання.

Програмний продукт для методики аналізу складності інтерфейсів складається з наступних класів:

- Клас Metodics містить головну форму та виклики функцій під час виконання певних подій;
- Клас ClOpenFile відповідає за відкриття та підготовку файлу для заповнення;
- Клас ClComplexity відповідає за аналіз складності;
- Клас ClTask відповідає за аналіз важливості завдання;
- Клас ClFormUsers відповідає за аналіз значимості користувачів.
- Клас ClScreen відповідає за розрахунок кількості екранів та розбиття на класи.

Спроектуємо діаграму організації класів, яка буде описувати структуру проєкту (див. рис. 3.2).

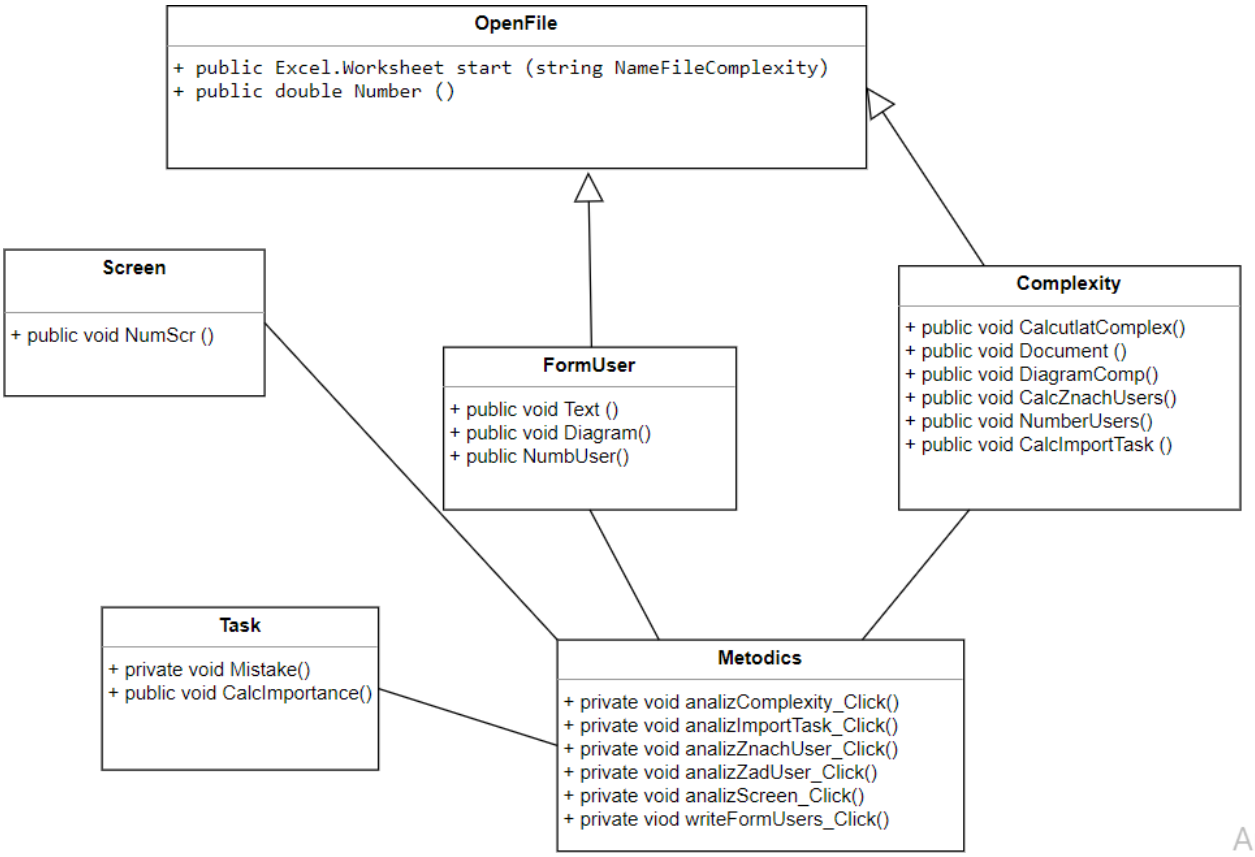

Рисунок 3.2 – Діаграма класів програмного продукту для методики аналізу складності інтерфейсу

Розглянемо докладніше кожен з класів.

Клас Metodics – це клас програмного продукту для методики аналізу складності інтерфейсів, який містить в собі головну форму програмного продукту та виклики функцій інших класів під час виконання певних подій.

Клас ClTask місить в собі функції обробки даних, які були введені на відповідній формі, для аналізу важливості завдання.

Функція Mistake() відповідає за вивід інформації про помилки, у випадку, коли користувач ввів недостатню кількість даних.

Функція CalcImportance() для розрахунку важливості викликається у класі Metodics після натиснення кнопки «Розрахувати» на відповідній формі. Викликана функції залежить від початкових даних, які вводе користувач.

Клас ClOpenFile містить в собі функції відкриття файлу для аналізу складності інтерфейсу. Вхідними даними є файли з необхідною інформацією.

Клас ClComplexity є спадкоємцем класу ClOpenFile містить в собі функції обчислення складності інтерфейсу та запису результатів обчислення у відкритий файл. Вихідними даними є таблиця, яка містить важливість кожного завдання та діаграми важливості кожного завдання.

Клас ClFormUsers є спадкоємцем класу ClOpenFile та містить в собі функції: Text(),NumbUser() та Diagrams().

Функція Text() визначає тип користувача: активний або пассивний.

Функція NumbUser() розподіляє по кожній зі шкал кількість користувачів кожного типу.

Функція Diagrams() будує діаграми розподілу кількості користувачів кожного типу.

Клас ClScreen визначає послідовність екранів для вирішування завдання та містить в собі функцію: NumScr().

3.6 Розробка алгоритму для отримання кількісної оцінки інтерфейсів

Алгоритм — послідовний набір систематизованих правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій. Для візуального зображення алгоритмів часто використовують блок-схеми. [9]

Опис алгоритму отримання кількісної оцінки:

1. Аналіз предметної області та статичних матеріалів.
2. Визначення завдань користувачів (не більше трьох завдань).

2.1 Присвоювання пріоритету завдань.

3. Визначаємо типи користувачів (не більше 10).

3.1. Обчислюємо коефіцієнт важливості користувачів.

4. Складання послідовності екранів для кожного користувацького завдання.
5. Розрахунок кількісної оцінки.

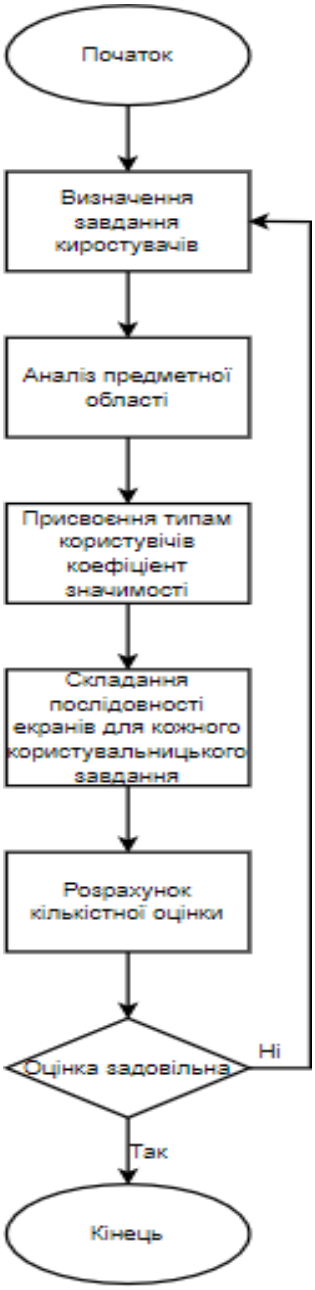

Рисунок 3.3 – Схема алгоритму отримання кількісної оцінки.

3.7 Висновок

У третьому розділі було розглянуто технології та інструментальні засоби, які використані для реалізації програмного продукту. Також розроблено алгоритм для отримання кількісної оцінки юзабіліті програмних систем.

4 КЕРІВНИЦТВО КОРИСТУВАЧА ТА ТЕСТОВИЙ ПРИКЛАД

4.1 Керівництво користувача системи

На початку роботи користувач бачить форму для аналізу складності інтерфейсу (див. рис. 4.1).

Рисунок 4.1 – Форма для аналізу складності інтерфейсу

Якщо користувач натиснув на кнопку «Продовжити» не обравши файл, з’являється надпис про необхідність вибрати файл (див. рис. 4.2).

Рис. 4.2 – Форма для аналізу складності інтерфейсу у тому випадку, коли не вибрали файл

Після завантаження файлу користувачу потрібно заповнити поле з кількістю завдань (див. рис. 4.3).

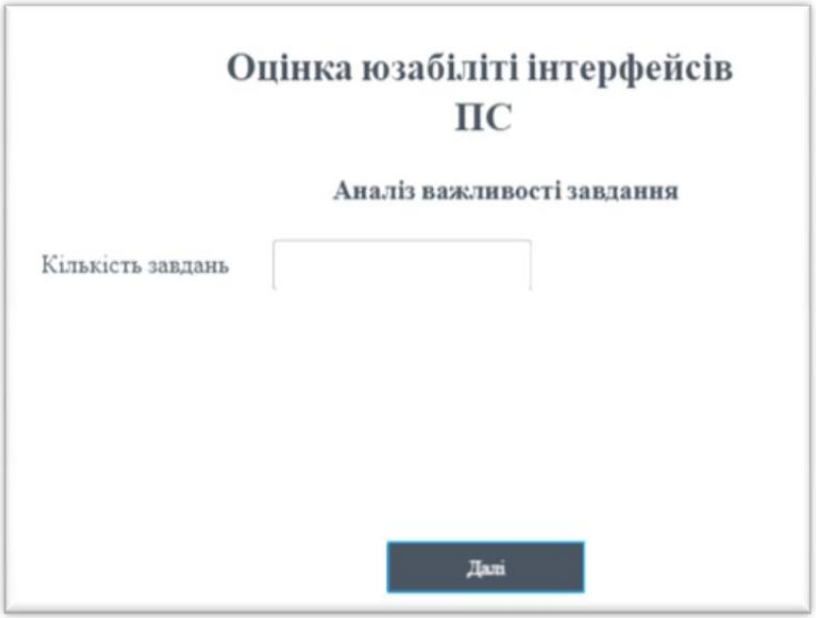

Рисунок 4.3 –Форма для аналізу важливості завдань

Якщо користувач натиснув на кнопку «Далі» залишивши поле для вводу порожнім, то з’являється надпис про необхідність введення даних (див. рис. 4.4).

Рисунок 4.4 – Повідомлення про помилку

Бачимо на рисунку 4.5, що користувач ввів кількість завдань.

Рисунок 4.5 – Форма для аналізу важливості завдань

Після того, як ввели кількість завдань та нажали на кнопку «Далі» перед користувачем з1являється наступна форма (див. рис. 4.6).

Рисунок 4.6 – Форма для аналізу важливості завданнь з просвоєнням коефіцієнтівважливості для кожного завдання

Користувач може обрати завдання та присвоїти їм коефіцієнти.

Рисунок 4.7 – Форма для аналізу важливості завданнь з просвоєнням коефіцієнтівважливості для кожного завдання

Якщо після присвоєння коефіцієнтів їх сума більша, за 100%, то приходить повідомлення про помилку (див. рис 4.8).

Рисунок 4.8 – Повідомлення про помилку

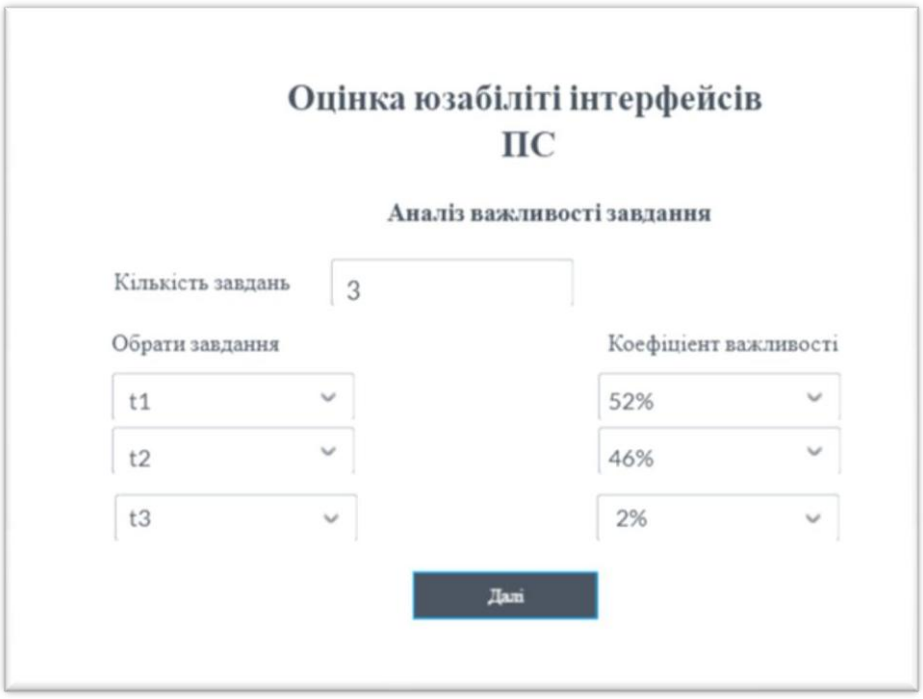

Рисунок 4.9 – Форма для аналізу важливості завданнь з просвоєнням коефіцієнтів важливості, яка дорівнює 100%

Також користувач може ознайомитись з матеріалом для розрахунку складності (див. рис. 4.10).

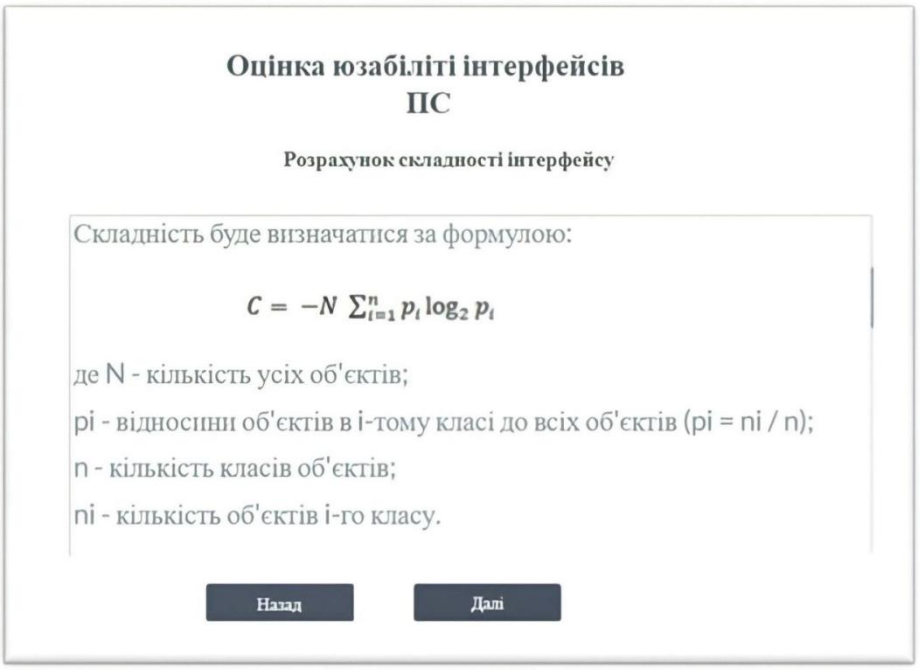

Рисунок 4.10 – Ознайомчий матеріал

4.2 Дослідження розрахунку кількісної оцінки

За розробленою методикою проведено оцінку кількох подібних за призначенням програм. Оцінюване ПЗ генерує QR-коди. QR-код «QR - Quick Response - Швидкий Відгук» - це двомірний штрих-код (бар-код), що надає інформацію для швидкого розпізнавання за допомогою камери на мобільному телефоні. За допомогою QR-коду можна закодувати будь-яку інформацію, наприклад текст, номер телефону, посилання на сайт або візитну картку.

Технічні особливості технології QR поділяють кілька категорій кодів:

— текст,

— адреса URL, або адреса URL, що використовується певним додатком (у тому числі точка на Google Maps, відеоролик з YouTube, повідомлення у Twitter, повідомлення у Facebook, повідомлення у LinkedIn, повідомлення у FourSquare,

посилання на пряму скачування в App Store, дзвінок у Skype, адреса електронної пошти),

— телефон,

— повідомлення, що надсилається на певний номер,

— контакт формату VCard.

Виділяємо 5 основних типів користувачів:

— U1: отримання QR-коду необхідного розміру з деяким текстом,

— U2: отримання QR-коду необхідного розміру зі стандартним посиланням, що відкривається в браузері або в деякій сторонній програмі,

— U3: отримання QR-коду необхідного розміру з номером телефону,

— U4: отримання QR-коду необхідного розміру, що містить повідомлення та телефонний номер

— U5: отримання QR-коду необхідного розміру з контактною інформацією у форматі VCard.

За непрямими даними експерт визначає значущість Ku (U1) = 44%, Ku(U2) = 36%, Ku(U1) = 10%, Ku(U1) = 5%, Ku(U1) = 5%.

Оскільки для користувачів виділено одне завдання, то Kt1(U1)=Kt1(U2) = Kt1(U3) = Kt1(U4) = Kt1(U5) = 100%.

Рисунок 4.11 – Послідовність екранів, необхідна для виконання завдання U1 (ліворуч) та U2 (праворуч)

Виділяємо класи об'єктів для всіх сценаріїв:

n1-посилання верхнього меню,

n2 - поле введення,

n3 - кнопка генерації зображення,

n4-контроль вибору розміру,

n5 - QR-код з можливістю скачування.

Кількість класів об'єктів у послідовності n = 5,

Для сценарію U1:

$$C_{s1}(U_1) = -6\left({}^{4}/_{5}\log_2{}^{4}/_{5} + 2\times{}^{1}/_{5}\log_2{}^{1}/_{5}\right) = 7,$$

$$C_{s2}(U_1) = -6\left({}^{4}/_{5}\log_2{}^{5}/_{5} + 2\times{}^{1}/_{5}\log_2{}^{1}/_{5}\right) = 7,$$

$$C_{s3}(U_1) = -8\left({}^{4}/_{5}\log_2{}^{4}/_{5} + 3\times{}^{1}/_{5}\log_2{}^{1}/_{5}\right) = 13.$$

$$C_{u1} = 7 + 7 + 13 = 27.$$

Для сценарію U2:

$$C_{s1}(U_2) = -6\left({}^{4}/_{5}\log_2{}^{4}/_{5} + 2\times{}^{1}/_{5}\log_2{}^{1}/_{5}\right) = 7,$$

$$C_{s2}(U_2) = -6\left({}^4/_5 \log_2 {}^4/_5 + 2 \times {}^1/_5 \log_2 {}^1/_5\right) = 7,$$

$$C_{s3}(U_2) = -6\left({}^4/_5 \log_2 {}^4/_5 + 2 \times {}^1/_5 \log_2 {}^1/_5\right) = 7,$$

$$C_{s4}(U_2) = -8\left({}^4/_5 \log_2 {}^4/_5 + 2 \times {}^1/_5 \log_2 {}^1/_5\right) = 13.$$

$$C_{u2} = 7 + 7 + 7 + 13 = 34.$$

Рисунок 4.12 – Послідовність екранів, необхідна для виконання завдання U3 (ліворуч) та U4 (праворуч)

Для сценарію U3:

$$C_{s1}(U_3) = -6\left({}^4/_5 \log_2 {}^4/_5 + 2 \times {}^1/_5 \log_2 {}^1/_5\right) = 7,$$

$$C_{s2}(U_3) = -6\left(\frac{4}{5}\log_2\frac{4}{5} + 2\times\frac{1}{5}\log_2\frac{1}{5}\right) = 7,$$

$$C_{s3}(U_3) = -6\left(\frac{4}{5}\log_2\frac{4}{5} + 2\times\frac{1}{5}\log_2\frac{1}{5}\right) = 7,$$

$$C_{s4}(U_3) = -8\left(\frac{4}{5}\log_2\frac{4}{5} + 3\times\frac{1}{5}\log_2\frac{1}{5}\right) = 13.$$

$$C_{u3} = 7 + 7 + 7 + 13 = 34.$$

Для сценарію U4:

$$C_{s1}(U_4) = -6\left(\frac{4}{5}\log_2\frac{4}{5} + 2\times\frac{1}{5}\log_2\frac{1}{5}\right) = 7,$$

$$C_{s2}(U_4) = -7\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{2}{5}\times\log_2\frac{2}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 12,$$

$$C_{s3}(U_4) = -7\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{2}{5}\times\log_2\frac{2}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 12,$$

$$C_{s4}(U_4) = -9\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{2}{5}\times\log_2\frac{2}{5} + 2\times\frac{1}{5}\log_2\frac{1}{5}\right) = 15,$$

$$C_{u4} = 7 + 12 + 12 + 15 = 46.$$

Для сценарію U5:

$$C_{s1}(U_5) = -6\left(\frac{4}{5}\log_2\frac{4}{5} + 2\times\frac{1}{5}\log_2\frac{1}{5}\right) = 7,$$

$$C_{s2}(U_5) = -9\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{4}{5}\times\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 12 = C_{s3} = C_{s4} = C_{s5} = C_{s6} = C_{s7},$$

$$C_{s8}(U_4) = -7\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{2}{5}\times\log_2\frac{2}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) = 12,$$

$$C_{u5} = 7 + 12 * 5 + 17 = 84.$$

Розрахована метрика показує абстрактну складність розв'язуваного користувачем завдання, засновану на різноманітності та кількості елементів, з якими доводиться взаємодіяти користувачеві в процесі вирішення задачі.

$Є_{u1}$= 11,8 ($K_u(U_1)$ = 44%, C_{u1} = 27);

$Є_{u2}$= 12,24 ($K_u(U_2)$ = 36%, C_{u1} = 34);

$Є_{u3}$= 3,4 ($K_u(U_3)$ = 10%, C_{u1} = 34);

$Є_{u4}$= 2,3 ($K_u(U_4)$ = 5%, C_{u1} = 46);

$Є_{u5}$= 4,2 ($K_u(U_5)$ = 5%, C_{u1} = 84);

$$Є = 11{,}8 + 12{,}24 + 3{,}4 + 2{,}3 + 4{,}2 = 33{,}94$$

Для порівняння оцінимо іншу програму, яка виконує схожі завдання.

Друга програма має дещо меншу кількість функцій, тому кількість виділених користувачів так само буде меншою.

Виділяємо 2 основних типи користувачів:

— U1: отримання QR-коду необхідного розміру з деяким текстом,

— U2: отримання QR-коду необхідного розміру зі стандартним посиланням, що відкривається в браузері або в деякій сторонній програмі.

За непрямими даними експерт визначає значущість Ku (U1) = 55%, Ku(U2) = 45%.

Оскільки для користувачів виділено одне завдання, то Ct1(U1)=Ct1(U2) = 100%.

Рисунок 4.13 – Послідовність екранів, необхідна для виконання завдання

U1 та U2

Виділяємо класи об'єктів для всіх сценаріїв:

n1 - поле введення,

n2 - QR-код із можливістю скачування.

Кількість класів об'єктів у послідовності n = 2.

Для сценарію U1:

$$C_{s1}(U_1) = -2\left({}^1\!/_2 \log_2 {}^1\!/_2 + {}^1\!/_2 \log_2 {}^1\!/_2\right) = 2,$$

$$C_{s2}(U_1) = -6\left({}^4\!/_5 \log_2 {}^5\!/_5 + 2 \times {}^1\!/_5 \log_2 {}^1\!/_5\right) = 2,$$

$$C_{u1} = 2 + 2 = 4.$$

Для сценарію U2 послідовність екранів буде така сама, тобто $C_{u1} = C_{u2}$=4.

$Є_{u1}$= 2,2 ($K_u(U_1)$ = 55%, C_{u1} = 4);

$Є_{u2}$= 1,8 ($K_u(U_2)$ = 45%, C_{u1} = 4);

$Є = 2{,}2 + 1{,}8 = 4$

Щоб перевірити результати методики, порівняємо показники програм 1 і 2 за аналогічними користувальницькими сценаріями один з одним і з результатами оцінки KLM-GOMS.

Для першої програми:

$T_1(U_1)$= M + BB + H + K + H + BB = 0,6 + 0,2 + 0,4 + 0,28 + 0,4 + 0,2 = 2,08 с.

$Є_1(U_1)$= 11,8

$T_1(U_2)$= M + BB + M + BB + H + K + H + BB = 0,6 + 0,2 + 0,6 + 0,2 + 0,4 + 0,28 + 0,4 + 0,2 = 2,88 с.

$Є_1(U_2)$= 12,24

T_1 = 4,96 с.

$Є_1(U_1 + U_2)$= 11,8 + 12,24 = 24,04

Для другої програми:

$T_2(U_1)$= $T_2(U_2)$= M + H + K + H = 0,6 + 0,4 + 0,28= 1,28

$Є_1(U_1)$= 1,8

$Є_2(U_2)$= 2,2

T_2 = 2,56 с

$Є_2(U_1 + U_2)$= 1,8 + 2,2 = 4

Вочевидь, що у результаті двох методик ми маємо подібні значення, але при розрахунку розробленої методики оцінки вдалось заощадити час на 3 хвилини 10 секунд.

4.3 Висновок

На даному етапі були перевіренні такі сценарії програми в яких найбільша ймовірність виникнення помилок. Програма пройшла тести успішно. Також за розробленою методикою проведено оцінку кількох подібних за призначенням программ та порівняно вже з існуючим методом оцінки юзабіліті.

ВИСНОВОК

В результаті дипломної роботи було розроблено програмний продукт та методику кількісної оцінки характеристик інтерфейсу користувача програмних систем, яка не вимагає великої кількості витрат для залучення користувачів або експертів і володіє досить низьким рівнем суб'єктивності.

У роботі було розглянуто існуючі на цей час методи оцінки характеристик інтерфейсу, розроблено методику оцінки характеристик інтерфейсу та програмний засіб для оцінки характеристик інтерфейсу.

Також у роботі було проведено дослідження розробленої кількісної оцінки, яке показало, що дана програмна система дозволяє заощадити час на оцінку характеристик інтерфейсу користувача програмних систем на 3 хвилини 10 секунд.

Програма для методики кількісної оцінки юзабіліті інтерфейсів ПС розроблена на мові програмування C#.

Робота почалася зі специфікації вимог до програмної системи. З самого початку був проведений аналіз предметної області, визначені проблеми користувачів та пошук і вивчення існуючих аналогів. Далі було визначено основний функціонал системи у вигляді діаграми варіантів використання, а потім були визначені не функціональні вимоги до майбутньої системи.

Наступним кроком стало проектування системи, під час якого були визначені програмні класи. У розділі описано, як саме відбувалася реалізація програмної системи, які технології були обрані, як реалізовані. Останнім етапом розробки програмної системи стало її тестування та оцінка подібних програм з порівнянням оцінки вже існуючої методики.

СПИСОК ЛІТЕРАТУРИ

1. Тео Мандел. Розробка користувальницького інтерфейса: Пер. с анг. – М.: ДМК Пресс, 2001 р..

2. Якоб Нільсен, Хоа Лоранжер. Web-дизайн: зручність використання веб-сайтів (юзабіліті): Пер. с анг. – СПб.: Видавництво «Віл'ямс», 2007р..

3. https://cyberpedia.su/5x6966.html
4. https://webtune.com.ua/statti/web-rozrobka/yuzabiliti-sajtu/
5. https://sprava.ua/blog/pochemu-juzabiliti
6. https://aboutmarketing.info/internet-marketynh/instrumenty/yuzabiliti-saytu-10-pryntsypiv-optymizatsiyi/

7. Крісілов В.А., Оніщенко Т. В., Писаренко К. О., Людино-Машинна взаємодія: Навчальний посібник для студентів / В. А. Крісілов, Т. В. Оніщенко, К.О. Писаренко — ОНУ, 2019 р..

8. Ватутін В.А. Теорія ймовірностей та математична статистика у завданнях: Навчальний посібник для вузів/ В. А. Ватутін, Г. І. Івченко, Ю. І. Мєдвєдєв,В. П. Чистяков— 2-е видавництво., випр. – М.: Дрофа, 2003.– 328 с.

9. Гмурман В. Е. Теорія ймовірностей та математична статистика/ В. Е. Гмурман - М., 2003.- 479 с.

10. Єрмолаєва С. С. Технологія проектування якості організації навчання у вузі/ С. С. Єрмолаєва - Молодий вчений №1, 2011 — С. 199-204

11. Герберт Шилдт. C# 4.0. Повний посібник /Герберт Шилдт - М.: Віл'ямс, 2011 – 1056 с.

ДОДАТОК А

Class Metodics

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;

namespace Func
{
    public partial class Metodics : Form
    {
        public Metodics()
        {
            InitializeComponent();
        }

        public Excel.Workbooks excelappworkbooks;
        public Excel.Workbook excelappworkbook;

        public Excel.Application excelapp;
        public Excel.Window excelWindow;

        public Excel.Sheets excelsheets;
        public Excel.Worksheet excelworksheet;

        public Excel.Range excelcells;
        public string ForFileName;

        private void writeFileTast_Click(object sender, EventArgs e)
        {
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
                writeFileTest.Text = openFileDialog1.FileName;
            ForFileName = writeFileTest.Text;
        }
        private void writeFileGroup_Click(object sender, EventArgs e)
        {
            if (openFileDialog2.ShowDialog() == DialogResult.OK)
                writeFileGroup.Text = openFileDialog2.FileName;
            ForFileNameValidG = writeFileGroup.Text;
        }

        private void analizComplexity_Click(object sender, EventArgs e)
        {
            if (writeFileTest.Text == "" || writeFileTest.Text == "Выберите файл
")
            {
                NameFile.Text = "Вы не выбрали файл.";
                NameFile.Location = new Point(97, 118);

                NameFile.Visible = true;
                analizComplexity.Location = new Point(160, 147);
           }
            else
            {
```

```
                ClComplexity c = new ClComplexity();
                Excel.Worksheet excelworksheet = c.start(ForFileName);
                double lLastRow = c.Number();
                c.Design(excelworksheet);
                c.NameDisc("B",lLastRow, "Названия дисциплин");
                c.SrednBal("E", lLastRow + 1, "Средний бал", lLastRow);
                c.NumberGuas(lLastRow + 2, "Количество неправильных ответов",
lLastRow);
                c.Complexity(lLastRow + 3, "Сложность каждого вопроса", lLastRow);
                c.StandardDeviation(lLastRow + 4, "Стандартное отклонение",
lLastRow);
                c.NumberStudents(lLastRow + 7, "Количество студентов", lLastRow);
                c.MinMark(lLastRow + 8, "Минимальный бал за тест", lLastRow);
                c.MaxMark(lLastRow + 9, "Максимальный бал за тест", lLastRow);
                c.NumberMark(lLastRow + 11, "Количество оценок", lLastRow);
                c.DiagramComplexity(lLastRow);
                c.DiagramCorrelationCoefficient(lLastRow);
                c.DiagramPlotn(lLastRow);
            }
        }
        private void analizGraf_Click(object sender, EventArgs e)
        {
            if (writeFileGroup.Text == "" || writeFileGroup.Text == "Введите путь
файла с рейтингом студентов")
            {
                NameFile2.Text = "Введите путь к файлу.";
                NameFile2.Visible = true;
            }
            else
            {
                ClFormGroup c2 = new ClFormGroup();
                Excel.Worksheet excelworksheet = c2.start(ForFileNameValidG);
                double lLastRowV = c2.Number();
                c2.Text(excelworksheet);
                c2.NumbMark(lLastRowV, excelworksheet);
                c2.Diagrams(lLastRowV, excelworksheet);
            }
        }
        private void analizValid_Click(object sender, EventArgs e)
        {
            if (writeFileValid.Text == "" || writeFileValid.Text == "Введите путь
файла с рейтингом выбранных студентов")
            {
                NameFile3.Text = "Введите путь к файлу.";
                NameFile3.Visible = true;
            }
            else
            {
                ClValid c3 = new ClValid();
                Excel.Worksheet excelworksheet = c3.start(ForFileNameValid);
                double lLastRowVG = c3.Number();
                c3.SrBal30balsh("Средний бал (по 30 бальной шкале)", lLastRowVG,
excelworksheet);
                c3.CalculationValidity("Валидость теста", lLastRowVG,
excelworksheet);
                c3.Interf(lLastRowVG, excelworksheet);
                c3.DiagramValid(lLastRowVG, excelworksheet);
            }
        }

        private void Helps(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Help.ShowHelp(this, Application.StartupPath + @"\help.chm",
```

```
HelpNavigator.TopicId);
        }

         private void comboVibPrior_SelectedIndexChanged(object sender, EventArgs
e)
        {
           ClRepres cr = new ClRepres ();
            cr.tableFunc();
        }
         private void comboVibWork_SelectedIndexChanged(object sender, EventArgs e)
         {
             tableFunc();
             if (comboVibWork.SelectedIndex == 0 || comboVibWork.SelectedIndex ==
1)
             {
                 comboVibWork.ForeColor = Color.Black;
                 KolVop.Font = new Font("Microsoft Sans Serif", 8,
FontStyle.Regular);
             }
         }
         private void analizRepresentat_Click(object sender, EventArgs e)
        {
           ClRepres cr2 = new ClRepres ();
            cr2.Mistake();
        }
    }
}
```

Class OpenFile

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;
namespace Func
{
    class ClOpenFile
    {
        public Excel.Workbooks excelappworkbooks;
        public Excel.Workbook excelappworkbook;

        public Excel.Application excelapp;
        public Excel.Window excelWindow;

        public Excel.Sheets excelsheets;
        public Excel.Worksheet excelworksheet;

        public Excel.Range excelcells;

        public Excel.Worksheet start(string NameFileValidG)
        {
           excelapp = new Excel.Application();
           excelapp.Visible = true;
           //Получаем набор ссылок на объекты Workbook
           excelappworkbooks = excelapp.Workbooks;
           //Открываем книгу и получаем на нее ссылку
           excelappworkbook = excelapp.Workbooks.Open(@NameFileValidG,
```

```
                                        Type.Missing, Type.Missing, Type.Missing,
                    "WWWWW", "WWWWW", Type.Missing, Type.Missing, Type.Missing,
                     Type.Missing, Type.Missing, Type.Missing, Type.Missing,
                     Type.Missing, Type.Missing);
            //Получаем массив ссылок на листы выбранной книги
            excelsheets = excelappworkbook.Worksheets;
            //Получаем ссылку на лист 1
            excelworksheet = (Excel.Worksheet)excelsheets.get_Item(1);
            return excelworksheet;
          }
          public double Number()
          {
             double lLastRowV = excelworksheet.UsedRange.Row +
excelworksheet.UsedRange.Rows.Count; // определение номера пустой строки
             return lLastRowV;
          }
    }
}
```

Class Task

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace Func
{
    class Task
    {
private void Mistake()
       {
           if (comboVibWork.SelectedIndex == -1)
           {
               KolVop.Text = "Введите количесвто \nзаданий.";
               KolVop.ForeColor = Color.Red;
               KolVop.Font = new Font("Microsoft Sans Serif", 8,
FontStyle.Italic);
           }

           if (comboVibPrior.SelectedIndex == -1)
           {
               comboVibPrior.Text = "Выберите приоритет.";
               comboVibPrior.ForeColor = Color.Red;
               comboVibPrior.Font = new Font("Microsoft Sans Serif", 8,
FontStyle.Italic);
           }
           if (KolVop.Text == "")
           {
           else
           {
               if (comboVibWork.SelectedIndex == 0)
               {
                   dataGridView1.Columns[4].Visible = true;
                   dataGridView1.Size = new Size(353, 155);//101
                   dataGridView1.Location = new Point(27, 113);
               }
           }
       }
        private void tableFunc()
       {
           if (comboVibWork.SelectedIndex == 0)
```

```
            {
                dataGridView1.Size = new Size(250, 155);//101
                dataGridView1.Location = new Point(89, 113);
                comboVibPrior.ForeColor = Color.Black;
                comboVibPrior.Font = new Font("Microsoft Sans Serif", 8,
FontStyle.Regular);
                if (comboVibPrior.SelectedIndex == -1)
                {
                    comboBox1.Text = "Выберите приоритет.";
                    comboBox1.ForeColor = Color.Red;
                    comboBox1.Font = new Font("Microsoft Sans Serif", 8,
FontStyle.Italic);
                }
                else if (comboVibPrior.SelectedItem.Equals("по важности темы"))
                {
                    dataGridView1.Visible = true;
                    dataGridView1.Columns[1].Visible = false;
                    dataGridView1.Columns[2].Visible = false;
                    dataGridView1.Columns[3].Visible = true;
                }

                tabControl1.Size = new Size(440, dataGridView1.Location.Y +
dataGridView1.Height + 135);
            }
            else if (comboVibWork.SelectedIndex == 1 && comboVibPrior.SelectedIndex
!= -1)
            {
                dataGridView1.Visible = true;
                dataGridView1.Size = new Size(248, 153);
                dataGridView1.Location = new Point(89, 113);

                dataGridView1.Columns[1].Visible = false;
                dataGridView1.Columns[2].Visible = false;
                dataGridView1.Columns[3].Visible = false;
                dataGridView1.Columns[4].Visible = false;
                dataGridView1.Columns[5].Visible = true;
            }
            ProvRep.Location = new Point(18, dataGridView1.Location.Y +
dataGridView1.Height + 15);
            ProvRep.Visible = true;
            ProvRep.Text = "Интерфейс является удобным в использовании\n";
            tabControl1.Size = new Size(440, dataGridView1.Location.Y +
dataGridView1.Height + 150);
        }
    }
}
```

Class ClFormGroup

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Excel = Microsoft.Office.Interop.Excel;

      namespace Func
{
    class ClFormGroup : ClOpenFile
    {
        public Excel.Workbooks excelappworkbooks;
        public Excel.Workbook excelappworkbook;

        public Excel.Application excelapp;
```

```
        public Excel.Window excelWindow;

        public Excel.Sheets excelsheets;
        public Excel.Worksheet excelworksheet;

        public Excel.Range excelcells;

        public void Text(Excel.Worksheet excelworksheet)
        {
            double lLastRowV = excelworksheet.UsedRange.Row +
excelworksheet.UsedRange.Rows.Count; // определение номера пустой строки
            //текст
        public void NumbMark(double lLastRow, Excel.Worksheet excelworksheet)
        {

        public void Diagrams(double lLastRow, Excel.Worksheet excelworksheet)
        {
            double L = 1000;
            for (double Number = 3;Number < lLastRow; Number = Number + 1 )
            {
                Excel.ChartObjects chartsobjrcts
=(Excel.ChartObjects)excelworksheet.ChartObjects(Type.Missing);
                Excel.ChartObject chartsobjrct = chartsobjrcts.Add(5, L, 430,
210);
                chartsobjrct.Chart.ChartWizard(excelworksheet.get_Range("BA" +
Number + ":BM" + Number), Excel.XlChartType.xlColumnClustered, 2,
Excel.XlRowCol.xlRows, Type.Missing, 0, true, excelworksheet.get_Range("C" +
Number), "Оценки", "Количество оценок", Type.Missing);

                L = L + 250;
            };
        }

    }
}
```

Class ClComplexity

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Excel = Microsoft.Office.Interop.Excel;

namespace Func
{
    class ClComplexity : ClOpenFile
    {

        public Excel.Workbooks excelappworkbooks;
        public Excel.Workbook excelappworkbook;

        public Excel.Application excelapp;
        public Excel.Window excelWindow;

        public Excel.Sheets excelsheets;
        public Excel.Worksheet excelworksheet;

        public Excel.Range excelcells;

        public void Design(Excel.Worksheet take)
        {
            excelworksheet = take;
```

```
            //Форматируем ячейки

            //Задаем ширину ячеек
            excelworksheet.get_Range("A1", "D1").ColumnWidth = 20;
            excelworksheet.get_Range("E1").ColumnWidth = 10;
            excelworksheet.get_Range("F1", "AE1").ColumnWidth = 5;

            //Задаем выравнивание по центру
            //excelworksheet.get_Range("A1", "AI1").HorizontalAlignment =
Excel.Constants.xlCenter;
            excelworksheet.get_Range("A1", "AI1").VerticalAlignment =
Excel.Constants.xlCenter;

            excelworksheet.get_Range("A1", "AI1").EntireRow.Font.Bold = true;
            excelworksheet.get_Range("A1", "AI1").EntireRow.Font.Shadow = true;
            excelworksheet.get_Range("A1", "AI1").EntireRow.Name = "Arial";
            excelworksheet.get_Range("A1", "AI1").EntireRow.Font.Size = 10;
        }

        public void NameTask(string letter, double newlLastRow, string l)
        {
            excelworksheet.get_Range(letter + (newlLastRow)).Value2 = l;
        }
       public void CoefZnach (string letter, double newlLastRow, string l, double
lLastRow)
        {
           excelworksheet.get_Range("B" + (newlLastRow)).Value2 = l;
            excelworksheet.get_Range(letter + (newlLastRow)).FormulaLocal = "=
ОКРУГЛ(СРЗНАЧ(E2:E" + (lLastRow - 1) + ");2)";
           excelworksheet.get_Range(letter +
(newlLastRow)).AutoFill(excelworksheet.get_Range(letter + (newlLastRow) + ":AI" +
(newlLastRow)), Excel.XlAutoFillType.xlFillMonths);
        }
         public void Complexity(double newlLastRow, string l, double lLastRow)
         {
             excelworksheet.get_Range("B" + (newlLastRow)).Value2 = l;
             excelworksheet.get_Range("F" + (newlLastRow)).FormulaLocal =
"=ОКРУГЛ(F" + (lLastRow + 2) + "/" + (lLastRow - 2) + ";2)";
             excelworksheet.get_Range("F" +
(newlLastRow)).AutoFill(excelworksheet.get_Range("F" + (newlLastRow) + ":AI" +
(newlLastRow)), Excel.XlAutoFillType.xlFillMonths);
         }
         public void StandardDeviation(double newlLastRow, string l, double
lLastRow)
         {
             excelworksheet.get_Range("B" + (newlLastRow)).Value2 = l;
             excelworksheet.get_Range("F" + (newlLastRow)).FormulaLocal =
"=ОКРУГЛ(СТАНДОТКЛОН(F2:F" + (lLastRow - 1) + ");2)";
             excelworksheet.get_Range("F" +
(newlLastRow)).AutoFill(excelworksheet.get_Range("F" + (newlLastRow) + ":AI" +
(newlLastRow)), Excel.XlAutoFillType.xlFillMonths);
         }
         public void NameCorrelationCoefficient(double newlLastRow, string l)
         {
             excelworksheet.get_Range("B" + (newlLastRow)).Value2 = l;
         }
         public void CorrelationCoefficient(double newlLastRow, double lLastRow,
string word)
         {
            excelworksheet.get_Range(word + (newlLastRow)).FormulaLocal =
"=PEARSON(" + word + "2:" + word + (lLastRow - 1) + ";E2:E" + (lLastRow - 1) +
")";
```

```
            // excelworksheet.get_Range("F" + (newlLastRow)).FormulaLocal =
"=PEARSON("+ word +"2:" + word+ (lLastRow - 1) + ";E2:E" + (lLastRow - 1) + ")";
            //нельзя этим// отдельно excelworksheet.get_Range("F" +
(newlLastRow)).AutoFill(excelworksheet.get_Range("F" + (newlLastRow) + ":AI" +
(newlLastRow)), Excel.XlAutoFillType.xlFillMonths);
        }



        public void DiagramComplexity(double lLastRow)
        {
            Excel.ChartObjects chartsobjrcts =
(Excel.ChartObjects)excelworksheet.ChartObjects(Type.Missing);
            Excel.ChartObject chartsobjrct = chartsobjrcts.Add(5, 800, 430, 210);
            chartsobjrct.Chart.ChartWizard(excelworksheet.get_Range("F" +
(lLastRow + 3) + ":AI" + (lLastRow + 3)), Excel.XlChartType.xlColumnClustered, 2,
Excel.XlRowCol.xlRows, Type.Missing, 0, true, "Сложность каждого вопроса", "Номер
вопроса", "Сложность", Type.Missing);
        }
       public void DiagramCorrelationCoefficient(double lLastRow)
        {
            //Определяем диаграммы как объекты Excel.ChartObjects
            Excel.ChartObjects chartsobjrcts =
(Excel.ChartObjects)excelworksheet.ChartObjects(Type.Missing);
            //Добавляем одну диаграмму  в Excel.ChartObjects - диаграмма пока
            //не выбрана, но место для нее выделено в методе Add
            Excel.ChartObject chartsobjrct = chartsobjrcts.Add(500, 800, 430,
210);
            excelcells = excelworksheet.get_Range("F" + (lLastRow + 5) + ":AI" +
(lLastRow + 5));
            //Получаем ссылку на созданную диаграмму
            Excel.Chart excelchart=chartsobjrct.Chart;
            //Устанавливаем источник данных для диаграммы
            excelchart.SetSourceData(excelcells,Type.Missing);

            //Тип диаграммы:
            excelchart.ChartType = Excel.XlChartType.xlColumnClustered;
            //Создаем надпись - Заглавие диаграммы
            excelchart.HasTitle=true;
            excelchart.ChartTitle.Text = "Коэфициент корреляции";
            excelchart.ChartTitle.Font.Size = 14;


((Excel.Axis)(excelchart.Axes(Excel.XlAxisType.xlCategory,Excel.XlAxisGroup.xlPrim
ary))).HasTitle =true;

((Excel.Axis)excelchart.Axes(Excel.XlAxisType.xlCategory,Excel.XlAxisGroup.xlPrima
ry)).HasTitle =true;
```