Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія прогнозування трансформації автомобільних фарб»

Виконав: студент 2
курсу, групи 1КН-19м,
спеціальності 122 «Комп'ютерні науки»
Сілагін Є.О.

Керівник: д.т.н., проф. Перевозніков С.І.

Рецензент: канд. техн. наук, доцент каф. ПЗ Романюк О.В.

Вінниця
2020

АНОТАЦІЯ

Робота присвячена розробці інформаційної технології прогнозування трансформації автомобільних фарб. Розглянуто можливості застосування для вирішення подібної задачі однієї з сучасних інтелектуальних методологій. Обгрунтовано застосування апарату нечітких множин із використанням нечітких баз знань и нечіткого логічного виводу.

Для розробки програмного модуля використане програмне середовище IntelliJ IDEA та мова програмування Java.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Людство здавна застосовує поняття «кольорового відтінку» - певного співвідношення значень кольорової моделі яке одержало свою специфічну назву. Ось приклади таких назв: «кремовий», «морська хвиля», «сріблястий металік», «маренго», «асфальтовий», «хакі» і т.д. В силу використання різних кольорових моделей, різних програм, пристроїв (моніторів, принтерів, сканерів) для кожного із таких кольорових відтінків не існує певного стандартного кількісного значення. Це стосується навіть таких галузей як лакофарбна або ткацька промисловість, виробництво будівельних та оздоблювальних матеріалів. Навіть виробники змішаних, «понтонних» фарб для поліграфії не мають єдиного стандарту. Більше того сам перелік модних кольорових відтінків постійно змінюється. Спеціалісти з психології навіть вказують на певну потребу людської психіки в «свіжих» кольорових відтінках. Так періодично «модними» стають кольорові відтінки, які раніше не використовувались.

В процесі експлуатації автомобіля виникає потреба відновити лако-фарбне покриття деяких кузовних деталей, або їх частин, наприклад, після аварії, або в результаті модернізації (тюнінгу). При спробі використати для цього стандартну (за кодом або назвою кольорового відтінку) фарбу ми одержимо бажаний результат тільки для абсолютно нових автомобілів. В процесі експлуатації, під дією сонця, температури, бруду, кольоровий відтінок фарби суттєво змінюється – фарба «старіє». Цей процес залежить від багатьох чинників: часу експлуатації, кліматичної зони, гаражного чи атмосферного зберігання, застосування миючих засобів і т.д. Проблему вимушені вирішувати підбором «ремонтної» фарби із CMYK складових. Процес підбору довготривалий і затратний, так як потребує спеціального обладнання і кількох тестових проб, доведених до повного висихання, а результат повністю залежить від досвідченості маляра.

Зв'язок роботи з науковими програмами, планами, темами.

Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Розробка спеціалізованих засобів штучного інтелекту на основі інтелектуального аналізу даних та машинного навчання». та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою даної роботи є зменшення трудовитрат в процесі прогнозування трансформації автомобільних фарб. Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати існуючі технології, методи і моделі прогнозування трансформації автомобільних фарб;

– сформулювати вимоги до роботи технології прогнозування трансформації автомобільних фарб.

– розробити математичну модель і технологію прогнозування трансформації автомобільних фарб на основі нечіткої логіки.

– спроектувати програмний додаток для прогнозування трансформації автомобільних фарб;

– реалізувати програмний додаток для прогнозування трансформації автомобільних фарб;

– виконати завдання економічної частини;

Об'єкт дослідження – процес прогнозування трансформації автомобільних фарб.

Предмет дослідження – технології, алгоритми та програмні засоби для прогнозування трансформації автомобільних фарб.

Область застосування – автоматизоване прогнозування трансформації автомобільної фарби та її підбір в залежності від умов експлуатації.

Наукова новизна одержаних результатів полягає в наступному: удосконалена інформаційна технологія прогнозування трансформації автомобільних фарб, що дозволяє отримати результат прогнозування з меншими трудовитратами. Зменшення трдовитрат досягається за рахунок

використання нечіткої бази експертних знань та продукційного логічного механізму виведення результату. Вперше формалізована і фазифікована задача прогнозування трансформації автомобільних фарб в результаті експлуатації.

Практичне значення одержаних результатів полягає у тому, що розроблена технологія, алгоритми і програмний додаток зможуть бути використані при створенні експертних систем підтримки прийняття рішень в колористиці.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати досліджень апробовані на XLIX науково-технічній конференції підрозділів ВНТУ, XII міжнародній науково-практичній конференції «ІОН-2020».

Публікації. За результатами опубліковано тези доповіді регіональної науково-технічної конференції «ФІТКІ ВНТУ-2019» [1] та міжнародної науково-практичної конференції «ІОН-2020» [2].

Результати, одержані в процесі виконання магістерської кваліфікаційної роботи, плануються до впровадження в розробки науково-виробничого підприємства ТОВ «ІТІ».

1 ОГЛЯД ТА АНАЛІЗ ЗАСОБІВ І ГОТОВИХ РІШЕНЬ ПРОГНОЗУВАННЯ ТРАНСФОРМАЦІЇ АВТОМОБІЛЬНИХ ФАРБ

1.1 Огляд відомих рішень у проблемі прогнозування трансформації автомобільних фарб

Задача ідентифікації [3] є однією із основних задач штучного інтелекту. Цей напрямок має множину практичних застосувань: медична діагностика, автоматичне розпізнавання рукописного тексту, біометрична ідентифікація людини по фото і відбиткам пальців, інтелектуальний пошук у базах даних зображень та інші. Одним із важливих напрямів використання задач розпізнавання є автоматичні чи автоматизовані системи розпізнавання (ідентифікації) кольорових відтінків. Відповідно, успішне вирішення цієї задачі забезпечує ефективне функціонування систем автоматизованого пошуку, дизайну, стискування інформації та інших.

Отже, проведемо аналіз готових рішень у проблемі автоматизованого підбору та ідентифікації фарб.

Колориметр. У житті так багато важливих і цікавих речей! Назви кольорів - не з їх числа. Запам'ятовувати відмінності якогось екрю від верблюжого потрібно лише деяким людям вкрай специфічних професій. Всім іншим достатньо мати під рукою Колориметр. Ця програма для iPhone, яке визначає колір по фотографії, надаючи вам максимум інформації по кожному відтінку. Пам'ятайте, як шукали шпалери певного кольору? Або предмет одягу під специфічний дресскод? А може, вам потрібна деталь інтер'єру, яка точно вписалася б в кольори меблів або розбавила акценти? Вирішити всі ці та інші дрібні побутові завдання допоможе нове вітчизняне додаток Колориметр для iPhone.

Колориметр діє просто. Запускаєте програму, робите фото і водите по ньому пальцем. Віртуальне збільшувальне скло аналізує колір в точці дотику і видає його культурне назву.

Побігавши по дому з айфоном і дізнавшись багато нових слів, починаєш усвідомлювати користь від практичного застосування програми. Зімітувати звук можна. Описати на слова форму предметах теж нескладно. А ось пояснити, чим відрізняється подобаються вам відтінок сірого від того, який уявляє собі співрозмовник, майже неможливо. І ось тут-то і стане в нагоді база кольорів колориметра.

Наприклад, вам потрібно купити шпалери певного кольору. Або ще гірше: хтось інший повинен купити шпалери, а ви сидите в передчутті, що ввечері вам додому привезуть десяток рулонів зовсім не того відтінку, про який ви мріяли. Сфотографувавши предмет і вибравши найближчий до нього відтінок, ви будете впевнені, що ваш партнер або помічник по ремонту точно знає, що потрібно купити.

Іноді навіть найменше відхилення в кольорі може стати критичним. Наприклад, при виборі фарби. Тут вже недостатньо одного назви або зображення. Потрібні стандарти, цифри. Добре, що в колориметрії все це теж є. Натиснувши на назву кольору, ви відкриєте його докладний опис. Тут вказані параметри відтінку в найпопулярніших колірних моделях - RGB (цифрове зображення) і CMYK, для друку. (рис. 1.1)

Рисунок 1.1 – Параметри відтінку в різних моделях

Якщо ж мова йде не про рідкі фарби і кольори, а про віртуальні, дизайнери оцінять і два додаткові параметри - HSB (тон, насиченість і яскравість) і HEX-код кольору для веб-сайтів. Крім цього, додаток показує три найбільш підходящі кольори з використанням системи Pantone, знаючи які, вам буде набагато простіше вести діалог з виробником або продавцем меблів і декору.

APDS-9960 - датчик, розроблений компаніэю Avago. Це комбінований цифровий датчик з цілим рядом різних цікавих і корисних функцій. Він вміє розпізнавати жести, визначати наближення, а ще він вміє реєструвати інтенсивність навколишнього освітлення і визначати колір [4].

Розпізнавання кольору і рівень зовнішньої освітленості (Color / ALS)

Згідно функціональної схемою, датчик визначає колір / рівень освітленості за допомогою відповідних фотодіодів. Також заявлено, що APDS-9960 має вбудовані фільтри, що блокують ультрафіолетове і інфрачервоне діапазони.

Спрощено це виглядає так: зареєстровані фотодіодами сигнали вимірюються за допомогою аналогово-цифрового перетворювача, заносяться в буфер і потім дані відправляються на обробку.

Графіки (рис. 1.2) взяті з документації на датчик, зліва зверху представлена спектральна характеристика Color Sense (RGBC).

Сигнал RGBC фотодіодів накопичується протягом періоду часу, встановленого значенням регістра ATIME. Значення цього регістру визначчається константою DEFAULT_ATIME. Коефіцієнт підсилення регулюється в діапазоні від 1x до 64x і є автоматично настроюваним параметром CONTROL AGAIN. Наприклад, значення константи DEFAULT_AGAIN рівне 1, відповідає посиленню в 4 рази.

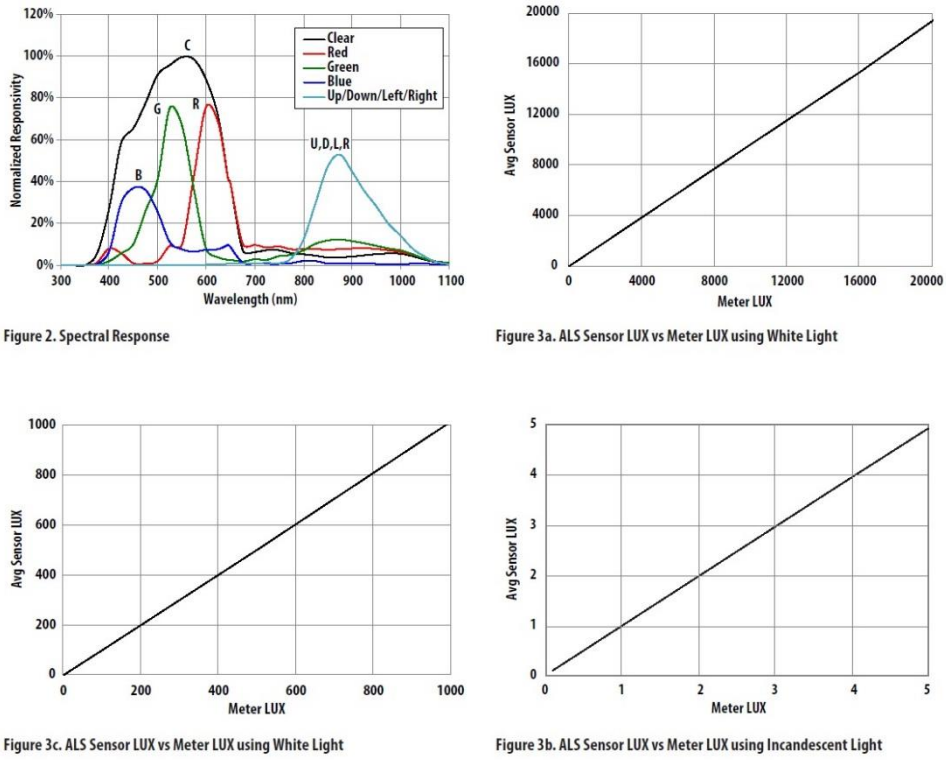


Рисунок 1.2 – Графіки та сенсори

Підбір фарби для авто з використанням стандартної комп’ютерної програми автосалону.

Комп’ютерний підбір фарби для авто – комплекс дій, що виконується з метою отримання повного збігу відтінку використовуваної фарби з поточним кольором машини. Підбір фарби починається з введення в програму VIN-коду – 17-значного номера, в якому вказана інформація про заводське забарвлення автомобіля.

Програма відшукує VIN-код в базі даних кольору машин і надає точне співвідношення кольору фарби і пропорції CMYK, які потрібно змішати щоб отримати відтінок, ідентичний тому, що використовувався виробником для фарбування вашого транспортного засобу.

Однак навіть при перевірці VIN-коду в програмі не вдається отримати відповідність реального кольору автомобілів і запропонованого комп’ютером варіанти. Причин тому кілька:

1. Відмінності в якості матеріалів – тональність фарб з ідентичним кодом від різних виробників відрізняється. Використання складових фарб, якими забарвлений саме ваш автомобіль, не представлених на вітчизняному ринку, тягне за собою додаткові фінансові витрати, пов'язані з необхідністю закупівлі матеріалу в іншій країні;

2. Автомобільні фарби змінюють свій колір під впливом атмосферних факторів в процесі експлуатації машини. Навіть відтінок, обраний в ідеальному відповідно до заводського забарвленням, не буде відповідати реальному кольором авто.

За допомогою комп'ютерного підбору фарби [5] вдасться домогтися збігу відтінку використовуваного матеріалу з поточним кольором машини. Підбір фарби на автомобіль після використання комп'ютерної програми тільки починається – в роботу вступають колористи, які доводять початковий колір складу додаванням в нього пігментів і змішуванням різних відтінків для отримання необхідної палітри. Як відбувається вибір фарби для автомобіля? Підбір кольору супроводжується комбінуванням великої кількості компонентів – в хід йде до 15-ти різних фарб, змішуваних в строгих пропорціях, вага яких вираховується на найточніших електронних вагах з похибкою менш 0,1 грама. Отримана фарба для машини зіставляється з оригінальною за допомогою тест-пластин, пофарбованих приготованим складом, відтінок яких при спеціальному освітленні порівнюється з кузовом машини.

Щоб вибрати колір фарби колорист витрачає велику кількість часу, і якість підсумкового результату повністю залежить від його досвіду, оскільки всі операції виконуються «на око». Додатковими проблемами супроводжується підбір автоемалі перламутрового відтінку і кольору «металік».

При неможливості отримання ідеального збігу реалізується метод «плавного переходу», при якому основна поверхню забарвлюється отриманим складом, а місця з'єднання оброблюваної ділянки фарбуються сумішшю з

приготованого матеріалу і складу, рецепт якого спочатку вказала комп'ютерна програма.

В сучасних автосервісах кольоровка фарби здійснюється механізованим способом за допомогою спеціального обладнання, яке дозволяє в мінімальні терміни підібрати матеріал з відтінком, повністю відповідним фактичного окрасу автомобіля.

У перелік обладнання, що використовується професійним колористами, входить:

1. Освітлювальний стенд, укомплектований світлодіодними лампами, теплота і інтенсивність випромінювання яких імітує природне сонячне світло;

2. Спектрограф – прилад для аналізу лакофарбового покриття автомобіля, який видає інформацію про співвідношення різних пігментів в нанесеною на кузов фарбі.

Спектрограф аналізує спектральну криву кольору, після чого вона завантажується в комп'ютерну програму і система видає рецепт складу, відтінок якого відповідає змінам кольору автомобіля в процесі експлуатації.

Для отримання необхідної комбінації тонів програма зіставляє фактичну спектральну криву і криві відтінків, занесених в її базу даних, і пропонує на вибір колористу варіанти з найменшим відхилення від еталону. Завдання колориста в даному випадку – вибрати ідеальний склад, що він здійснює за допомогою фарбування тестових пластинок і їх візуального порівняння з відтінком машини.

Обрана комп'ютерним способом фарба для автомобілів, за рахунок якої складності процесу підбору – недешеве задоволення. При ремонті транспортного засобу в серйозному автосервісі фахівці занесуть рецепт використовуваного для фарбування складу в свою базу даних, що дозволимо їм при повторному зверненні відразу приступати до фарбування авто, а не витрачати час на підбір матеріалу, а вам – економити гроші на послугах колориста.

Отже, тепер порівняємо ці 3 аналоги з тим, що очікується отримати від нашої розробки (табл. 1.1)

Таблиця 1.1 – Порівняння аналогів з розробкою.

Варіанти	Функціональність	Точність	Обслуговування	Вартість
Аналог №1 Колориметр	Середня	Низька	Мінімальне	Низька
Аналог №2 APDS-9960	Середня	Середня	Мінімальне	Висока
Аналог №3 Стандартний підбір автосалону	Висока	Висока	Максимальне	Висока
Розробка	Висока	Висока	Мінімальне	Середня

Отже, можна зробити висновок, що майбутній програмний модуль матиме найкраще співвідношення ціна-якість(продуктивність).

1.2 Аналіз та вибір методів вирішення задачі

Серед інтелектуальних технологій, за допомогою яких можна вирішити усі задачі проекту, можна виділити нейронні мережі, генетичні алгоритми і нечіткі множини. Розглянемо їх детальніше:

Нейронні мережі [6] – це один з напрямків наукових досліджень в галузі створення штучного інтелекту (ШІ), в основі якого лежить прагнення імітувати нервову систему людини. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу. Наприклад, у розпізнаванні зображень вони можуть навчатися ідентифікувати зображення, які містять котів, аналізуючи приклади зображень, мічені як «кіт» і «не кіт», і використовуючи результати для ідентифікування котів в інших зображеннях.

Вони роблять це без жодного апріорного знання про котів, наприклад, що вони мають хутро, хвости, вуса та котоподібні писки. Натомість, вони розвивають свій власний набір доречних характеристик з навчального матеріалу, який вони оброблюють.

Штучні нейронні мережі ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього.

В поширених реалізаціях штучних нейронних мереж сигнал на з'єднанні між штучними нейронами є дійсним числом, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів. Сигнали проходять від першого (входового) до останнього (виходового) шару, можливо, після проходження шарами декілька разів.

Первинною метою підходу штучних нейронних мереж було розв'язання задач таким же способом, як це робив би людський мозок. З часом увага зосередилася на відповідності певним розумовим здібностям, ведучи до відхилень від біології. ШНМ використовували в ряді різноманітних задач, включно з комп'ютерним баченням, розпізнаванням мовлення, машинним перекладом, соціально-мережевим фільтруванням, грою в настільні та відеоігри, та медичним діагностуванням.

Недоліком цього методу є відсутність твердих правил щодо вибору швидкості навчання та розміру мережі для вирішення конкретного завдання, невизначеність у підборі кількості нейронів у шарі мережі та кількості шарів

нейронної мережі. Це потребує проведення дуже великої кількості експериментів.

Генети́чний алгори́тм (англ. genetic algorithm) — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі. «Батьком-засновником» генетичних алгоритмів вважається Джон Голланд (англ. John Holland), книга якого «Адаптація в природних і штучних системах» (англ. Adaptation in Natural and Artificial Systems) є фундаментальною в цій сфері досліджень.

Задача кодується таким чином, щоб її вирішення могло бути представлено в вигляді масиву подібного до інформації складу хромосоми. Цей масив часто називають саме так «хромосома». Випадковим чином в масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція. Особи оцінюються з використанням функції пристосування, в результаті якої кожній особі присвоюється певне значення пристосованості, яке визначає можливість виживання особи. Після цього з використанням отриманих значень пристосованості вибираються особи, допущені до схрещення (селекція). До осіб застосовується «генетичні оператори» (в більшості випадків це оператор схрещення (crossover) і оператор мутації (mutation)), створюючи таким чином наступне покоління осіб. Особи наступного покоління також оцінюються застосуванням генетичних операторів і виконується селекція і мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму. Таким критерієм може бути:

1. знаходження глобального, або надоптимального вирішення;
2. вичерпання числа поколінь, що відпущені на еволюцію;

3. вичерпання часу, відпущеного на еволюцію.

Генетичні алгоритми можуть використати для пошуку рішень в дуже великих і важких просторах пошуку.

Зразу ж відзначимо, що застосування генетичного алгоритму в даній задачі є недоцільним, тому що критерій добору хромосом і використання процедур є евристичним, що зовсім не гарантує відшукання найкращого рішення. Іншим недоліком є велика обчислювана складність.

Нечіткі множини. Логіка в звичайному сенсі слова є уявленням механізму мислення, повинна бути завжди строго формалізованою. Однак в дійсності існує не одна логіка (наприклад, булева), а стільки, скільки ми побажаємо, тому що все визначається вибором відповідної системи аксіом. Звичайно, як тільки аксіоми прийняті, всі твердження, що побудовані на їх основі, повинні бути строгими, без протиріч пов'язані за правилами, встановленими в цій системі аксіом.

Нечітка логіка [7] є узагальненням класичної логіки на випадок, коли істинність розглядається як лінгвістична змінна, що приймає значення типу: "дуже істинно", "більш-менш істинно", "не дуже хибно" і т.п.

Зазначені лінгвістичні значення представляються нечіткими множинами. Основна відмінність від класичної логіки полягає в тому, що замість значень “ Істина “ і “ Хибність “ в нечіткій логіці використовується ступінь істинності, що приймає значення з нескінченної множини від 0 (Хибність) до 1 (Істина) включно.

Нечіткі множини – це засоби формалізації природно-лінгвістичних висловлювань та логічних висновків. Ідея, що лежить в основі формалізації причинно-наслідкових зв’язків між змінними «входи-виходи», полягає в описі цих зв’язків на природній мові з використанням теорії нечітких множин та лінгвістичних змінних. Моделі об’єктів будуються шляхом проектування та налаштування нечітких баз знань, що представляють собою сукупності лінгвістичних висловлювань типу ЯКЩО <входи>, ТО <виходи>. Налаштовуючи нечітку базу знань можна ідентифікувати нелінійні залежності

з необхідною точністю. Нечіткі межі множини кількісних значень, що відповідають певному лінгвістичному терму кольорового відтінка, є серйозною запорукою для використання нечітких баз знань і нечіткого логічного виводу в задачі підбору автомобільних фарб.

Серед трьох, незалежних одна від одної, теорій ідентифікацій та прийняття рішень – нечітких множин, нейронних мереж та генетичних алгоритмів, для задачі підбору автомобільних фарб найбільш доцільним є застосування теорії нечітких множин та нечіткої логіки. Основою створюваної системи підбору є формування із застосуванням методів нечіткої логіки, матричної бази знань та застосуванням до неї продукційної системи навчання та виводу.

1.3 Формулювання вимог та постановка задач на дослідження

Прогнозування “ремонтної” фарби має відбуватись на основі таблиці стандартних автомобільних фарб (рис. 1.3), бази знань з процесів старіння фарб та бази знань з умов експлуатації/зберігання автомобіля і те, як вони впливають на процес старіння. Слід звернути увагу на те, що таблиця кольорів містить не лише назву та код, а ще й склад LAB-моделі.

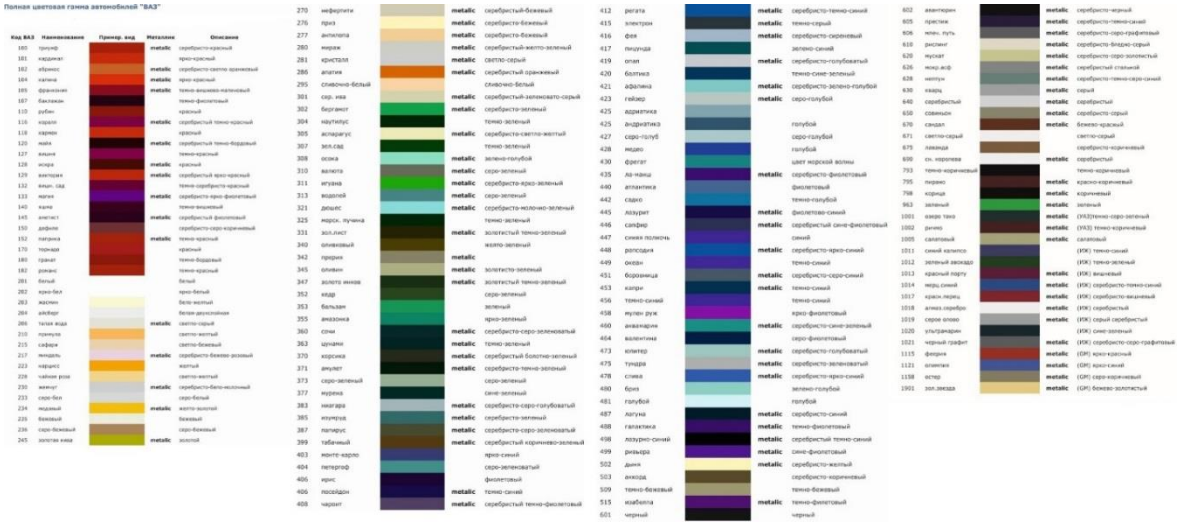


Рисунок 1.3 – Таблиця стандартних кольорів автомобільних фарб

Вхідні дані програми представлені заводським відтінком автомобіля та умовами, при яких він зберігався та експлуатувався власником. На виході ми повинні отримати склад прогнозованого відтінку у LAB-моделі, або навіть назву та код відтінку, якщо його склад співпав з тим що вже є у таблиці.

Більш детальніше на рисунках 1.4, 1.5.

Вхідні дані

- Назва заводського відтінку (C)
- Клімат (W)
- Срок експлуатації (T)
- Пробіг (P)
- Утримання (S)
- Частота мийки (N)
- Використання хімікатів (H)
- Вид мийки (V)
- Покриття доріг (R)

Вихід

- L (освітленість)
- LAB

Рисунок 1.4 – Вхідні дані

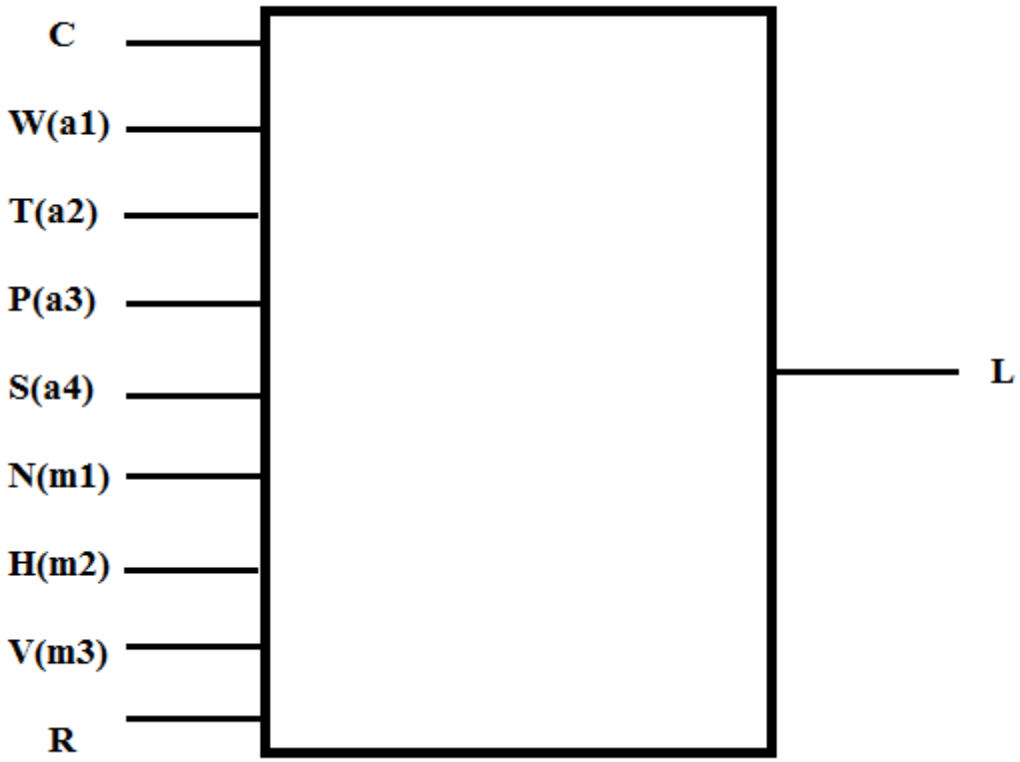


Рисунок 1.5 – Входи та виходи модуля підбору автомобільних фарб

Технічне завдання у повному об’ємі приведене у додатку А.

1.4 Висновок

В першому розділі магістерської кваліфікаційної роботи розглянута предметна область ідентифікації та прогнозування трансформації кольорових відтінків, проведено детальний аналіз відомих рішень, проаналізовані методи вирішення задачі і обгрунтовано вибір технології вирішення. В результаті приходимо до висновку, що найкращим є метод нечітких множин. На основі сформульованих вимог та вибраного методу зроблена постановка задачі на математичне моделювання та проектування програмного додатку.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОУВАННЯ ТРАНСФОРМАЦІЇ АВТОМОБІЛЬНИХ ФАРБ НА ОСНОВІ АПАРАТУ НЕЧІТКОЇ ЛОГІКИ

2.1 Формалізація задачі прогнозування трансформації автомобільних фарб

Ідея, що лежить в основі формалізації причинно-наслідкових зв'язків між змінними «входи-виходи», полягає в описі цих зв'язків на природній мові з використанням теорії нечітких множин та лінгвістичних змінних. Мета даного підрозділу полягає у введенні основних формалізмів, необхідних для визначення нечітких баз знань, що є носієм експертної інформації. В основу цього покладена робота[8].

Нами розглядається об'єкт з одним виходом та n входами виду:

$$L = f_y\,(x_1, x_2, \dots, x_n)\,, \tag{2.1}$$

де L – вихідна змінна; $x_1, x_2, \dots, x_n$ – вхідні змінні.

Змінні $x_1, x_2, \dots, x_n$та у можуть бути кількісними і якісними. Для нашого випадку кількісними змінними є: Срок експлуатації [0,50] років, частота мийки автомобіля [0,15] разів у рік, та інші змінні, які легко вимірюються в прийнятих для них кількісних шкалах.

Окрім «кольорового відтінка», прикладом змінної, для якої не існує природної кількісної шкали, є ПОКАЗНИК ВИКОРИСТАННЯ ХІМІКАТІВ, який може бути оцінений якісними термами (низький, помірний, високий) або вимірюватися в штучних шкалах, наприклад, по 5-бальній, 10-бальній, …, 100-бальній системах.

Для кількісних змінних передбачаються відомими зміни:

$$U_i = \left[\underline{x}_i, \overline{x}_j\right], i = \overline{1,n} \ , \tag{2.2}$$

$$Y = \left[\underline{y}, \overline{y}\right] \ , \tag{2.3}$$

де $\underline{x}_i$ ($\overline{x}_i$) - нижнє (верхнє) значення вхідної змінних $\overline{x}_j$, $i = \overline{1,n}$,

$\underline{y}$ ($\overline{y}$) – нижнє (верхнє) значення вихідної змінної y.

Для якісних змінних $x_l \div x_n$ та y передбачається, що змінюються множини всіх можливих значень:

$$U_i = [v_i^1, v_i^2, \dots, v_i^{q_i}], i = \overline{1,n} \ , \tag{2.4}$$

$$Y = [y^1, y^2, \dots, y^{q_m}] \ , \tag{2.5}$$

де $v_i^l (v_i^{q_i})$ - бальна оцінка, що відповідає найменшому (найбільшому) значенню вхідної змінної x_j;

$y^1 (y^{q_m})$ - бальна оцінка, що відповідає найменшому (найбільшому) значенню вихідної змінної y;

$q_j, i = \overline{1,n}$ та q_m - потужності множин (3.4) та (3.5), при чому в загальному випадку $q_1 \neq q_2 \neq \cdots \neq q_n \neq q_m$.

2.2 Ієрархічна модель логічного виведення

Ієрархічний взаємозв’язок між вхідними параметрами, класами вхідних параметрів і вихідної змінної (інтегральним показником) представимо у вигляді дерева (рис. 2.2), якому відповідає система відношень:

$$L = f_L(C, A, M, R), \tag{2.6}$$

$$A = f_M(W, T, S), \tag{2.7}$$

$$M = f_A(N, H, V), \tag{2.8}$$

де L – вихідна змінна, глобальний показник корекції фарби;

C – код відтінку фарби від заводу виробника.

A – локальний показник корекції фарби в залежності від умов експлуатації.

M – локальний показник корекції фарби в залежності від режиму мийки.

W – показник різновиду клімату експлуатації авто.

T - термін експлуатації автомобіля.

P – пробіг автомобіля.

S – режим стоянки автомобіля.

W, T, P, S – вхідні змінні, віднесені до класу A, показники режиму експлуатації.

N – частота мийки.

H – частота використання хімікатів

V – вид мийки.

N, H, V– вхідні змінні, віднесені до класу M, показники режиму мийки.

Фазифікація змінних:

L - глобальний показник корекції фарби, приймає значення одного із 7 лінгвістичних термів, що представлені нечіткими множинами показника L із кольорової моделі LAB (-3, -2, -1, 0, 1, 2, 3)

C – код відтінку заводської фарби, ціле натуральне число в межах від 100 до 967.

A - локальний показник корекції фарби в залежності від умов експлуатації. Приймає наступні значення зміни відтінку: ДМ – дуже малі, М – малі, МС – мало-середні, С – середні, БС – більш середні, В – велике, ДВ – дуже великі.

M - локальний показник корекції фарби в залежності від умов мийки. Приймає наступні значення зміни відтінку: ДМ – дуже малі, М – малі, МС – мало-середні, С – середні, БС – більш середні, В – велике, ДВ – дуже великі.

W – клімат, в умовах якого експлуатується автомобіль (полярний, субполярний, помірний, субтропічний, тропічний, субекваторіальний, екваторіальний).

T – термін експлуатації автомобіля в роках (0-5, 5-10, 10-15, 15-20, 20-25, 25-30).

P – пробіг автомобіля у кілометрах (0 – 100.000, 100.000 – 200.000, 200.000 – 300.000, 300.000 – 400.000, 400.000 – 500.000, 500.000 – 600.000)

S – режим стоянки автомобіля (гаражне утримання, утримання під відкритим небом)

N – частота мийки (рідко, помірно, часто).

H – кількість використаних хімікатів під час мийки (мала, середня, велика).

V – вид мийки (ручна, автоматизована).

R – покриття доріг, на яких в основному експлуатувався автомобіль (асфальт, грунт, гравій, пісок, сніг).

Дерево логічного виведення результату на рис. 2.3 відображає лише дворівневу ієрархію вхідних змінних, проте деталізація може бути продовжена по рівням.

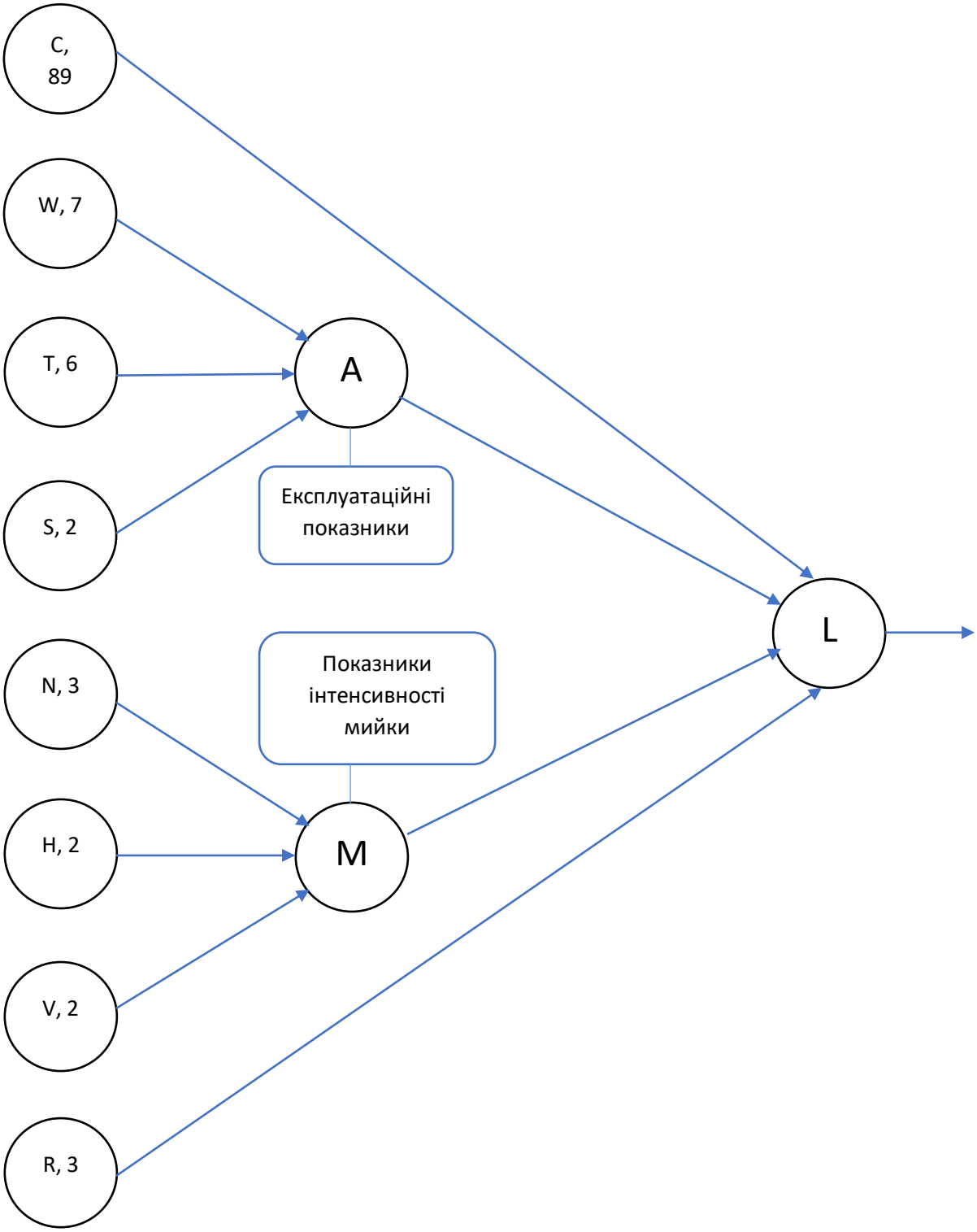


Рисунок 2.1 – Дерево логічного виведення

2.3 Нечітка база знань

Настройка нечіткого класифікатора є знаходженням таких параметрів функцій приналежностей термів вхідних змінних і вагових коефіцієнтів правил, які мінімізують відхилення між бажаною і дійсною поведінкою нечіткою класифікатора на навчальній вибірці. Критерій близькості можна визначити різними способами.

Візьмемо N експериментальних даних, які зв`язують входи і виходи об`єкта ідентифікації, і розподілимо їх наступним чином:

$$N = k_1 + k_2 + \cdots k_m,$$

Де k_j – число експериментальних даних, які відповідають вихідному рішенню $d_j, j = \overline{1,m},\ m$ – число вихідних рішень, причому, в загальному випадку $k_1 \neq k_2 \neq \cdots \neq k_m$.

Передбачається, що $N < l_1 \cdot l_2 \cdot \ldots \cdot l_n$, тобто число відібраних експериментальних даних менше повного перебору різних поєднань рівнів $(l_i, i = \overline{1,n})$ зміни вхідних змінних об`єкта.

Пронумеруємо N експериментальних даних наступним чином:

11, 12, $1k_1$ – номери комбінацій вхідних змінних для вирішення d_1;

21, 22, $2k_2$ – номери комбінацій вхідних змінних для вирішення d_2;

…

j1, j2, jk_j – номери комбінацій вхідних змінних для вирішення d_j;

…

m1, m2, mk_m – номери комбінацій вхідних змінних для вирішення d_m.

Матрицею знань назвемо таблицю, сформовану за такими правилами (табл.2.1):

Копіювання навчаючої вибірки в базу знань – для кожного екземпляра навчаючої вибірки формується окреме правило. Перевагою даного методу є простота та висока швидкість роботи, недоліком – відсутність узагальнюючих властивостей і громіздкість одержуваної мережі.

1) Розмірність матриці дорівнює $(n+1) \times N,$ де $(n+1)$ – число стовбців, а $N = k_1 + k_2 + \cdots k_m$ – число рядків.
2) Перші n стовбців матриці відповідають вхідним змінним $x_i,\ i = \overline{1,n}$, а $(n+1) -$ й стовбець відповідає значенням d_j вихідної змінної $y(j = \overline{1,m})$.
3) Кожен рядок матриці являє собою деяку комбінацію значень вхідних змінних, віднесених експертом до одного з можливих значень вихідної змінної y. При цьому : перші k_1 рядків відповідають значенню вихідної змінної $y = d_1$, другі k_2 рядків $y = d_2$,…, останні k_m рядків – значенню $y = d_m$
4) Елемент a_i^{jp}, який стоїть на перетині i-го стовбця і jp-го рядка відповідають лінгвістичній оцінці параметра x_i в рядку нечіткої бази даних з номером jp. При цьому лінгвістична оцінка a_i^{jp} вибирається із терм-множини, яка відповідає змінній x_i, тобто $a_i^{jp} \in A_i\ ,\ i = \overline{1,n},\ j = \overline{1,m}\ , p = \overline{1,k_j}$

Таблиця 2.1 – Узагальнена матриця знань

Номер вхідної комбінації значень	Вхідні змінні				Вихідна змінна
	x_1	x_2	$\dots x_i \dots$	x_n	
11 12 … $1k_1$	a_1^{11} a_1^{12} $a_1^{1k_1}$	a_2^{11} a_2^{12} $a_2^{1k_1}$	$\dots a_i^{11} \dots$ $\dots a_i^{12} \dots$ $\dots a_i^{1k_1} \dots$	a_n^{11} a_n^{12} $a_n^{1k_1}$	d_1
…					
j1 *j2* … jk_j	a_1^{j1} a_1^{j2} $a_1^{jk_j}$	a_2^{j1} a_2^{j2} $a_2^{jk_j}$	$\dots a_i^{j1} \dots$ $\dots a_i^{j2} \dots$ $\dots a_i^{jk_j} \dots$	a_n^{j1} a_n^{j2} $a_n^{jk_j}$	d_j
…					
m1 *m2* … mk_m	a_1^{m1} a_1^{m2} $a_1^{mk_m}$	a_2^{m1} a_2^{m2} $a_2^{mk_m}$	$\dots a_i^{m1} \dots$ $\dots a_i^{m2} \dots$ $\dots a_i^{mk_m} \dots.$	a_n^{m1} a_n^{m2} $a_n^{mk_m}$	d_m

Розглянемо частичну матрицю знань на прикладі двох відтінків автомобільної фарби – Сафарі і Тріумф (табл. 2.2; 2.3; 2.4)

Для початку опишемо матрицю знань для локального показника корекції фарби в залежності від умов експлуатації, тобто А.

Таблиця 2.2 – Матриця знань для показника А.

W	T	P	S	A
Помірний	0-5	0-100.000	Гаражне	ДМ
Помірний	5-10	0-100.000	Гаражне	ДМ
Помірний	0-5	100.000-200.000	Гаражне	ДМ
Помірний	5-10	200.000-300.000	Гаражне	М
Помірний	10-15	100.000-200.000	Гаражне	М
Помірний	10-15	200.000-300.000	Гаражне	М
Субекваторіальний	10-15	300.000-400.000	Гаражне	МС
Субекваторіальний	15-20	200.000-300.000	Гаражне	МС
Субекваторіальний	15-20	300.000-400.000	Гаражне	МС
Субекваторіальний	10-15	200.000-300.000	Відкрита стоянка	С
Субекваторіальний	15-20	200.000-300.000	Відкрита стоянка	С
Субекваторіальний	15-20	300.000-400.000	Відкрита стоянка	С
Екваторіальний	15-20	300.000-400.000	Відкрита стоянка	БС
Екваторіальний	15-20	400.000-500.000	Відкрита стоянка	БС
Екваторіальний	20-25	400.000-500.000	Відкрита стоянка	БС
Субполярний	20-25	300.000-400.000	Відкрита стоянка	В
Субполярний	25-30	400.000-500.000	Відкрита стоянка	В
Субполярний	25-30	500.000-600.000	Відкрита стоянка	В
Полярний	20-25	500.000-600.000	Відкрита стоянка	ДВ
Полярний	25-30	400.000-500.000	Відкрита стоянка	ДВ
Тропічний	25-30	500.000-600.000	Відкрита стоянка	ДВ

Введена матриця знань визначає систему логічних висловлювань типу «ЯКЩО-ТО, ІНАКШЕ»[9], які пов`язують значення вхідних змінних з одним із можливих типів вирішення:

Якщо (W = “помірний”) і (T = 0-5) і (P = 0 – 100.000) і (S = “гаражне”) або (W = “помірний”) і (T = 5-10) і (P = 0 – 100.000) і (S = “ гаражне ”) або (W = “помірний”) і (T = 0-5) і (P = 100.000 – 200.000) і (S = “гаражне”), то А = “ДМ”.

Тепер опишемо таку ж матрицю знань для локального показника корекції фарби в залежності від умов мийки, тобто М.

Таблиця 2.3 - Матриця знань для показника М.

N	H	V	M
Рідко	Мала	Ручна	ДМ
Рідко	Середня	Ручна	ДМ
Помірно	Мала	Ручна	М
Помірно	Середня	Ручна	М
Рідко	Мала	Автоматизована	МС
Рідко	Середня	Автоматизована	МС
Помірно	Мала	Автоматизована	С
Помірно	Середня	Автоматизована	С
Часто	Мала	Автоматизована	БС
Часто	Середня	Автоматизована	БС
Помірно	Висока	Ручна	В
Помірно	Висока	Автоматизована	В
Часто	Висока	Ручна	ДВ
Часто	Висока	Автоматизована	ДВ

Введена матриця знань визначає систему логічних висловлювань типу «ЯКЩО-ТО, ІНАКШЕ», які пов`язують значення вхідних змінних з одним із можливих типів вирішення:

Якщо (N = “рідко”) і (H = “мала”) і (V = “ручна”) або (N = “рідко”) і (H = “середня”) і (V = “ручна”) або (N= “помірно”) і (H = “мала”) і (V = “ручна”), то M = “ДМ”.

Якщо (N = “часто”) і (H = “мала”) і (V = “ручна”) або (N = “рідко”) і (H = “середня”) і (V = “ручна”) або (N= “помірно”) і (H = “мала”) і (V = “ручна”), то M = “С ”.

Якщо (N = “рідко”) і (H = “мала”) і (V = “автоматична”) або (N = “рідко”) і (H = “середня”) і (V = “ручна”) або (N= “помірно”) і (H = “мала”) і (V = “ручна”), то M = “М”.

Тепер, маючи дані як впливають на відтінок умови експлуатації та мийки, ми можемо описати матрицю знань для глобального показника корекції фарби, а саме L.

Таблиця 2.4 – Матриця знань на прикладі двох відтінків

C	A	M	R	L
100.Тріумф	ДМ	ДМ	Асфальт	0
	ДМ	М	Асфальт	0
	М	ДМ	Асфальт	0
	М	М	Асфальт	+1
	МС	М	Асфальт	+1
	М	МС	Асфальт	+1
	МС	МС	Асфальт	+1
	МС	С	Асфальт	+1
	С	МС	Асфальт	+1
	С	С	Асфальт	+2
	БС	С	Асфальт	+2
	С	БС	Асфальт	+2
	БС	БС	Асфальт	+2
	В	БС	Асфальт	+3
	БС	В	Асфальт	+3
	В	В	Асфальт	+3
	ДВ	В	Асфальт	+3
	В	ДВ	Асфальт	+3
	ДВ	ДВ	Асфальт	+3
215.Сафарі	ДМ	ДМ	Асфальт	0
	ДМ	М	Асфальт	0
	М	ДМ	Асфальт	0
	М	М	Асфальт	-1
	МС	М	Асфальт	-1
	М	МС	Асфальт	-1
	МС	МС	Асфальт	-1
	МС	С	Асфальт	-1
	С	МС	Асфальт	-1
	С	С	Асфальт	-2
	БС	С	Асфальт	-2
	С	БС	Асфальт	-2
	БС	БС	Асфальт	-2
	В	БС	Асфальт	-3
	БС	В	Асфальт	-3
	В	В	Асфальт	-3
	ДВ	В	Асфальт	-3
	В	ДВ	Асфальт	-3
	ДВ	ДВ	Асфальт	-3

Введена матриця знань визначає систему логічних висловлювань типу «ЯКЩО-ТО, ІНАКШЕ», які пов`язують значення вхідних змінних з одним із можливих типів вирішення:

Якщо (С = “Тріумф”) і (А = “дуже малі”) і (М = “дуже малі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “дуже малі”) і (М = “малі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “малі”) і (М = “дуже малі”) і (R = “асфальт”), то L = 0.

Якщо (С = “Тріумф”) і (А = “малі”) і (М = “малі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “менш середні”) і (М = “малі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “малі”) і (М = “менш середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “менш середні”) і (М = “менш середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “менш середні”) і (М = “середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “середні”) і (М = “менш середні”) і (R = “асфальт”), то L = (+1).

Якщо (С = “Тріумф”) і (А = “середні”) і (М = “середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “більш середні”) і (М = “середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “середні”) і (М = “більш середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “більш середні”) і (М = “більш середні”) і (R = “асфальт”), то L = (+2).

Якщо (С = “Тріумф”) і (А = “високі”) і (М = “більш середні”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “більш середні”) і (М = “високі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “високі”) і (М = “високі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “дуже високі”) і (М = “високі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “високі”) і (М = “дуже високі”) і (R = “асфальт”) або (С = “Тріумф”) і (А = “дуже високі”) і (М = “дуже високі”) і (R = “асфальт”), то L = (+3).

Будемо називати подібну систему логічних висловлювань *нечіткою базою знань.*

2.4 Функції належності

За визначенням, функція приналежності[8] характеризує суб'єктивну міру (в діапазоні [0,1]) впевненості експерта в тому, що чітке значення відповідає нечіткому терму. Найбільшого поширення в практичних застосуваннях [8] отримали трикутні, трапецієподібні і дзвоновидні (гаусові) функції приналежності, параметри яких дозволяють змінювати форму функцій.

Ми використовуєм просту і зручну для налаштування аналітичну модель функцій приналежності змінної нечіткому терму у вигляді:

$$\mu^T = \frac{1}{1+\left(\frac{x-b}{c}\right)^2}, \tag{2.9}$$

Де b і c - параметри настройки: b - координата максимуму функції,

$\mu^T(b) = 1$; c - коефіцієнт концентрації - розтягування функції (рис. 2.4). Для нечіткого терма T число b представляє найбільш можливе значення змінної x.

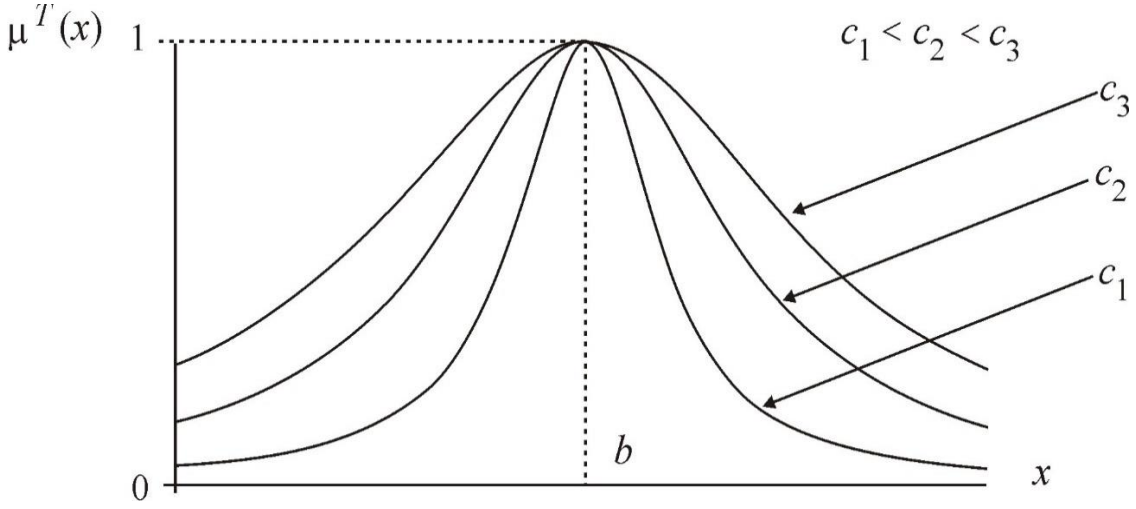


Рисунок 2.2 – Модель функції належності

Для прикладу моделювання використовувалися дзвоноподібні функції приналежності, задані на єдиній універсальній множині $L^* = [0,256]$, з параметрами центру (b) і стиснення-розтягування (c), представленими в табл. 2.5.

Таблиця 2.5 - Параметри функцій належності

Терм	ДМ	М	МС	С	БС	В	ДВ
b	0	42,7	85,3	128	170,7	213,4	256
c	15	15	15	15	15	15	15

2.5 Розробка структури технології прогнозування трансформації автомобільних фарб

Сформуємо послідовність етапів технології прогнозування трансформації автомобільних фарб. Ця технологія базується на технології ідентифікації на основі апарату нечіткої логіки, описаній в [8].

1. Формування вектору вхідних змінних.
2. Фазифікація змінних. Для кожної вхідної змінної надаємо перелік лінгвістичних термів, що відповідають станам вхідної змінної.
3. Для кожного лінгвістичного терму певної змінної формуємо вид та параметри функцій належності.
4. Будуємо дерево ієрархічного виведення результату.
5. На основі дерева ієрархічного виведення результату формуємо експертну базу знань у вигляді матриць відповідності станів вхідних та вихідних змінних та продукційних правил виведення результату «ЯКЩО-ТО».
6. Для одержаної експертної бази знань створюємо програмний модуль виведення результату

7. Етап «грубого» налаштування – на основі думок експертів уточнюється кількість лінгвістичних термів (станів) кожної змінної вхідного вектору та параметри функцій належності цих термів.
8. Етап «тонкого» налаштування (навчання експертної системи) – результати, одержані в процесі логічного виведення порівнюються з результами колориметрії реальних автомобільних фарб, що пройшли певний срок експлуатації в певних умовах. При цьому коректуються параметри функцій належності лінгвістичних термів до повного співпадіння результатів

Процес навчання можна буде організувати в автоматизованому режимі в режимі пробної експлуатації експертної системи.

Структура технології прогнозування трансформації автомобільних фарб показана на рис. 2.3.

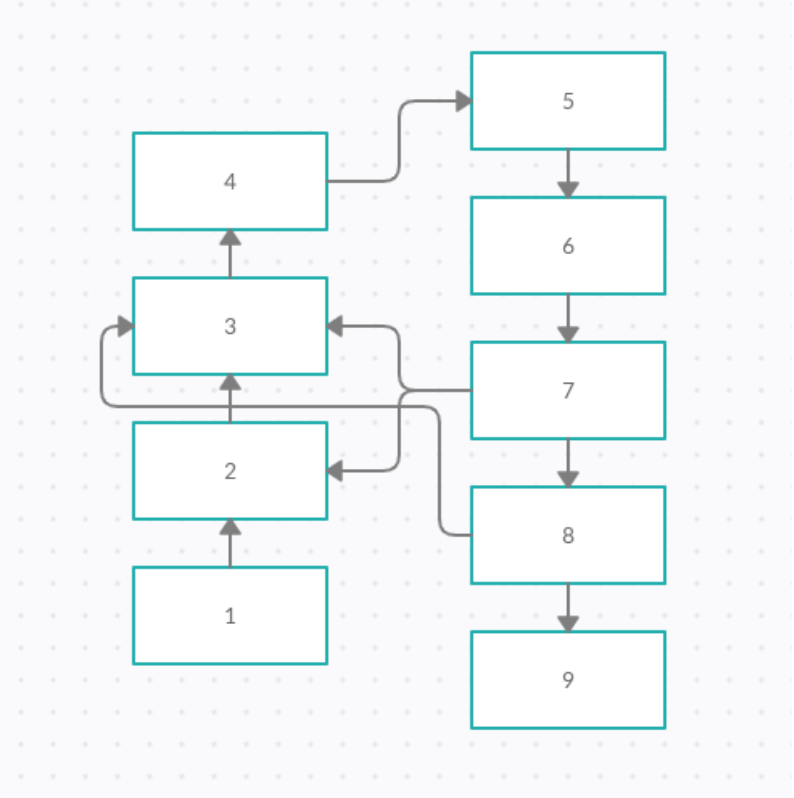


Рисунок 2.3 – структура технології

2.6 Висновок

У другому розділі магістерської кваліфікаційної роботи, який присвячений проектуванню та розробці нової технології, була розроблена структура модуля, проведена формалізація задачі підбору фарб, описана ієрархічна модель логічного виведення, розроблені та описані нечіткі бази знань впливу різноманітних факторів та умов експлуатації, описана функція належності. Також була розроблена структура технології. Вона являє собою наступні пункти:

1. Формування вектору вхідних змінних.
2. Фазифікація змінних. Для кожної вхідної змінної надаємо перелік лінгвістичних термів, що відповідають станам вхідної змінної.
3. Для кожного лінгвістичного терму певної змінної формуємо вид та параметри функцій належності.
4. Будуємо дерево ієрархічного виведення результату.
5. На основі дерева ієрархічного виведення результату формуємо експертну базу знань у вигляді матриць відповідності станів вхідних та вихідних змінних та продукційних правил виведення результату «ЯКЩО-ТО».
6. Для одержаної експертної бази знань створюємо програмний модуль виведення результату.
7. Етап «грубого» налаштування.
8. Етап «тонкого» налаштування.

3 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ТРАНСФОРМАЦІЇ АВТОМОБІЛЬНИХ ФАРБ НА ОСНОВІ АПАРАТУ НЕЧІТКОЇ ЛОГІКИ

3.1 Декомпозиція середовища та розробка структури компонентів

В результаті аналізу попередніх даних, представимо діаграму компонентів так, як це показано на рис. 3.1.

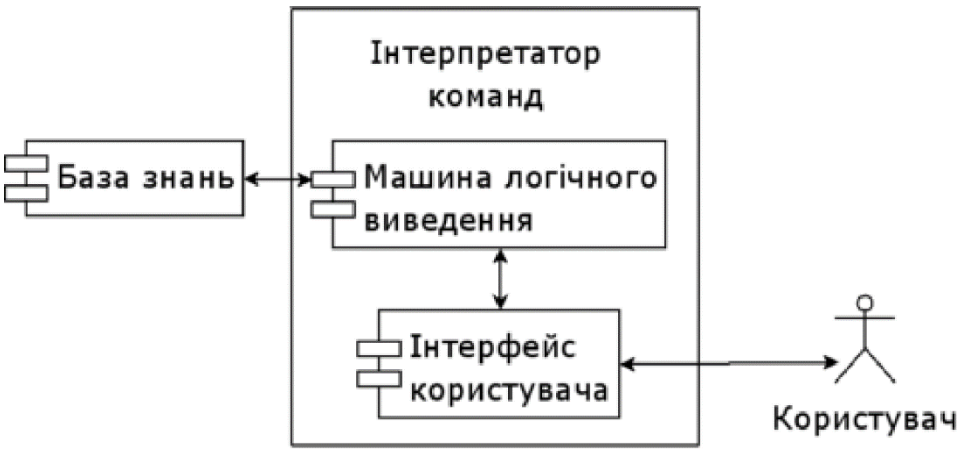


Рисунок 3.1 – Діаграма компонентів модуля

Для реалізаці бази знань та її налаштування використаємо середовище автоматизованого проектування експертних систем на основі нечіткої логіки «FUZZY EXPIRIENS». Структура середовища зображена на рис. 3.2. Вона містить 9 програмних блоків, 2 з яких складають власне експертну систему, а решта, - середовище розробки нечітких експертних систем. Пояснимо їх призначення.

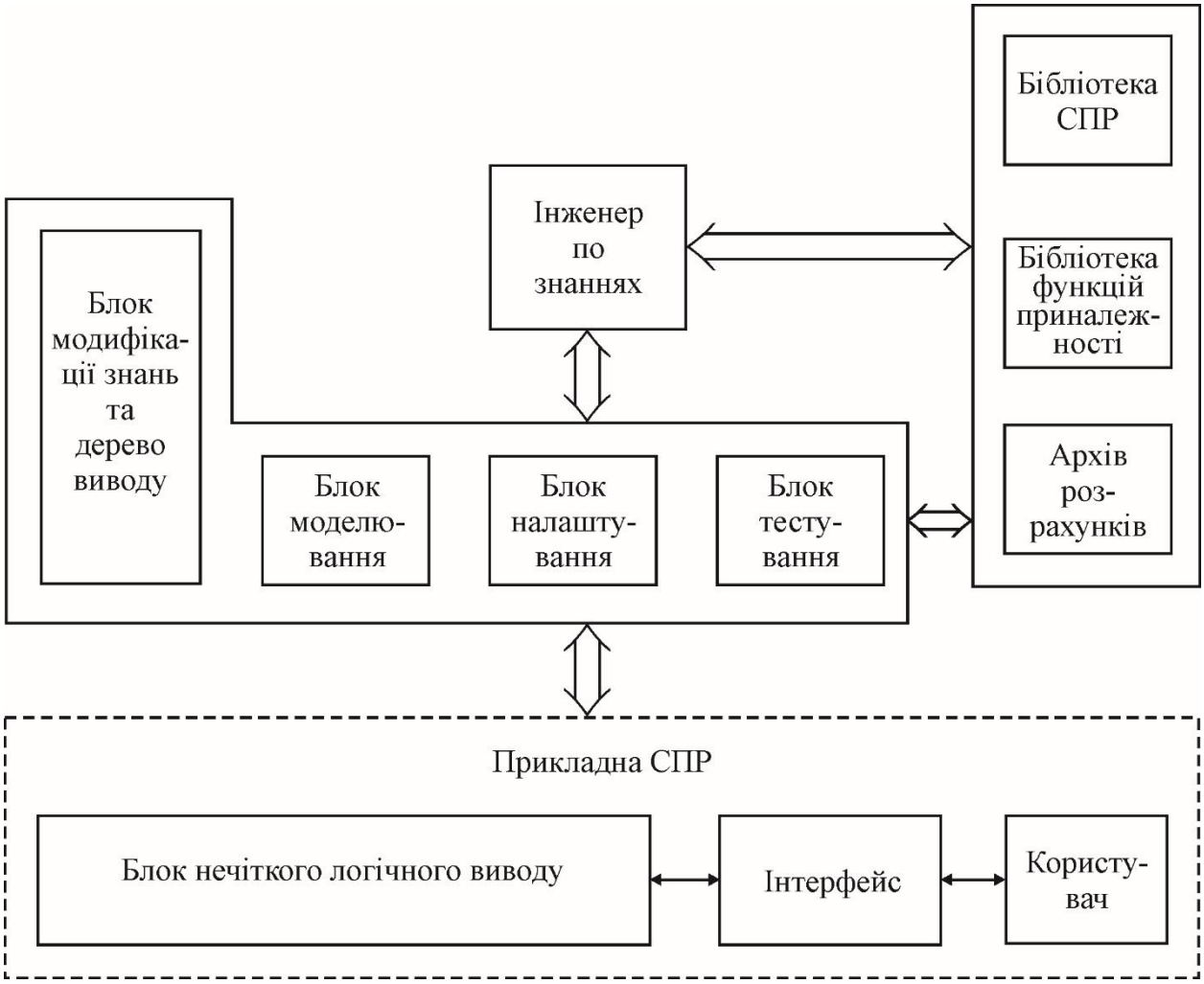


Рисунок 3.2 – Структурна схема середовища прийняття рішень.

Блок модифікації нечіткої бази знань і дерева виводу містить всі екранні форми, необхідні для внесення змін в блоки експертної системи: дерево логічного висновку, базу знань і функції належності. Це основний блок середовища розробки нечітких баз знань. Він виконує наступні функції:

а) формування дерева логічного виводу;

б) задання функцій належності лінгвістичних термів:

- Модифікованим методом Сааті,
- Методом статистичної обробки експертної інформації,

в) заповнення нечітких баз знань;

г) збереження (витяг) варіантів експертних систем з бібліотеки.

Блок моделювання використовується для отримання графіків і поверхонь, що відбивають залежність вихідної змінної від однієї або двох вхідних змінних при фіксованих значеннях інших змінних. Мета подібного моделювання полягає в дослідженні поведінки об'єкта в різних областях факторного простору.

Блок налаштування призначений для вирішення завдань оптимізації нечіткої бази знань з метою підвищення якості ідентифікації нелінійних об'єктів. Цей блок виконує наступні функції:

а) запит навчальної вибірки;

б) вирішення задач оптимізації нечітких баз знань градієнтним методом;

в) вирішення завдань оптимізації нечітких баз знань з параметричними функціями належності за допомогою генетичного алгоритму;

г) рішення задач оптимізації нечітких баз знань з α-рівневими функціями належності за допомогою генетичного алгоритму.

Блок тестування експертної системи призначений для виконання наступних дій:

а) запит тестируючої вибірки;

б) оцінка якості ідентифікації в точках тестируючої вибірки.

Блок документування здійснює видачу інформації про побудовану експертну систему у вигляді, зручному для інженера по знаннях.

Бібліотека експертних систем використовується для зберігання створених експертних систем на різних етапах їх розробки (до налаштування і після налаштування).

Бібліотека функцій належності містить набір стандартних моделей функцій належності:

а) трапецієподібні;

б) трикутні;

в) дзвоноподібні (з параметрами b і c);

г) експоненціальні.

Архів розрахунків дозволяє зберігати результати виконуваних розрахунків на різних етапах створення і функціонування системи

3.2 Розробка алгоритма виведення логічного результату

Лінгвістична оцінка a_i^{jp} змінних $x_1 \div x_n$, які входять в логічне висловлювання про розв`язки d_j (2.10), будемо розглядати як нечіткі множини, визначені на універсальних множинах $U_i = \left[\underline{x_i}, \overline{x_i}\right]$, $i = \overline{1,n}$, $j = \overline{1,m}$.

Нехай $\mu^{a_i^{jp}}(x_i)$ – функція належності параметра $x_i \in \left[\underline{x_i}, \overline{x_i}\right]$ нечіткому терму a_i^{jp}, $i = \overline{1,n}$, $j = \overline{1,m}$, $p = \overline{1,k_j}$;

$\mu^{d_j} = (x_1, x_2, \dots, x_n)$ – залежна від n змінних функція належності вектора вхідних змінних $X = (x_1,\ x_2, \dots, x_n)$ значенню вихідної змінної $y = d_{j,}$ $j = \overline{1,m}$.[10]

Зв`язок між цими функціями визначається нечіткою базою знань (табл. 2.1 – 2.4) і може бути представлена у вигляді наступних рівнянь:

$$\mu^L(A,M,R) = \frac{max}{p=\overline{114,h_{j(1\dots7)}}}\left\{min\left[\mu^{A^{jp}}(A), \mu^{M^{jp}}(M), \mu^{R^{jp}}(R)\right]\right\}, \quad (3.1)$$

$$\mu^A(W,T,P,S) = \frac{max}{p=\overline{114,e_{j(1\dots7)}}}\left\{\frac{min}{i=\overline{1,l}}\left[\mu^{A_i^{jp}}\left(A_j\right)\right]\right\}, \quad (3.2)$$

$$\mu^{M}(N,H,V) = \frac{max}{p=114, g_{j(1\ldots7)}}\left\{\frac{min}{i=1,m}\left[\mu^{M_i^{jp}}(M_j)\right]\right\}, \quad (3.3)$$

$$\mu^{R} = \frac{max}{p=114, t_{j(1\ldots7)}}\left\{\left[\mu^{R_i^{jp}}(R_j)\right]\right\}, \quad (3.4)$$

Таким чином нами отримана система співвідношень, яка повністю відповідає узагальненому дереву логічного висновку і дозволяє обчислити ступені приналежності вектора значень вхідних змінних нечітким термам-оцінками вихідної змінної. Алгоритм нечіткого логічного висновку, що використовує узагальнене дерево виведення має вигляд:

1. Зафіксуємо вектор значень вхідних змінних

(a, m, R);

2. Визначимо значення функцій приналежності термів-оцінок вхідних змінних.

3. Використовуючи співвідношення (3.1) - (3.4) обчислимо функції приналежності $\mu^{L}(A,M,R)$ термів-оцінок вихідної величини L , яка відповідає вектору значень вхідних змінних/

4. Визначимо оцінку D_j, функція приналежності якої максимальна:

$$\mu^{L_j}(A,M,R) = max_{j=7}\left[\mu^{L_j}(A,M,R)\right]$$

Відповідна схема алгоритму приведена на рис. 3.3.

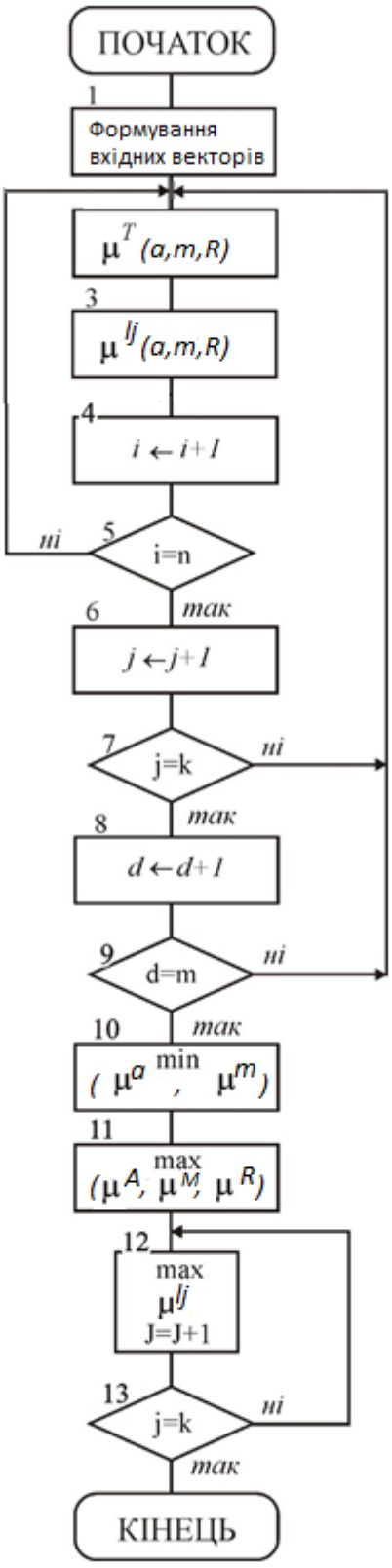


Рисунок 3.3 – Схема алгоритму логічного виведення рішення.

3.3 Розробка діаграми класів

Для розробки використаємо шаблон «FUZZY CONCLUSION», Діаграма класів блоку логічного виведення на основі шаблону приведена на рис. 3.4.

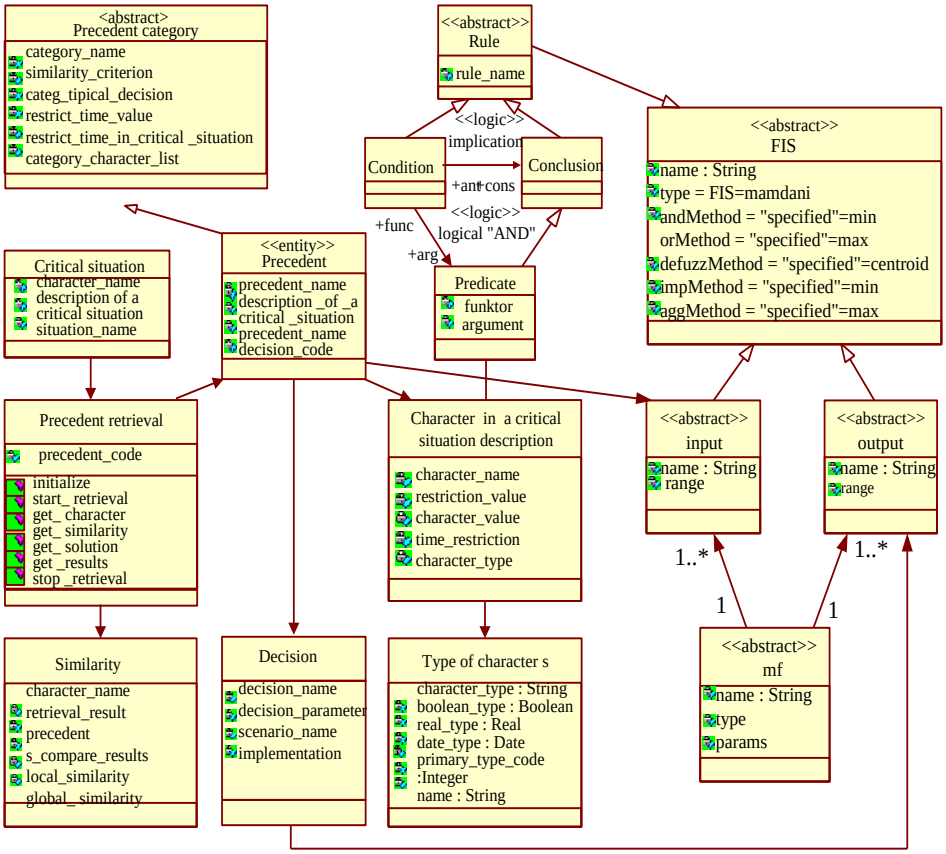


Рисунок 3.4 – Шаблон «FUZZY CONCLUSION»

Діаграма класів представлена на рис. 3.5 Вона складається із 13 наступних класів: FUNCT, TERM, HIGH, HIGHM, MIDDLE, LOWM, LOW, CYCLE, CYCLE1, CYCLE2, CYCLE3, CYCLE4, FCYCLE.

Клас FUNCT Призначення.

Клас FUNCT зберігає бібліотеку функцій належності логічних термів. Призначений для генерації через наслідування класів HIGH, HIGHM, MIDDLE, LOWM, LOW.

Відповідальність:

1. Навчання (тонке налаштування функцій належності логічних термів)
2. Формування бази знань.
3. Атрибути.
4. Зберігає параметри функцій належності різних форм для кожного із дінгвістмчних термів.
5. Методи.
6. Приведення функцій належності до єдиного універсального інтервалу. Поповнення та редагування бібліотеки функцій належності. Вибірки та корекції функцій.

Клас TERM – абстрактний клас для створення класів HIGH, HIGHM, MIDDLE, LOWM, LOW.

Класи HIGH, HIGHM, MIDDLE, LOWM, LOW Призначення.

Класи HIGH, HIGHM, MIDDLE, LOWM, LOW призначені для зберігання параметрів функцій належності, відповідно до кожного із логічних термів. Успадковує параметри та методи класу FUNCT.

Клас CYCLE Призначення.

Абстрактний клас для створення класів CYCLE1, CYCLE2, CYCLE3, CYCLE4.

Атрибути.

Кількість циклів та кількіість ітерацій у циклі.

Класи CYCLE1, CYCLE2, CYCLE3, CYCLE4 Призначення.

Зберігають мережу залежностей в межах кожного циклу.

Клас FCYCLE

Призначення - формування лічильників циклу

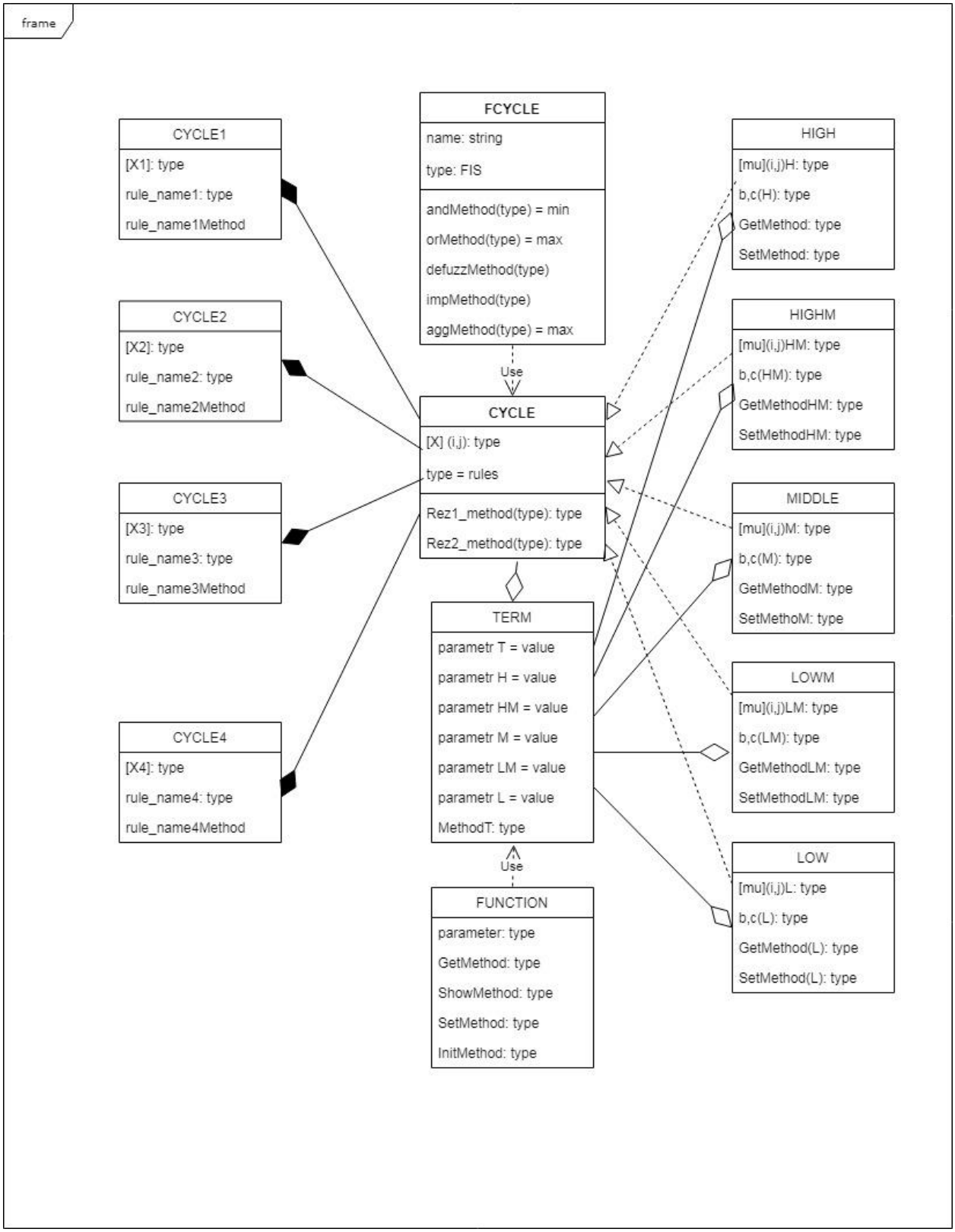


Рисунок 3.5 - Діаграма класів модуля виведення частинного показника

3.4 Висновок

В результаті виконання проектування програмного додатку, була зроблена декомпозиція середовища та розроблена структура компонентів. Також були розроблені алгоритм логічного виведення результату та діаграма класів.

Алгоритм нараховую у собі 4 цикли, при виконанні яких у матрицях знань підбираются співпадіння користування автомобіля у первних умовах і на виході виводиться результат зміни відтінка автомобільної фарби, тобто параметр освітленості.

Діаграма класів представлена на базі шаблона «FUZZY CONCLUSION». Вона складається із 13 наступних класів: FUNCT, TERM, HIGH, HIGHM, MIDDLE, LOWM, LOW, CYCLE, CYCLE1, CYCLE2, CYCLE3, CYCLE4, FCYCLE.

Блок модифікації нечіткої бази знань і дерева виводу містить всі екранні форми, необхідні для внесення змін в блоки експертної системи: дерево логічного висновку, базу знань і функції належності. Це основний блок середовища розробки нечітких баз знань.

Блок моделювання використовується для отримання графіків і поверхонь, що відбивають залежність вихідної змінної від однієї або двох вхідних змінних при фіксованих значеннях інших змінних. Мета подібного моделювання полягає в дослідженні поведінки об'єкта в різних областях факторного простору.

Блок налаштування призначений для вирішення завдань оптимізації нечіткої бази знань з метою підвищення якості ідентифікації нелінійних об'єктів.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ ПРОГНОЗУВАННЯ ТРАНСФОРМАЦІЇ АВТОМОБІЛЬНИХ ФАРБ

4.1 Обгрунтування вибору середовища та мови програмування

IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains.

Перша версія з'явилася в січні 2001 року і швидко набула популярності як перше середовище для Java з широким набором інтегрованих інструментів для рефакторінга, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача. Серед інших можливостей, середа добре сумісна з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування на мові Ruby.

Починаючи з версії 9.0, середовище доступне в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Kotlin, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition, доступною під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків.

Середовище було обране саме через доступність, високу функціональність та можливість встановлювати плагіни з додатковими функціями.

C # і Java - дві мови програмування, розвиваючих мову програмування C ++, з синтаксисом, який багато в чому успадковує синтаксис C ++, і створених багато в чому в умовах конкуренції, і, внаслідок цього, володіють певним схожістю, а також мають і ряд відмінностей .

Мови C # і Java з'явилися в різний час. Мова Java була створена задовго до появи C #. Під назвою Oak Java була розроблена компанією Sun Microsystems в 1990 р, а в 1995 була випущена перша бета-версія Java. Створення C # було анонсовано в 2000 році, а в 2002 році вийшла перша версія платформи .NET, що підтримує C #. Таким чином, якщо Java створювалась спираючись більшою мірою на досвід мов Objective C і C, то для C # такою опорою були C ++ і сама Java. І, незважаючи на свою назву, C # виявилась ближче до Java, ніж до C ++ [12].

З точки зору розробника мови Java і C # дуже схожі. Обидві мови є строго типізованими, об'єктними. Обидві увібрали в себе багато чого з синтаксису C ++, але на відміну від C ++, простіші в освоєнні для початківців. Обидві запозичили з C набір основних ключових слів і службових символів, в тому числі фігурні дужки для виділення блоків. Обидві мови спираються на збірку сміття. Обидві мови супроводжуються багатими колекціями бібліотек. Але є в мовах також свої особливості і відмінності, сильні і слабкі сторони. C # врахував багато недоліків Java, і виправив їх у своїй реалізації. Але і Java не стоїть на місці, розвиваючись паралельно з C #.

Кік Редек з Microsoft вважає C # більш складною мовою, ніж Java. На його думку, «мова Java була побудована таким чином, щоб уберегти розробника від стрільби собі в ногу» (англ. «Java was built to keep a developer from shooting himself in the foot»), а «C # була побудована так, щоб дати розробникові пістолет, але залишити його на запобіжнику »(англ.« C # was built to give the developer a gun but leave the safety turned on »).

Java старше, ніж C # і побудована на великій і активній користувальницької базі, ставши lingua franca в багатьох сучасних областях інформатики, особливо таких, де задіяні мережі. Java домінує в курсах програмування американських університетів і коледжів, і літератури по Java сьогодні набагато більше, ніж по C #. Зрілість і популярність Java привели до більшої кількості бібліотек і API на Java (багато з яких відкриті), ніж на C #.

На відміну від Java, C # - мова відносно нова. Microsoft вивчила існуючі мови, такі як Java, Delphi і Visual Basic, і змінила деякі аспекти мови для кращої відповідності потребам деяких типів додатків.

Відносно Java можна почути критику, що ця мова повільно розвивається, в ній не вистачає деяких можливостей, які полегшують модні шаблони програмування та методології. Мову C # критикують в тому, що її розробники, можливо, занадто поспішають догодити миттєвим течіям в програмуванні ціною фокусування і простоти мови. Очевидно, проектувальники Java зайняли більш консервативну позицію по додаванню великих нових можливостей в синтаксис мови, ніж в інших сучасних мовах - можливо, не бажаючи застосування якоїсь мови до течіям, які в довгостроковій перспективі можуть завести в глухий кут. З випуском Java 5.0 ця тенденція багато в чому була порушена, оскільки в ній ввели кілька великих нових можливостей мови: цикл типу foreach, автоматичне загортання, методи зі змінним числом параметрів, перелічуваних типи, узагальнені типи і анотації (всі вони присутні і в C #) . Починаючи з Java 8, почалося активне впровадження нових функцій, зокрема: лямбда-вирази, ключове слово var, модульність в рамках проекту Jigsaw і так далі.

C #, в свою чергу, розвивається швидше, набагато слабкіше обмежуючи себе в додаванні нових проблемно-орієнтованих можливостей. Особливо ця тенденція проявилася в версії C # 3.0, в якій, наприклад, з'явилися SQL-подібні запити. (Нові можливості при цьому будуються так, щоб мова залишалася мовою загального призначення. Проблемно-орієнтовані доповнення до Java розглядалися, але, у крайньому разі на сьогоднішній день, були відкинуті.

Отже, широкі можливості Java, простота застосування, незалежність від платформи і вбудовані функції захисту роблять цю мову програмування однією з найкращих для створення різноманітних додатків та ПП.

4.2 Вибір моделі змішування кольорів

LAB - абревіатура назви двох різних (хоча і схожих) кольорових просторів. Більш відомим і поширеним є CIELAB (точніше, CIE 1976 L * a * b *), інших - Hunter Lab (точніше, Hunter L, a, b). Таким чином, Lab - це неформальна абревіатура, не визначальна колірний простір однозначно. Найчастіше, говорячи про Lab, мають на увазі CIELAB [13].

При розробці Lab переслідувалася мета створення кольорового простору, зміна кольору в якому буде більш лінійною з точки зору людського сприйняття (в порівнянні з XYZ), тобто з тим, щоб однакова зміна значень координат кольору в різних областях кольорового простору виробляла однакове відчуття зміни кольору. Таким чином математично коректувалася б нелінійність сприйняття кольору людиною. Обидва кольорових простори розраховуються щодо певного значення точки білого. Якщо значення точки білого додатково не вказується, мається на увазі, що значення Lab розраховані для стандартного освітлювача D50.

У кольоровому просторі Lab значення освітлення відокремлено від значення хроматичної складової кольору (тон, насиченість). Освітленість задана координатою L (змінюється від 0 до 100, тобто від самого темного до самого світлого), хроматична складова - двома декартовими координатами a і b. Перша позначає стан кольору в діапазоні від зеленого до червоного, друга - від синього до жовтого.

На відміну від колірних просторів RGB або CMYK, які є, по суті, набором апаратних даних для відтворення кольору на папері або на екрані монітора (колір може залежати від типу друкарської машини, марки фарб, вологості повітря в цеху або виробника монітора і його налаштувань) , Lab однозначно визначає колір. Тому Lab знайшов широке застосування в

програмному забезпеченні для обробки зображень в якості проміжного колірного простору, через яке відбувається конвертація даних між іншими колірними просторами (наприклад, з RGB сканера в CMYK друкованого процесу). При цьому особливі властивості Lab зробили редагування в цьому просторі потужним інструментом корекції.

Завдяки характеру визначення кольору в Lab з'являється можливість окремо впливати на яскравість, контраст зображення і на його колір. У багатьох випадках це дозволяє прискорити обробку зображень, наприклад, при додрукарській підготовці. Lab надає можливість виборчого впливу на окремі кольори в зображенні, посилення колірного контрасту, незамінними є і можливості, які це кольоровий простір надає для боротьби з шумом на цифрових фотографіях.

З огляду на те що в перетворенні з XYZ в LAB використовуються формули, що містять кубічні корені, LAB є сильно нелінійною системою. Це ускладнює застосування звичних операцій над 3-мірними векторами в цьому кольоровому просторі.

4.3 Використання бібліотек та допоміжних додатків

UML (англ. Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.[14]

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які

підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML чудово зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Swing[15] — інструментарій для створення графічного інтерфейсу користувача (GUI) мовою програмування Java. Це частина бібліотеки базових класів Java (JFC, Java Foundation Classes).

Swing розробляли для забезпечення функціональнішого набору програмних компонентів для створення графічного інтерфейсу користувача, ніж у ранішого інструментарію AWT. Компоненти Swing підтримують

специфічні look-and-feel модулі, що динамічно підключаються. Завдяки ним можлива емуляція графічного інтерфейсу платформи (тобто до компоненту можна динамічно підключити інші, специфічні для даної операційної системи вигляд і поведінку). Основним недоліком таких компонентів є відносно повільна робота, хоча останнім часом це не вдалося підтвердити через зростання потужності персональних комп'ютерів. Позитивна сторона — універсальність інтерфейсу створених програм на всіх платформах.

Переваги:

1. Незалежність від платформи: Swing — платформо-незалежна бібліотека, що означає, що програму з використанням Swing можна запустити на всіх платформах, які підтримують JVM.

Можливість для розширення: Swing — дуже розподілена архітектура, яка дозволяє «підключати» реалізації користувача вказаної інфраструктури інтерфейсів: користувачі можуть створити свою власну реалізацію цих компонентів, щоб замінити компоненти без обумовлення (за замовчуванням). Взагалі, користувачі Swing можуть розширити структуру, продовжуючи (з допомогою extends) існуючі класи і/або створюючи альтернативні реалізації основних компонентів.

4.4 Розробка графічного інтерфейсу користувача

Інтерфейс користувача (англ. user interface, UI) є своєрідним комунікаційним каналом, по якому здійснюється взаємодія користувача і комп'ютера.

Кращий користувацький інтерфейс - це такий інтерфейс, якому користувач не повинен приділяти багато уваги, майже не помічати його. Користувач просто працює, замість того, щоб розмірковувати, яку кнопку натиснути або де клацнути мишею. Такий інтерфейс називають прозорим - користувач як би дивиться крізь нього на свою роботу.

Щоб створити ефективний інтерфейс, який робив би роботу з програмою приємною, потрібно розуміти, які завдання будуть вирішувати користувачі за допомогою даної програми і які вимоги до інтерфейсу можуть виникнути у користувачів. Це зробити набагато легше, якщо ви використовуєте свою програму для власних потреб, адже в даному випадку ви є не тільки розробником, а й користувачем програми, дивіться на неї очима її аудиторії.

Величезну роль відіграє інтуїція - якщо розробник сам терпіти не може некрасиві і незручні інтерфейси, то при створенні власної програми він буде відчувати, де і який саме елемент потрібно прибрати або додати. Необхідно мати художній смак, щоб розуміти, що саме додасть інтерфейсу красу і привабливість.

Західні дослідники в області HCI сформулювали основні принципи проектування користувацьких інтерфейсів комп'ютерних програм. Як і в будь-який інший науці, існує досить багато різних методик і класифікацій, які можна знайти в книгах з HCI, облямованих за кордоном, а також на іноземних Web-сайтах.

Якщо говорити про найбільш загальні принципи проектування користувацьких інтерфейсів, то можна назвати три основні положення:

1. Програма повинна допомагати виконати завдання, а не ставати цим завданням.
2. При роботі з програмою користувач повинен відчувати себе добре.
3. Програма повинна працювати так, щоб користувач не вважав комп'ютер повільним.

Інтерфейс модуля повинен забезпечувати наступні функціональні можливості:

- вибір заводського відтінку автомобільної фарби;
- введення даних (чинників, які впливають на зміну відтінку);
- скинути усі вхідні дані;
- запуск процесу підбору LAB-складових;
- показ візуальних змін відтінку.

– виведення формули LAB-складових.

Для обробки програми основного модулю, а також для вводу в неї даних та їх редагування пропонується застосовувати середовище розробки Intellij Idea.

Сама програма складається із 13 наступних класів: FUNCT, TERM, HIGH, HIGHM, MIDDLE, LOWM, LOW, CYCLE, CYCLE1, CYCLE2, CYCLE3, CYCLE4, FCYCLE, кожен з яких виконує свої функції, що разом роблять можливим правильне функціонування програми.

Всі класи з початку включають бібліотеки для можливості подальшого доступу до функцій, які використовує програма для функціонування. Кожна бібліотека має великий набор функцій, що допомагають програмісту спростити реалізацію того, що він задумав та зменшити клопоти по написанню деяких функцій, які є дуже популярними

Перша кнопка - “Chose basic color”. Вона дозволяю обрати із бази знань і вивести на екран відтінок заводської фарби автомобіля по його коду. Саме з ним і його LAB-складовими програма працюватиме надалі. Кнопка та загальний вигляд користувацького інтерфейсу зображені на рисунку 4.1.

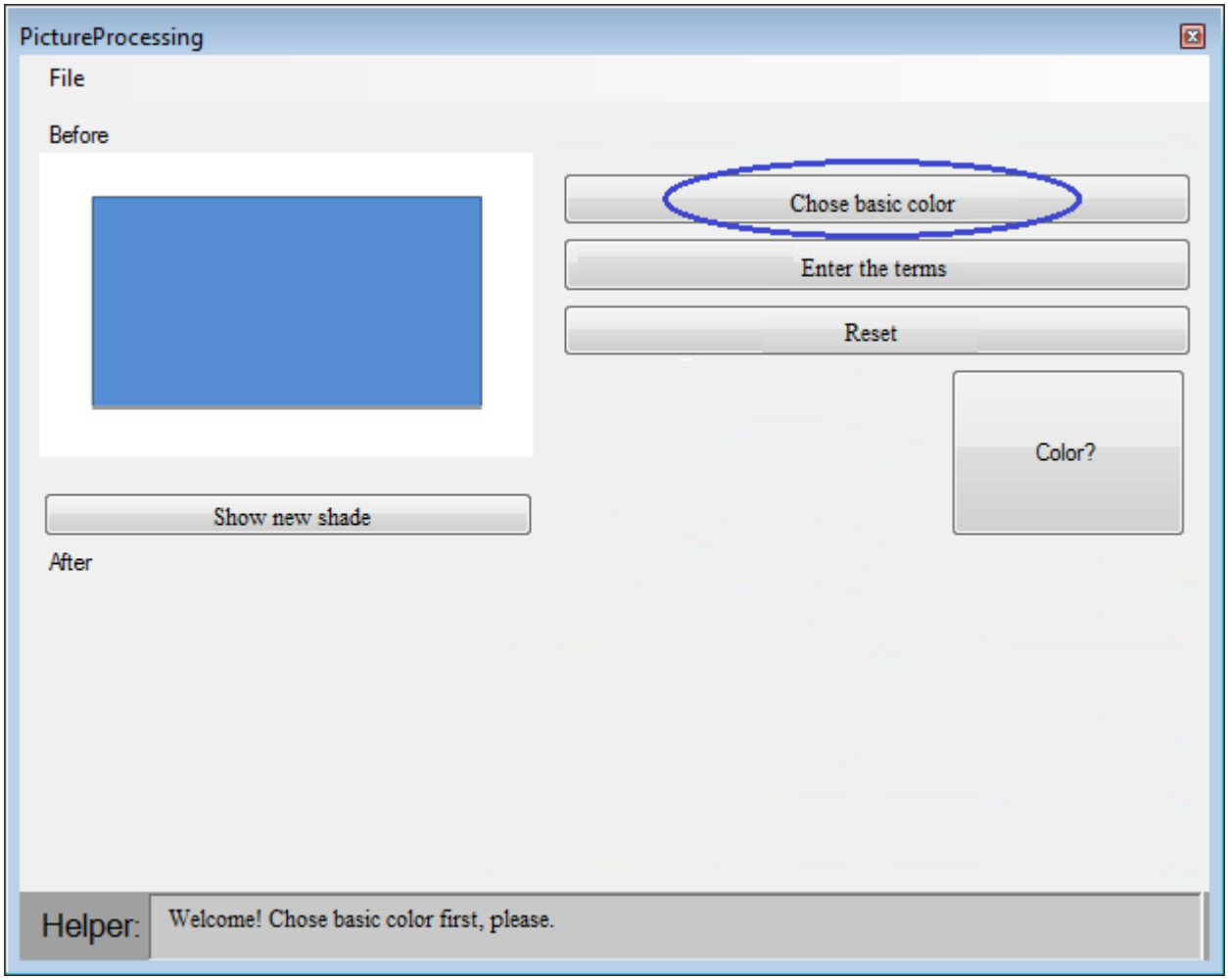


Рисунок 4.1 – Загальний вигляд користувацького інтерфейсу та вибір заводського відтінку

Після того як ми обрали відтінок заводської фарби, нам потрібні дані про чинники, які тим чи іншим чином впливали на зміну відтінку, а саме експлуатаційні умови: клімат, срок експлуатації, стоянка або інакше утримання автомобіля, пробіг автомобіля; умови мийки: частота, використання хімікатів, вид мийки; і, нарешті, покриття доріг. Усе це нам дозволяє зробити кнопка “Enter the terms”. Щоб скинути інформацію про дані, потрібно натиснути кнопку “Reset” (рис. 4.2).

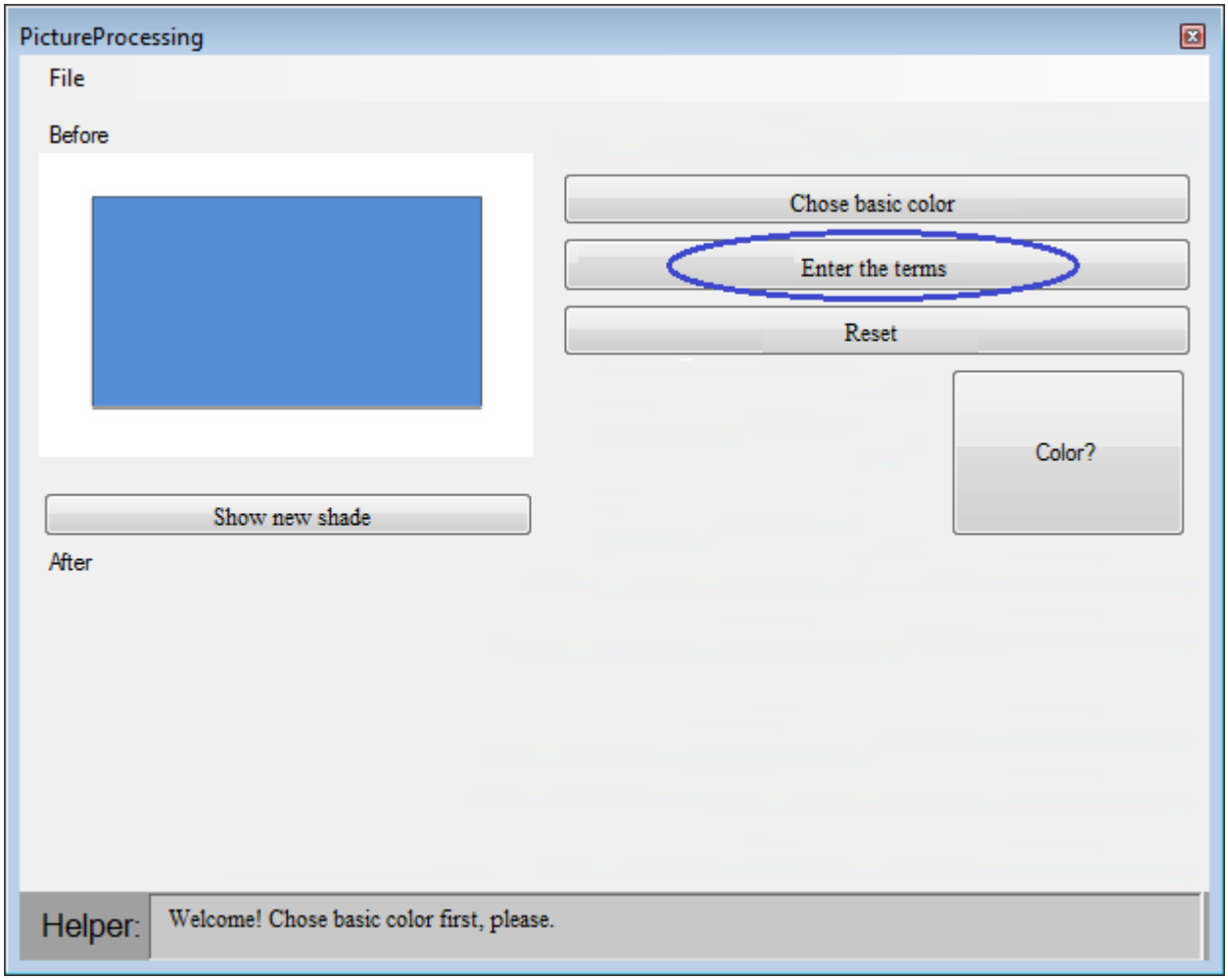


Рисунок 4.2 – Введення даних експлуатації

Тепер після того як програма має усі дані про базовий відтінок заводської фарби та умови, в яких власник користувався автомобілем, натиснувши кнопку “Show new shade”, ми можемо візуально побачити як змінився відтінок (рис. 4.3).

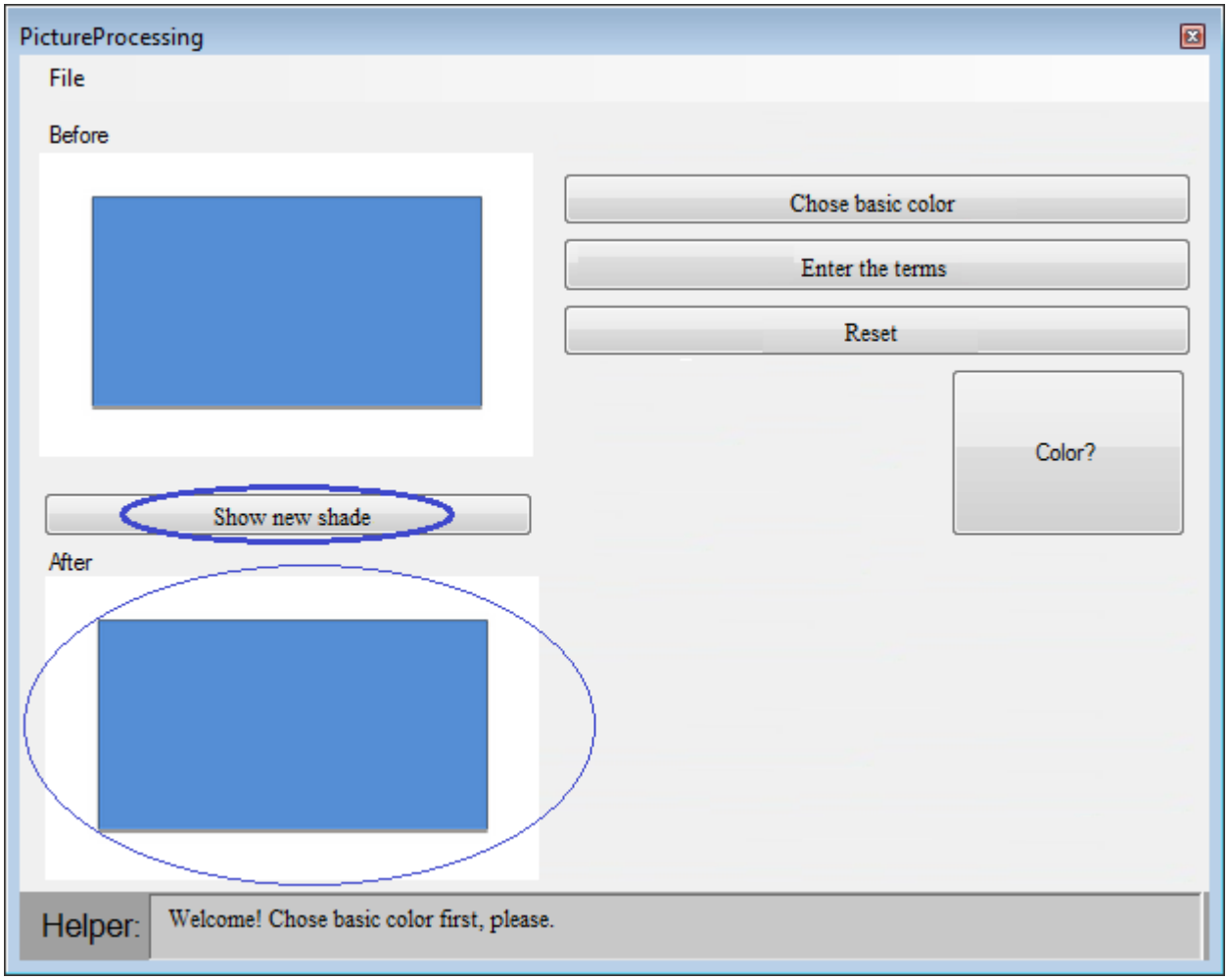


Рисунок 4.3 – Виведення візуальної зміни відтінку

Щоб дізнатись LAB-складову нового відтінку, слід натиснути на кнопку “Color?” (рис. 4.4), яка виводить на экран формулу.

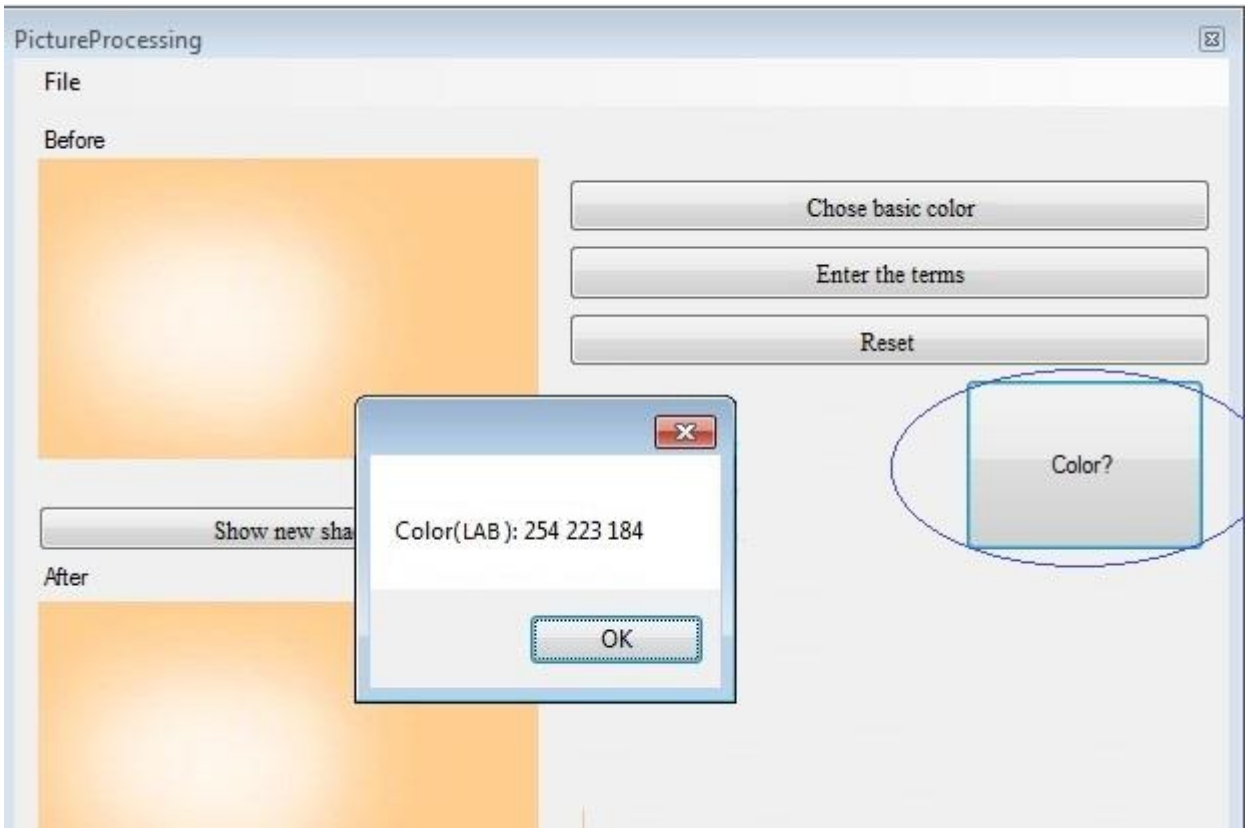


Рисунок 4.4 – Визначення фомули LAB-складових відтінку

4.5 Реалізація та тестування програмного модуля

За допомогою UML згенеровано лістинг програми розпізнавання кольорових відтінків, який розміщено у ДОДАТКУ Б. Інструкція користувача цієї програми розміщена у ДОДАТКУ А.

При цьому інтерфейс головного модуля забезпечує наступні функціональні можливості:

- вибір заводського відтінку автомобільної фарби;
- введення даних (чинників, які впливають на зміну відтінку);
- скинути усі вхідні дані;
- запуск процесу підбору LAB-складових;
- показ візуальних змін відтінку.

- виведення формули LAB-складових.

Остаточно, отримаємо вікно (рис. 4.5), що має наступний вигляд:

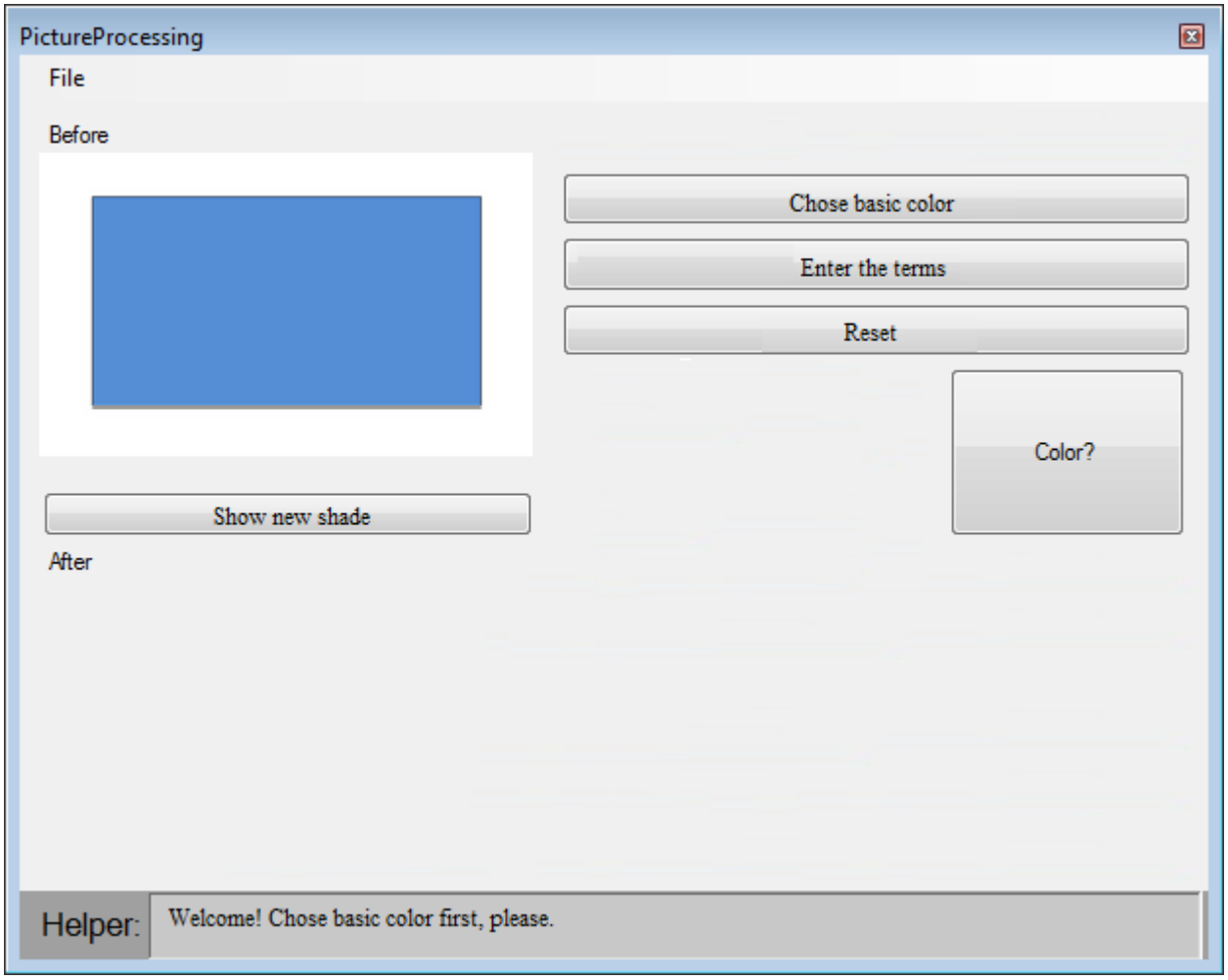


Рисунок 4.5 – Загальний вигляд інтерфейсу користувача

Результати виконання програми наведені на рисунку 4.6.

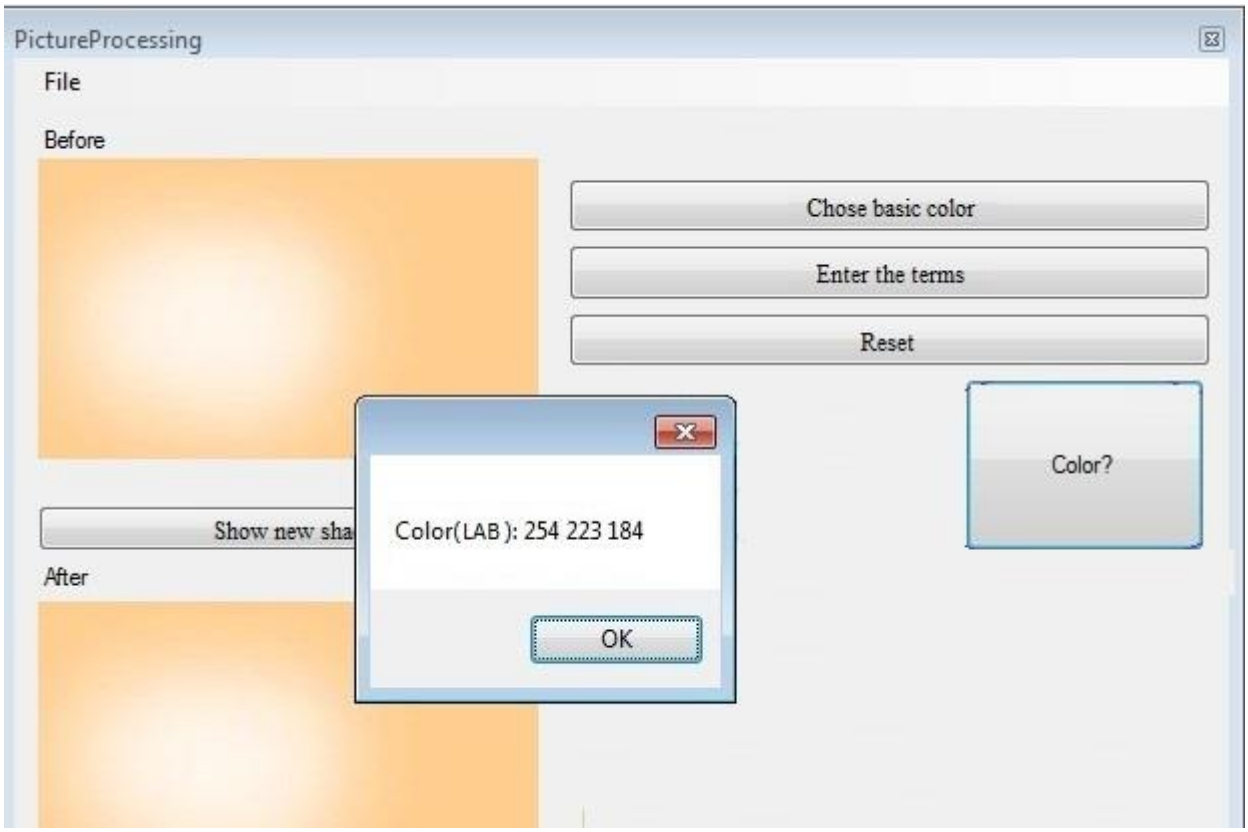


Рисунок 4.6 – Результати виконання програми.

4.6 Висновок

Четвертий розділ магістерської кваліфікаційної роботи присвячений реалізації програмного модуля. Під час його виконання були розглянуті варіанти середовищ, там мов програмування, серед яких були обрані найзручніші. Також було обгрунтовано вибір моделі змішування кольорів. Окрім цього, були описані використані бібліотеки та інші допоміжні програмні засоби. Після цього було реалізовано програмний додаток, який супроводжується графічним інтерфейсом користувача. Під кінець додаток успішно пройшов тестування.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу інформаційної технології прогнозування трансформації автомобільних фарб з використанням методології штучного інтелекту. В результаті оцінювання можна буде зробити висновок щодо напрямів (особливостей) організації подальшого її впровадження з врахуванням встановленого рейтингу.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів: Озеранського В.С., Колесницького О.К., Арсенюка І.Р. Здійснюємо оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати або навчати фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси. Джерела фінансування ідеї відсутні	Потрібні незначні фін. ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у ВПК	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший 5 р. Термін окупності інвестицій більше 10 р.	Термін реалізації ідеї від 3- 5 років. Термін окупності інвестицій більше 5 р.	Термін реалізації ідеї менше 3-х років. Термін окупнос-ті інвестицій від 3-х до5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Колесницький О.К.	Озеранський В.С.	Арсенюк І.Р.
	Бали, виставлені експертами:		
1	2	2	2
2	2	2	1
3	3	4	4
4	3	3	4
5	2	3	1
6	4	3	4
7	2	2	1
8	3	4	4
9	2	3	3
10	4	3	4
11	3	4	4
12	3	2	3
Сума балів	$СБ_1$=33	$СБ_2$=35	$СБ_3$=35
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ}$=34,33≈34		

Отже, з отриманих даних таблиці 5.2 видно, що нова розробка має вище середнього рівень комерційного потенціалу.

5.2 Прогноз попиту на програму розпізнавання відтінків кольорів

Розроблювана інформаційна технологія прогнозування трансформації автомобильних фарб та програма, побудована на цій основі потрібна для ідентифіації кольорового відтінку фарби автомобіля, в залежності від умов, в яких він експлоатувався. Вона може використовуватись як експертна система для оцінки кольорового відтінку, порівняння їх с другими, та вирішеннями деяких інших питань в колористиці. Тому сфера її використання є досить широкою Дану програму можуть використовувати як пересічні користувачі, так і підприємства та організації для зберігання потрібної інформації.

Визначимо потенційну кількість споживачів, яким потрібно вирішувати зазначені задачі. Для цього, опрацювавши статистичні дані, приймемо:

- середня кількість споживачів, які використовують програми, аналогічні розроблюваній П = 22000 споживачів;
- середній відсоток споживачів, які зацікавляться придбанням інноваційного продукту Пп = 80%;
- середній термін заміни інноваційного продукту Т = 2 роки;
- середній відсоток споживачів, що захочуть придбати розроблювану програму повторно Сп = 5%.

Посилаючись на прогнозовані дані, за формулами (5.1) – (5.4) розрахуємо наступні показники:

- потребу в інноваційному продукті

$$Пі=\frac{П \cdot Пп\%}{100\%}, \quad (5.1)$$

$$П_і = \frac{22000 * 80}{100} = 17600 \text{ (шт.)}$$

- оптимістичний прогноз попиту на інноваційне рішення

$$ОП=\frac{Пі}{Т}, \quad (5.2)$$

$$ОП=17600:2=8800 \text{ шт.}$$

- песимістичний прогноз попиту на інноваційне рішення

$$ПП=\frac{ОП \cdot Сп}{100\%}, \quad (5.3)$$

$$ПП=8800*0,05=440 \text{ шт.}$$

- реалістичний прогноз попиту на інноваційне рішення

$$РП=\frac{ОП+ПП}{2}, \quad (5.4)$$

$$РП= (8800+440) /2= 4620 \text{ (шт.)}.\approx 4600 \text{ шт.}$$

Таким чином, приймемо, що потенційний попит на розробку становить 4600 одиниць програмного забезпечення, які можуть бути реалізовані протягом першого року виходу на ринок інноваційного програмного продукту.

Найбільший попит на програмне забезпечення слід очікувати тоді, коли більшість споживачів зрозуміють переваги, які притаманні даній програмі, зручність її використання та здатність самонавчатися, чим вона вигідно вирізняється серед аналогів.

5.3 Вибір каналів збуту та післяпродажного обслуговування

Канали розподілу (збуту) — це сукупність фірм чи окремих осіб, які виконують посередницькі функції щодо фізичного переміщення товарів і перебирають на себе або сприяють переданню права власності на товари на шляху їх просування від виробника до споживача.

Зважаючи, що інноваційний програмний продукт буде розповсюджуватись через мережу Інтернет або шляхом безпосереднього контакту потенційних споживачів з розробниками, найбільш оптимальним каналом збуту є канал нульового рівня:

ВИРОБНИК (РОЗРОБНИК)→СПОЖИВАЧ.

Післяпродажного обслуговування для пересічних користувачів не передбачено, а для підприємств та організацій розробником пропонуються послуги по встановленню програми та навчанню персоналу користування нею.

5.4 Виявлення основних конкурентів і опис їх товарів

Основними конкурентами можна вважати виробників, які пропонують на ринку аналогічні методи, програми чи пристрої для розпізнавання відтінків кольорів. Базою є ручний підбір у автосалоні(3000 грн).

Близькими аналогами даної розробки є мобільний додаток Колориметр(вартість 2000 грн), та APDS-9960 - датчик, розроблений компаніэю Avago(вартість 5000 грн).

Колориметр – мобільний додаток, що дозволяє за допомогою камери ідентифікувати кольоровий відтінок певного предмета. Колориметр діє просто. Запускаєте програму, робите фото і водите по ньому пальцем. Віртуальне збільшувальне скло аналізує колір в точці дотику і видає його назву та склад кольорової моделі. Великим недоліком являється залежність від якості камери вашого смартфона. Чим якісніші будут знімки, тим точніше будет результат.

APDS-9960 - датчик, розроблений компаніэю Avago. Це комбінований цифровий датчик з цілим рядом різних цікавих і корисних функцій. Він вміє розпізнавати жести, визначати наближення, а ще він вміє реєструвати інтенсивність навколишнього освітлення і визначати колір [4].

Розпізнавання кольору і рівень зовнішньої освітленості (Color / ALS)

Згідно функціональної схемою, датчик визначає колір / рівень освітленості за допомогою відповідних фотодіодів. Також заявлено, що APDS-9960 має вбудовані фільтри, що блокують ультрафіолетове і інфрачервоне діапазони.

Спрощено це виглядає так: зареєстровані фотодіодами сигнали вимірюються за допомогою АЦП, заносяться в буфер і потім дані відправляються на обробку. Недоліком являється висока вартість приладу та складність у використанні. Прилад потребую певних навичок та вмінь.

5.5 Вибір методу ціноутворення

Ціна відіграє центральну роль у системі ринкового механізму і є інструментом, який функціонує тільки на основі економічних законів.

Прогноз цінової політики полягає в тому, щоб встановити ціну на свій товар і змінювати її в залежності від ситуації на ринку, отримати намічений обсяг прибутку, відповісти на діяльність конкурентів тощо.

Ціноутворення слід базувати на основі цін конкурентів і встановити початкову ціну на розроблений програмний продукт на ріні цін продуктів конкурентів.

Далі, аналізуючи ситуацію на ринку, при можливості можна знизити її, тим самим залучаючи потенційних клієнтів.

5.6 Оцінка рівня якості та конкурентоспроможності інноваційної програми розпізнавання відтінків кольорів

Оцінка рівня якості інноваційного рішення проводиться з метою порівняльного аналізу і визначення найбільш ефективного в технічному відношенні варіанта рішення. Визначимо абсолютний та відносний рівні якості розроблюваної програми прогнозування трансформації автомобільних фарб.

Таблиця 5.3 - Основні технічні показники нової розробки

Параметри	Абсолютне значення параметра			Вагомість параметра
	краще	середнє	гірше	
Періодичність налаштування експертної бази знань	9			40%
Автоматизація прийняття рішення		8		20%
Час прийняття рішення			4	15%
Ергономічність		6		10%
Вихід ODBC	9			15%

Визначимо абсолютний рівень інноваційного рішення за формулою:

$$К_{я.а.}=\sum_{i=1}^{n} Р_{Нi} \cdot \alpha_i , \qquad (5.5)$$

де $Р_{Нi}$ – числове значення *i*-го параметру інноваційного рішення;

n – кількість параметрів інноваційного рішення, що прийняті для оцінки;

α_i – коефіцієнт вагомості відповідного параметра.

$$К_{я.а.}=9*0,4+8*0,2+4*0,15+6*0,1+9*0,15=7,75$$

Далі визначимо відносний рівень якості окремих параметрів інноваційного рішення, порівнюючи його показники з абсолютними показниками якості аналогу.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.6) та (5.7) і занесемо їх у відповідну колонку табл. 5.4.

$$q_i = \frac{P_{Hi}}{P_{Бi}}, \tag{5.6}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}}, \tag{5.7}$$

де P_{Hi}, $P_{Бi}$ – числові значення *i*-го параметру відповідно нового і базового виробів.

Таблиця 5.4 – Основні технічні й економічні параметри інноваційного рішення та товару-конкурента

Параметр	Варіанти		Відносний показник якості q_i	Коефіцієнт вагомості, α_i
	Ручний підбір автосалону	Інноваційне рішення		
Періодичність налаштування експертної бази знань	Не потребуєтьс я	1раз на 4роки	4	0,4
Автоматизація прийняття рішення	часткова	повна	1,5	0,2
Час прийняття рішення	Декілька днів	0,5с	0,2	0,15
Ергономічність	Відсутня	Ергономічний інтерфейс	0,7	0,1
Вихід ODBC	Відсутній	Наявний	1,5	0,15
Ціна	3000	4000	-	-

Відносний рівень якості інноваційного рішення визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \tag{5.8}$$

$К_{я.в.}$=4*0,4+1,5*0,2+0,2*0,15+0,7*0,1+1,5*0,15= 2,23

Відносний коефіцієнт показника якості інноваційного рішення більший одиниці, це означає , що інноваційний продукт якісніший базового товару-конкурента у 1,2 рази.

Під конкурентоспроможністю продукції прийнято розуміти сукупність її властивостей, що відображає ступінь задоволення конкретної потреби споживачів проти представленої на ринку аналогічної продукції. Вона визначає здатність конкурувати на ринку з продукцією фірм-конкурентів, тобто мати технічні або економічні переваги над виробами інших виробників.

При визначенні конкурентоспроможності програмного продукту розглядають властивості розроблюваної програми і програм конкурентів за технічними та економічними параметрами.

Загальний показник конкурентоспроможності інноваційного рішення (К) можна визначити за формулою:

$$К = \frac{I_{т.п.}}{I_{е.п.}}, \quad (5.9)$$

де $I_{т.п.}$ — індекс технічних параметрів (відносний рівень якості інноваційного рішення);

$I_{е.п.}$ – індекс економічних параметрів.

Індекс економічних параметрів визначається за формулою:

$$I_{е.п.} = \frac{\sum_{i=1}^{n} P_{Нei}}{\sum_{i=1}^{n} P_{Бei}}, \quad (5.10)$$

де $P_{Нei}$, $P_{Бei}$ – економічні параметри відповідно нового та базового товарів.

Якщо $K>1$, то інноваційне рішення вважається більш конкурентоспроможним, ніж товар-конкурент; якщо $K<1$, то рівень конкурентоспроможності інноваційного рішення є нижчим, ніж у товара-конкурента; якщо $K=1$, то ця ситуація інтерпретується як тотожність рівнів конкурентоспроможності обох товарів.

Оскільки індекс технічних параметрів дорівнює відносному рівню якості нашого інноваційного продукту, то він буде рівним 2,23. За формулою 5.10 розрахуємо індекс економічних параметрів інноваційного рішення:

$$I_{е.п.} = \frac{4000}{3000} = 1{,}33$$

Тоді, користуючись формулою 5.9, розрахуємо загальний показник конкурентоспроможності:

$$К = \frac{2{,}23}{1{,}33} = 1{,}68$$

Оскільки К>1 (К=1,68), то розроблювана програма розпізнавання відтінків кольорів є більш конкурентоспроможною, ніж найкращий аналог, обраний за базу порівняння – додаток Колориметр.

5.7 Прогнозування витрат на виконання науково-дослідної роботи

Для створення програми розпізнавання відтінків кольорів необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.11):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (5.11)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробників.

Розрахунки заробітних плат для керівника й провідного спеціаліста наведені в таблиці 5.5.

Таблиця 5.5 – Розрахунок основної заробітної плати розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	8500	386,4	245	94658,9
Провідний спеціаліст	7000	318,18	132	42000,0
Всього				136658,9

Основна заробітна плата робітників $З_р$, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_р = \sum_{1}^{т} t_i \cdot C_i \text{ , грн.} \qquad (5.12)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин;

n – число робіт по видах та розрядах;

C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{М_м \cdot К_і}{Т_р \cdot Т_{зм}}, \qquad \text{грн/годину} \qquad (5.13)$$

де $М_м$ – розмір мінімальної заробітної плати за місяць, *грн*.; у 2020 році мінімальна заробітна плата становить –3200грн.

$К_і$ – тарифний коефіцієнт робітника відповідного розряду. $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = 21...23 дні;

$Т_{зм}$ – тривалість зміни, зазвичай $Т_{зм}$ = 8 годин.

Мінімальна погодинна ставка робітника 1-го розряду з 1.01.2020 року встановлена на рівні –19,34 грн/год.

Розрахунок основної заробітної плати для робітників, які брали участь у проекті зведений до таблиці 5.6.

Таблиця 5.6 – Заробітна плата робітників, які приймали участі у проекті

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина оплати, грн.
Формулювання вимог та розробка ТЗ	120	4	24,56	1,27	2947,2
Розробка структури та діаграми класів	240	6	28,043	1,45	6730,32
Розробка алгоритмів роботи	240	6	28,043	1,45	6730,32
Реалізація програмного забезпечення	240	6	28,043	1,45	6730,32
Тестування програмного забезпечення	80	5	26,3	1,36	2104,0
Ліквідація багів та покращення коду	120	6	28,043	1,45	3365,16
Налаштування експертної бази знань	660	5	26,3	1,36	17358,0
Оформлення документації	240	4	24,56	1,27	5894,4
Разом	1960				51859,72

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10... 12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = (0{,}1 \dots 0{,}12) \cdot (З_о + З_р), \text{грн.} \quad (5.14)$$

$З_д$ = (136658,9+51859,72)·0,12=22622,23 *грн*.

Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = (З_о + З_р + З_д) \cdot \frac{\beta}{100}, \text{ грн.} \quad (5.15)$$

де $З_о$ – основна заробітна плата розробників, *грн*.;

$З_р$– основна заробітна плата робітників, *грн*.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, *грн*.;

β – ставка єдиного соціального внеску, %. ЄСВ –22%.

$$H_{зп} = (136658,9+51859,72+22622,23)\cdot 0,22 = 46450,99\ грн.$$

Амортизацію обладнання, комп'ютерів та приміщень A, які використовувались під час (чи для) виконання даного етапу роботи розраховують по кожному виду обладнання, приміщенням тощо:

$$A_{обл} = \frac{Ц}{T_{кор.}} \cdot \frac{T_{фак.}}{12}, \quad (5.16)$$

де $Ц$ – балансова вартість обладнання, приміщень тощо, які використовувались для розробки нового технічного рішення без ПДВ, грн.;

$T_{фак.}$ – термін фактичного використання обладнання, приміщень під час розробки, місяців;

$T_{кор.}$ – термін використання обладнання, приміщень згідно діючого ПКУ, років.

Таблиця 5.7 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Термін корисного використання, роки	Термін фактичного використання, міс.	Величина амортизаційних відрахувань, грн
Комп'ютер	9500	2	11	4354,2
Пристрій безперебійного живлення	2500	2	11	1145,8
Принтер	3500	2	2	291,7
Роутер	500	2	11	229,2
Меблі	15000	5	12	3000,0
Приміщення	160000	25	12	6400,0
Всього				15420,9

Розрахуємо витрати на допоміжні матеріали за формулою:

$$ДМ = \sum_1^n N_i Ц_i К_i \quad (5.17)$$

де n – кількість допоміжних матеріалів;

Ni - кількість допоміжних матеріалів і-го виду;

Ці – покупна ціна допоміжних матеріалів і-го виду, грн;

Кі – коефіцієнт транспортних витрат (Кі = 1,0).

Таблиця 5.8 - Витрати на допоміжні матеріали, що були використані для розробки ПЗ

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість комплектуючі, грн.
Папір	уп.	150	3	450
Послуги Інтернет-провайдера	міс.	100	11	1100
Всього з урахуванням транспортних витрат				1550,0

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п \quad (5.18)$$

де В – вартість 1кВт-години електроенергії (В=1,83 грн/кВт);

П – установлена потужність обладнання (П=0,9кВт);

Ф – фактична кількість годин роботи обладнання (Ф=1936 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,8$).

$$В_е= 1,83 \cdot 0,9 \cdot 1936 \cdot 0,8 = 2550,9 \text{ (грн.)}$$

Інші витрати $В_{ін}$.можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.19)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1,8 * (136658,9 + 51859,72) = 339333,5 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_р + З_д + Н_{зп} + А + ДМ + В_е + І_в \quad (5.20)$$

$$В = 136658,9 + 51859,72 + 22622,23 + 46450,99 + 15420,9 + 1550,0 + +2550,9 + 339333,5 = 616447,2 \text{ грн.}$$

На другому етапі розрахуємо величину загальних витрат на виконання розробки програмного продукту. Загальна вартість розробки розраховується за формулою:

$$B_{заг} = \frac{B_{інш}}{\alpha}, \quad (5.21)$$

де α — частка витрат, які безпосередньо здійснює виконавець розробки. Підставивши значення у формулу (4.21) отримаємо:

$$B_{заг} = \frac{616447,2}{0,8} = 770596,5 \text{ грн.}$$

На третьому етапі здійснюється прогнозування загальних витрат на виконання та впровадження результатів розробки. Прогнозування загальних витрат на виконання та впровадження результатів виконаної дослідної роботи здійснюється за формулою:

$$ЗВ = \frac{B_{заг}}{\beta}, \quad (5.22)$$

де β — коефіцієнт, який характеризує етап виконання дослідної роботи. В нашому випадку 0,9. Підставивши значення у формулу (5.22) отримаємо:

$$ЗВ = \frac{770596,5}{0,9} = 856218,3 \text{ грн.}$$

5.8 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (5.23)$$

де $ПП$ – приведена вартість всіх чистих прибутків, що їх отримає розробник від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій; $PV = ЗВ$, грн.

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.24)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$П = \frac{1366612,0}{(1+0,1)^1} + \frac{1093289,6}{(1+0,1)^2} + \frac{683306,0}{(1+0,1)^3} = 2659297,6 \text{ грн.}$$

Тоді розрахуємо $Е_{абс}$:

$$Е_{абс} = 2659297,6 - 856218,3 = 1803079,3 \; грн.$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Рух платежів (інвестицій та прибутків) відображено на рисунку 5.1.

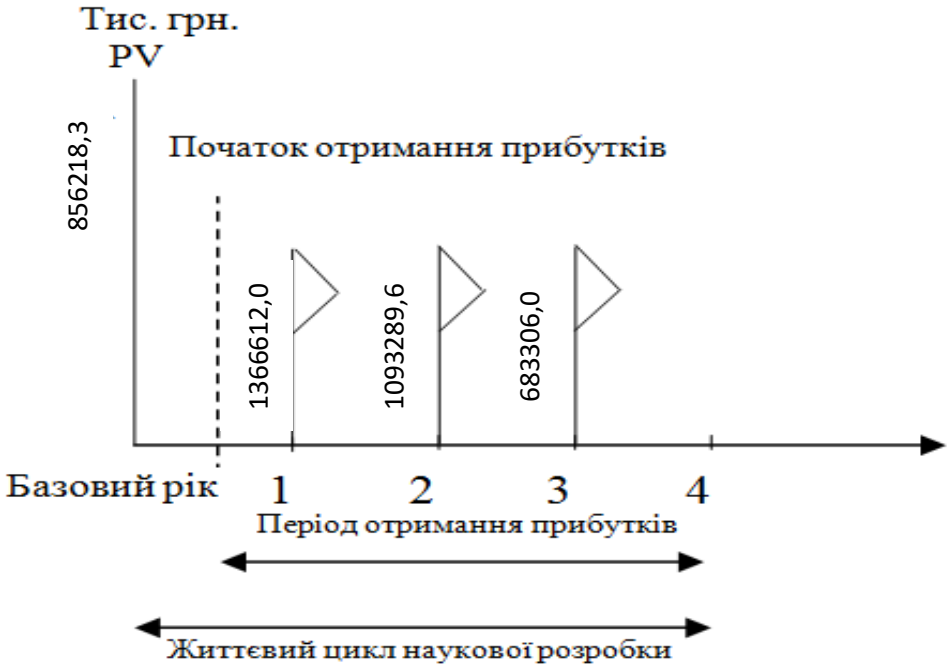


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (5.25)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{1803079,3}{856218,3}} - 1 = 1,459 - 1 = 0,459 \text{ або } 45,9\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \tag{5.26}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2017 році в Україні d = 0,15;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}15 + 0{,}1 = 0{,}25$$

Оскільки $E_в$ = 45,9 % > $\tau_{мін}$ = 0,25 = 25%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.27}$$

$$T_{ок} = \frac{1}{0{,}459} = 2{,}18 \text{ років}$$

Визначивши термін окупності даної наукової розробки, можна зробити висновок, що її фінансування буде доцільним, оскільки нормальним значенням терміну окупності інвестицій вважається 3-5 років, а отриманне значення є меншим за нормативне.

5.9 Висновок

В даному розділі було здійснено економічне обґрунтування доцільності розробки інформаційної технології прогнозування трансформації автомобільних фарб та її програмної реалізації.

За результатами оцінювання комерційного потенціалу розробки визначено її вище середнього рівень комерційного потенціалу, оскільки сума балів становить 34.

Також було виконано розрахунки якості та конкурентоспроможності розроблюваного програмного продукту порівняно з найкращим аналогом – Колориметром, ринкова вартість якого 3000 грн. Встановлено, що інноваційна програма є кращою за аналог, оскільки за показниками якості вона перевищує результати ручного підбору майже у 1,2 рази, а за показником конкурентоспроможності на 68 %.

Було виконано прогнозування витрат на виконання розробки інформаційної технології розпізнавання відтінків кольорів, де розраховано основну заробітну плату кожного із розробників та робітників, які приймали участь у підготовці проекту, додаткову заробітну плату, нарахування на заробітну плату, амортизацію обладнання, комп’ютерів та приміщень, витрати на допоміжні матеріали, витрати на електроенергію. Загальна сума витрат на розробку , за попередніми розрахунками становить 616447,2 грн.

Також було здійснено прогнозування витрат на виконання та впровадження результатів науково-дослідної роботи. Визначено,що для реалізації даного етапу розробки необхідно залучити 856218,3 грн.

Виконано розрахунок ефективності вкладених інвестицій та періоду їх окупності. Абсолютна ефективність вкладених інвестицій складає 1803079,3 грн, і як висновок вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним. Було розраховано відносну (щорічну) ефективність вкладених в наукову розробку інвестицій, її величина 45,9% більша за мінімальну (бар'єрну) ставку дисконтування 0,25, отже інвестор буде зацікавлений у фінансуванні даної наукової розробки.

Проведено розрахунок терміну окупності, який складає 2,18 роки, що менше нормального його значення 3-5 років. Це свідчить про економічну доцільність фінансування розробки інформаційної технології розпізнавання відтінків кольорів.

ВИСНОВКИ

Всі задачі, поставлені в магістерській кваліфікаційній роботі виконано в повному об’ємі, а саме:

– проаналізувати існуючі технології, методи і моделі прогнозування трансформації автомобільних фарб;

– сформулювати вимоги до роботи технології прогнозування трансформації автомобільних фарб.

– розробити математичну модель і технологію прогнозування трансформації автомобільних фарб на основі нечіткої логіки.

– спроектувати програмний додаток для прогнозування трансформації автомобільних фарб;

– реалізувати програмний додаток для прогнозування трансформації автомобільних фарб;

виконати завдання економічної частини;

Поставлена мета - зменшення трудозатрат в процесі прогнозування трансформаціхї автомобільних фарб досягнута за рахунок повної автоматизації процесу визначення складових “ремонтної” фарби.

За думкою экспертів, в порівнянні з існуючим ручним процесом підбору ремонтних автомобільних фарб, застосування удосконаленої технології прогнозування дозволяє зменшити трудовитрати приблизно втричі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Озеранський В.С., Сілагін Є.О. Підхід до створення програми підбору автомобільних фарб. Тези доповіді регіональної НТК “ВНТУ ІНТКІ-2019”. Тези доступні у репозитарії ВНТУ.
2. Озеранський В.С., Сілагін Є.О. Нечіткий підхід до вирішення задачі ідентифікації автомобільних фарб. Тези доповіді міжнародної науково-практичної конференції “ІОН-2020”. Тези доступні у репозитарії ВНТУ. – https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/30898/WORK-IES-2020-61-63.pdf?secuence=1&isAllowed=y
3. Бондарев В.Н. Искусственный интеллект. Учебное пособие для вузов.В.Н. Бондарев, Ф.Г. Аде – Севастополь, СевНТУ, 2002. – 615 с.
4. Іванюк В.Г., Капшій О.В., Косаревич Р.Я., Лау Г. Інформаційна оцінка і виділення фрагментів кольорових зображень // Радиоэлектроника и информатика, № 3(28), 2004. – С. 122-125.
5. Путятин Е.П. Нормализация и распознавание изображений [Электронный ресурс]. – Режим доступа: http://sumschool.sumdu.edu.ua/is-02/rus/lectures/pytyatin/pytyatin.htm
6. Антон Элиенс. Принципы объектно-ориентированной разработки программ. 2-е издание. – М.: Издательский дом “Вильямс”, 2002. – 496 с.
7. Митюшкин Ю. И., Мокин Б. И., Ротштейн А. П. “Soft Computing: идентификация закономерностей нечеткими базами знаний” – Вінниця: УНІВЕРСУМ-Вінниця, 2002 – 145с.
8. Ротштейн О.П. Інтелектуальні технології ідентифікації: нечіткі множини, генетичні алгоритми, нейронні мережі. – Вінниця: Універсум – Вінниця, 1999. – 320с., іл.
9. Арсенюк І.Р.,Кукунін С.В.,Сілагін О.В. Застосування апарату нечіткої логіки для оцінки якості графічних растрових зображень // «IES-2014» Зб. наук. Праць. – Вінниця ; ВНТУ, 2014 – 321с

10. Русинов М.М. Композиция оптических систем. Л., «Машино-строение», 1989. – 484с.
11. Волосов Д.С. Фотографическая оптика. М., «Искусство», 1971. – 352с.
12. Шапиро Л., СтокманДж. Компьютерное зрение / Пер. с англ. – М.: БИНОМ, 2006. – 752 с.
13. Визильтер Ю.В., Желтов С.Ю., Князь В.А. и др. Обработка и анализ цифрових изображений с примерами на LabVIEW и IMAQ Vision. – М.: ДМК Пресс, 2007. – 464 с.
14. Vladimir Vezhnevets, VassiliSazonov, AllaAndreeva. A Survey on Pixel-Based Skin Color Detection // In Proceedings of the GraphiCon 2003 (2003), pp. 85-92.
15. Kruppa H., Bauer M.A., Schiele B. Skin patch detection in real-world images // In Annual Symposium for Pattern Recognition of the DAGM2002, Springer LNCS2449, 2002. - 109–117.
16. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів магістерської підготовки спеціальності 122 – «Комп'ютерні науки» усіх форм навчання / Уклад. А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2019. – 52 с.

ДОДАТОК А

Інструкція користувача
програмного додатку «Підбір автомобільних фарб»

Перша кнопка - “Chose basic color”. Вона дозволяю обрати із бази знань і вивести на екран відтінок заводської фарби автомобіля по його коду. Саме з ним і його LAB-складовими програма працюватиме надалі. Кнопка та загальний вигляд користувацького інтерфейсу зображені на рисунку А.1.

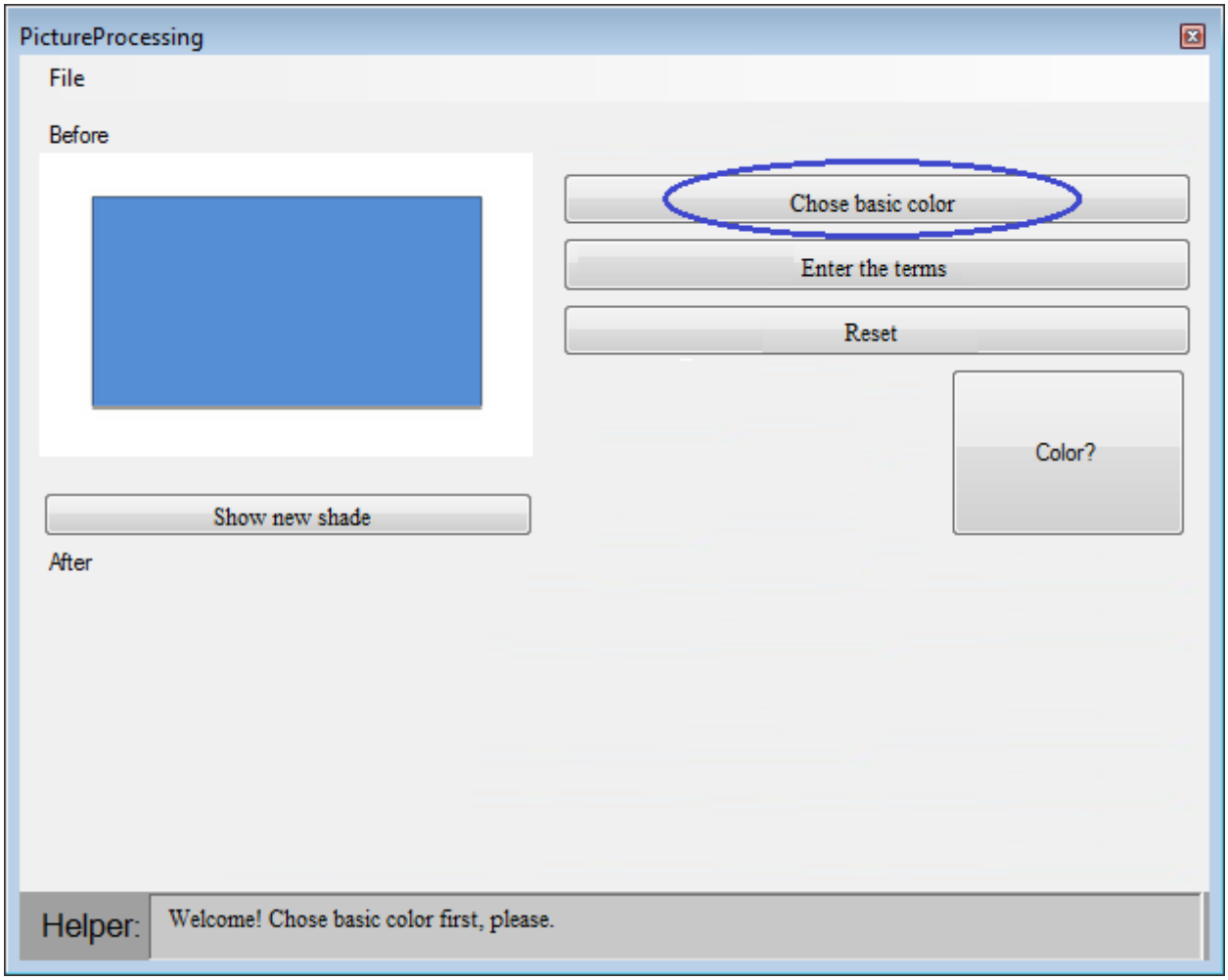


Рисунок А.1 – Загальний вигляд користувацького інтерфейсу та вибір заводського відтінку

Після того як ми обрали відтінок заводської фарби, нам потрібні дані про чинники, які тим чи іншим чином впливали на зміну відтінку, а саме

експлуатаційні умови: клімат, срок експлуатації, стоянка або інакше утримання автомобіля, пробіг автомобіля; умови мийки: частота, використання хімікатів, вид мийки; і, нарешті, покриття доріг. Усе це нам дозволяє зробити кнопка “Enter the terms”. Щоб скинути інформацію про дані, потрібно натиснути кнопку “Reset” (рис. А.2).

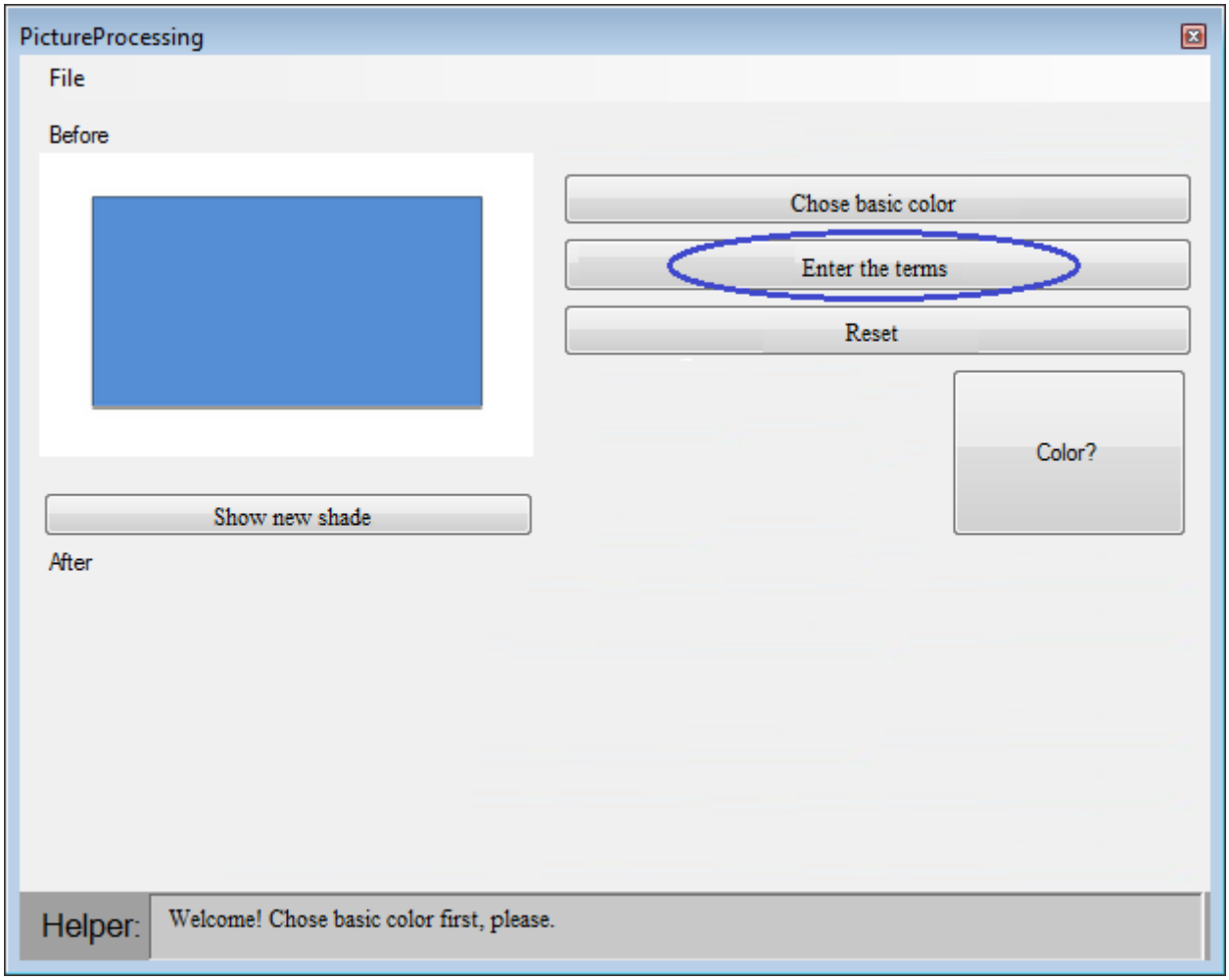


Рисунок А.2 – Введення даних експлуатації

Тепер після того як програма має усі дані про базовий відтінок заводської фарби та умови, в яких власник користувався автомобілем, натиснувши кнопку “Show new shade”, ми можемо візуально побачити як змінився відтінок (рис. А.3).

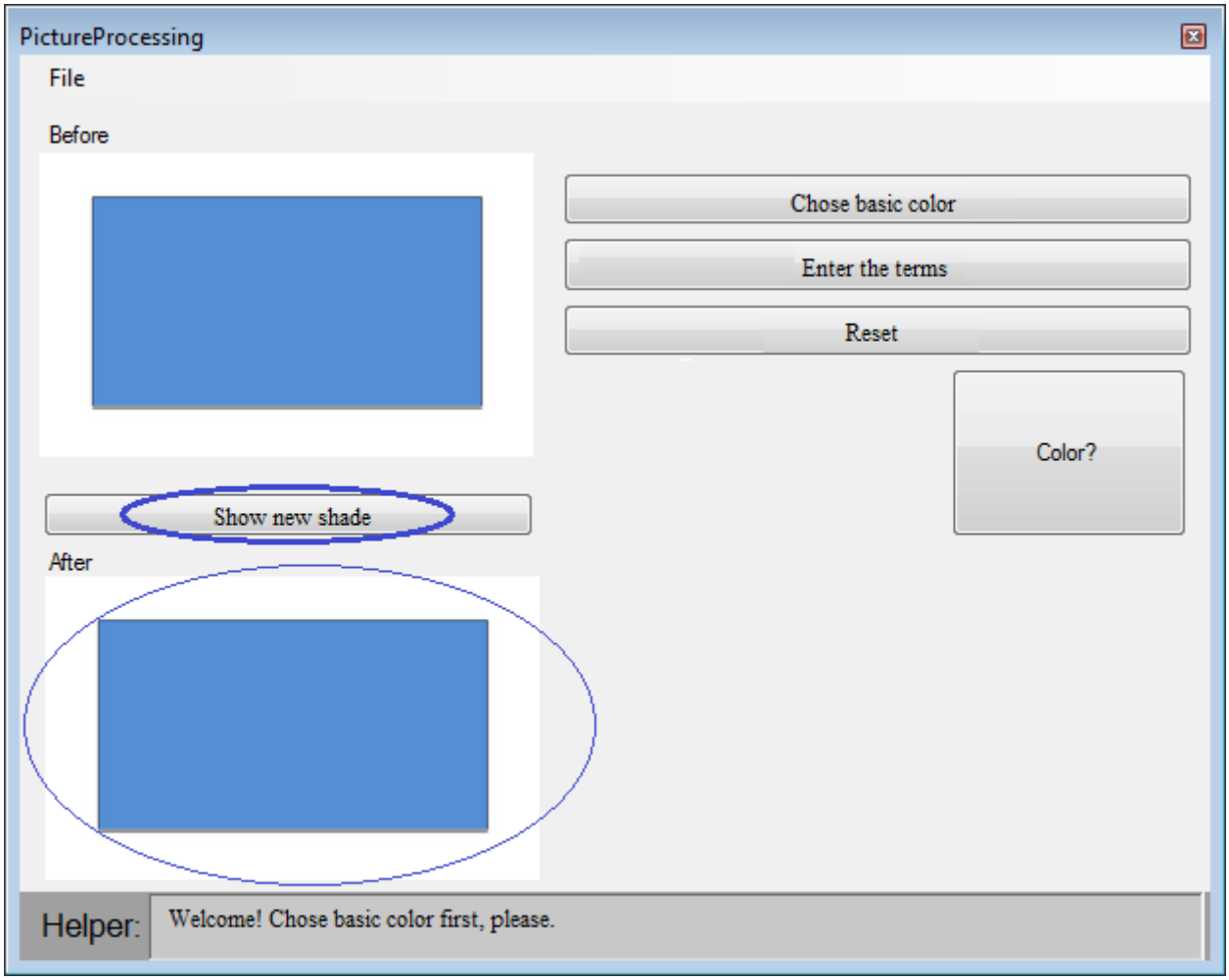


Рисунок А.3 – Виведення візуальної зміни відтінку

Щоб дізнатись LAB-складову нового відтінку, слід натиснути на кнопку “Color?” (рис. А.4), яка виводить на экран формулу.

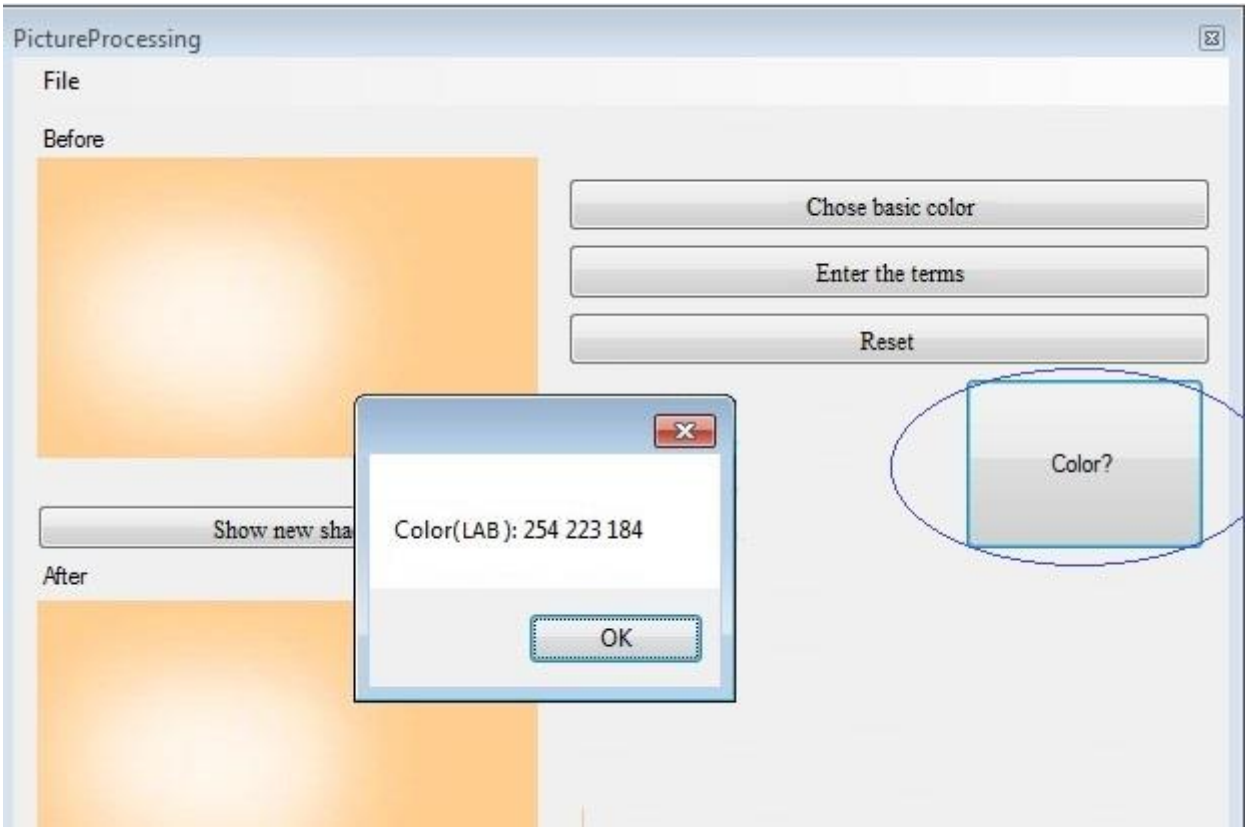


Рисунок А.4 – Визначення фомули LAB-складових відтінку

За допомогою UML згенеровано лістинг програми розпізнавання кольорових відтінків, який розміщено у ДОДАТКУ Г. Інструкція користувача цієї програми розміщена у ДОДАТКУ Б.

При цьому інтерфейс головного модуля забезпечує наступні функціональні можливості:

- вибір заводського відтінку автомобільної фарби;
- введення даних (чинників, які впливають на зміну відтінку);
- скинути усі вхідні дані;
- запуск процесу підбору LAB-складових;
- показ візуальних змін відтінку.
- виведення формули LAB-складових.

Остаточно, отримаємо вікно (рис. А.5), що має наступний вигляд:

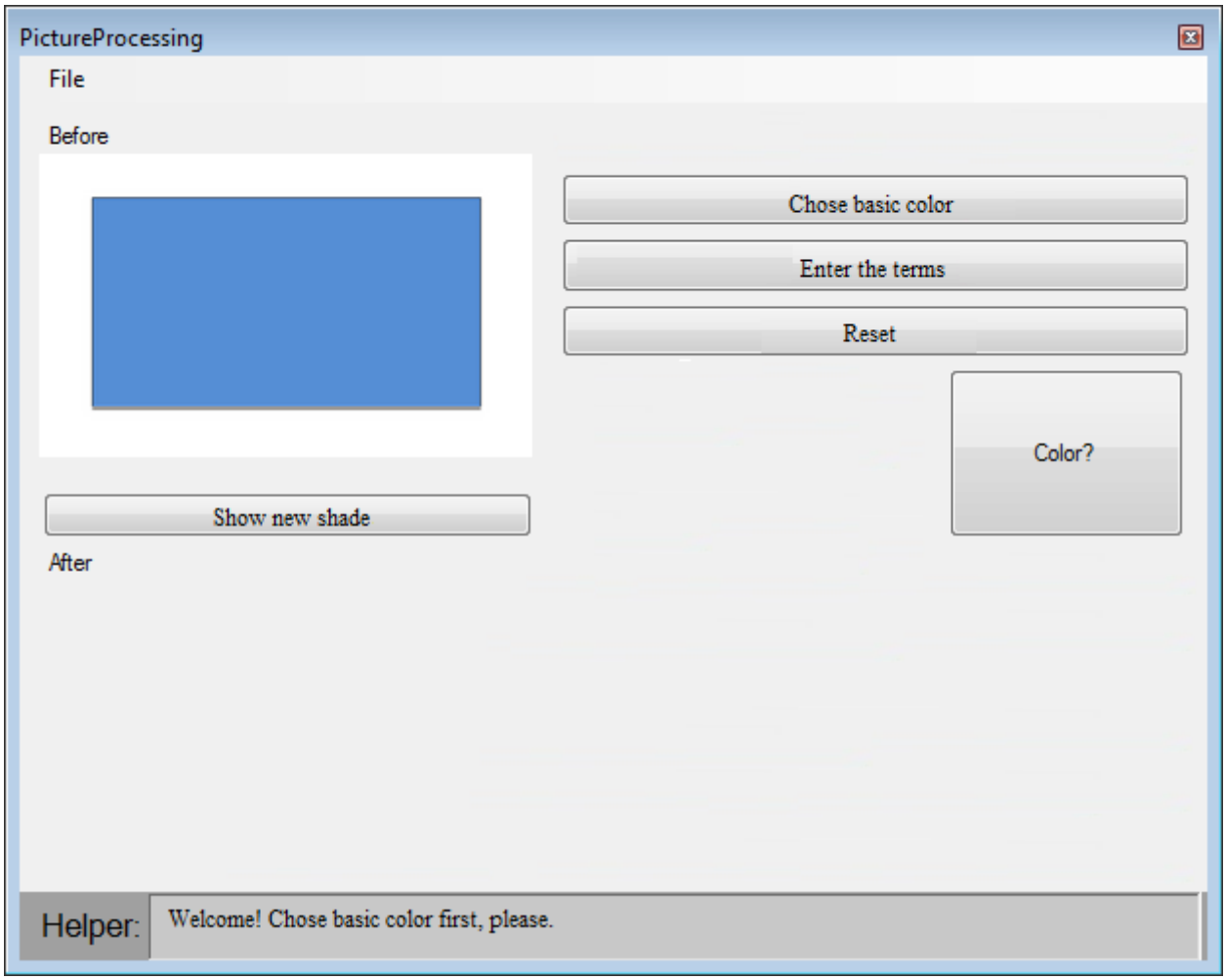


Рисунок А.5 – Загальний вигляд інтерфейсу користувача.

Результати виконання програми наведені на рисунку А.6.

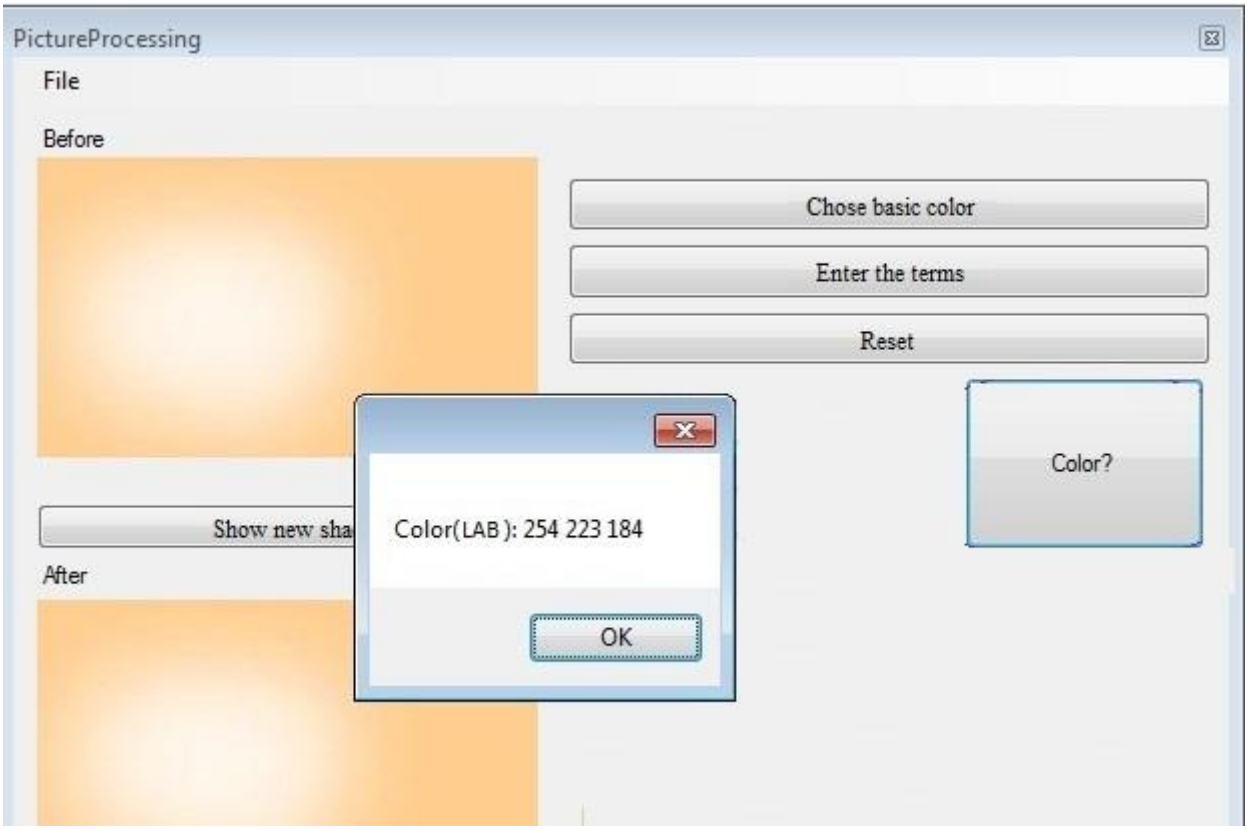


Рисунок А.6 – Результати виконання програми.

ДОДАТОК Б

Лістинг програмного забезпечення

```
packagecom.tuskiomi.image;

publicclassColorVector {

publicstatic final int twoPctLeniency = 328965;
publicstatic final int fivePctLeniency = 855309;
publicstatic final int tenPctLeniency = 1644825;


publicint x, y;
publicint RGB;
publicint leniency;

publicColorVector(int x, int y, int RGB){
this.x = x;
this.y = y;
this.RGB = RGB;
this.leniency = ColorVector.twoPctLeniency;
    }

publicColorVector(int x, int y, int RGB, int leniency){
this.x = x;
this.y = y;
this.RGB = RGB;
this.leniency = leniency;
    }

publicstatic boolean match( ColorVector vec, int rgb){
int r = (rgb >> 16) & 0xFF;
int g = (rgb >> 8) & 0xFF;
int b = (rgb >> 0) & 0xFF;
//System.out.print(rgb);
//System.out.print("-"+vec.RGB);

int er = (vec.RGB >> 16) & 0xFF;
int eg = (vec.RGB >> 8) & 0xFF;
int eb = (vec.RGB >> 0) & 0xFF;

int tr = (vec.leniency >> 16) & 0xFF;
int tg = (vec.leniency >> 8) & 0xFF;
int tb = (vec.leniency >> 0) & 0xFF;
```

```
      int rdiff = Math.abs(r-er);
int gdiff = Math.abs(g-eg);
int bdiff = Math.abs(b-eb);
//System.out.println("-"+rdiff+"-"+gdiff+"-"+bdiff);

return (rdiff<tr)||(gdiff<tg)||(bdiff<tb);
    }

public boolean match(int rgb){
int r = (rgb >> 16) & 0xFF;
int g = (rgb >> 8) & 0xFF;
int b = (rgb >> 0) & 0xFF;

        System.out.print(rgb);
        System.out.println("-"+this.RGB);

int er = (this.RGB >> 16) & 0xFF;
int eg = (this.RGB >> 8) & 0xFF;
int eb = (this.RGB >> 0) & 0xFF;

int tr = (this.leniency >> 16) & 0xFF;
int tg = (this.leniency >> 8) & 0xFF;
int tb = (this.leniency >> 0) & 0xFF;

int rdiff = Math.abs(r-er);
int gdiff = Math.abs(g-eg);
int bdiff = Math.abs(b-eb);

return (rdiff<tr)&&(gdiff<tg)&&(bdiff<tb);
}


}

import java.awt.Color;
import java.util.ArrayList;

/**
 * Java Code to get a color name from rgb/hex value/awt color
 *
 * The part of looking up a color name from the rgb values is edited from
 * https://gist.github.com/nightlark/6482130#file-gistfile1-java (that has some
errors) by Ryan Mast (nightlark)
 *
 *
 *
 */
publicclassColorUtils {

/**
     * Initialize the color list that we have.
     */
private ArrayList<ColorName>initColorList() {
        ArrayList<ColorName> colorList = new ArrayList<ColorName>();
        colorList.add(new ColorName("AliceBlue", 0xF0, 0xF8, 0xFF));
        colorList.add(new ColorName("AntiqueWhite", 0xFA, 0xEB, 0xD7));
        colorList.add(new ColorName("Aqua", 0x00, 0xFF, 0xFF));
        colorList.add(new ColorName("Aquamarine", 0x7F, 0xFF, 0xD4));
        colorList.add(new ColorName("Azure", 0xF0, 0xFF, 0xFF));
```

```
        colorList.add(new ColorName("Beige", 0xF5, 0xF5, 0xDC));

colorList.add(new ColorName("Bisque", 0xFF, 0xE4, 0xC4));
        colorList.add(new ColorName("Black", 0x00, 0x00, 0x00));
        colorList.add(new ColorName("BlanchedAlmond", 0xFF, 0xEB, 0xCD));
        colorList.add(new ColorName("Blue", 0x00, 0x00, 0xFF));
        colorList.add(new ColorName("BlueViolet", 0x8A, 0x2B, 0xE2));
        colorList.add(new ColorName("Brown", 0xA5, 0x2A, 0x2A));
        colorList.add(new ColorName("BurlyWood", 0xDE, 0xB8, 0x87));
        colorList.add(new ColorName("CadetBlue", 0x5F, 0x9E, 0xA0));
        colorList.add(new ColorName("Chartreuse", 0x7F, 0xFF, 0x00));
        colorList.add(new ColorName("Chocolate", 0xD2, 0x69, 0x1E));
        colorList.add(new ColorName("Coral", 0xFF, 0x7F, 0x50));
        colorList.add(new ColorName("CornflowerBlue", 0x64, 0x95, 0xED));
        colorList.add(new ColorName("Cornsilk", 0xFF, 0xF8, 0xDC));
        colorList.add(new ColorName("Crimson", 0xDC, 0x14, 0x3C));
        colorList.add(new ColorName("Cyan", 0x00, 0xFF, 0xFF));
        colorList.add(new ColorName("DarkBlue", 0x00, 0x00, 0x8B));
        colorList.add(new ColorName("DarkCyan", 0x00, 0x8B, 0x8B));
        colorList.add(new ColorName("DarkGoldenRod", 0xB8, 0x86, 0x0B));
        colorList.add(new ColorName("DarkGray", 0xA9, 0xA9, 0xA9));
        colorList.add(new ColorName("DarkGreen", 0x00, 0x64, 0x00));
        colorList.add(new ColorName("DarkKhaki", 0xBD, 0xB7, 0x6B));
        colorList.add(new ColorName("DarkMagenta", 0x8B, 0x00, 0x8B));
        colorList.add(new ColorName("DarkOliveGreen", 0x55, 0x6B, 0x2F));
        colorList.add(new ColorName("DarkOrange", 0xFF, 0x8C, 0x00));
        colorList.add(new ColorName("DarkOrchid", 0x99, 0x32, 0xCC));
        colorList.add(new ColorName("DarkRed", 0x8B, 0x00, 0x00));
        colorList.add(new ColorName("DarkSalmon", 0xE9, 0x96, 0x7A));
        colorList.add(new ColorName("DarkSeaGreen", 0x8F, 0xBC, 0x8F));
        colorList.add(new ColorName("DarkSlateBlue", 0x48, 0x3D, 0x8B));
        colorList.add(new ColorName("DarkSlateGray", 0x2F, 0x4F, 0x4F));
        colorList.add(new ColorName("DarkTurquoise", 0x00, 0xCE, 0xD1));
        colorList.add(new ColorName("DarkViolet", 0x94, 0x00, 0xD3));
        colorList.add(new ColorName("DeepPink", 0xFF, 0x14, 0x93));
        colorList.add(new ColorName("DeepSkyBlue", 0x00, 0xBF, 0xFF));
        colorList.add(new ColorName("DimGray", 0x69, 0x69, 0x69));
        colorList.add(new ColorName("DodgerBlue", 0x1E, 0x90, 0xFF));
        colorList.add(new ColorName("FireBrick", 0xB2, 0x22, 0x22));
        colorList.add(new ColorName("FloralWhite", 0xFF, 0xFA, 0xF0));
        colorList.add(new ColorName("ForestGreen", 0x22, 0x8B, 0x22));
        colorList.add(new ColorName("Fuchsia", 0xFF, 0x00, 0xFF));
        colorList.add(new ColorName("Gainsboro", 0xDC, 0xDC, 0xDC));
        colorList.add(new ColorName("GhostWhite", 0xF8, 0xF8, 0xFF));
        colorList.add(new ColorName("Gold", 0xFF, 0xD7, 0x00));
        colorList.add(new ColorName("GoldenRod", 0xDA, 0xA5, 0x20));
        colorList.add(new ColorName("Gray", 0x80, 0x80, 0x80));
        colorList.add(new ColorName("Green", 0x00, 0x80, 0x00));
        colorList.add(new ColorName("GreenYellow", 0xAD, 0xFF, 0x2F));
        colorList.add(new ColorName("HoneyDew", 0xF0, 0xFF, 0xF0));
        colorList.add(new ColorName("HotPink", 0xFF, 0x69, 0xB4));
        colorList.add(new ColorName("IndianRed", 0xCD, 0x5C, 0x5C));
        colorList.add(new ColorName("Indigo", 0x4B, 0x00, 0x82));
        colorList.add(new ColorName("Ivory", 0xFF, 0xFF, 0xF0));
        colorList.add(new ColorName("Khaki", 0xF0, 0xE6, 0x8C));
        colorList.add(new ColorName("Lavender", 0xE6, 0xE6, 0xFA));
        colorList.add(new ColorName("LavenderBlush", 0xFF, 0xF0, 0xF5));
        colorList.add(new ColorName("LawnGreen", 0x7C, 0xFC, 0x00));
        colorList.add(new ColorName("LemonChiffon", 0xFF, 0xFA, 0xCD));
        colorList.add(new ColorName("LightBlue", 0xAD, 0xD8, 0xE6));
        colorList.add(new ColorName("LightCoral", 0xF0, 0x80, 0x80));
```

```
        colorList.add(new ColorName("LightCyan", 0xE0, 0xFF, 0xFF));
        colorList.add(new ColorName("LightGoldenRodYellow", 0xFA, 0xFA, 0xD2));

colorList.add(new ColorName("LightGray", 0xD3, 0xD3, 0xD3));
        colorList.add(new ColorName("LightGreen", 0x90, 0xEE, 0x90));
        colorList.add(new ColorName("LightPink", 0xFF, 0xB6, 0xC1));
        colorList.add(new ColorName("LightSalmon", 0xFF, 0xA0, 0x7A));
        colorList.add(new ColorName("LightSeaGreen", 0x20, 0xB2, 0xAA));
        colorList.add(new ColorName("LightSkyBlue", 0x87, 0xCE, 0xFA));
        colorList.add(new ColorName("LightSlateGray", 0x77, 0x88, 0x99));
        colorList.add(new ColorName("LightSteelBlue", 0xB0, 0xC4, 0xDE));
        colorList.add(new ColorName("LightYellow", 0xFF, 0xFF, 0xE0));
        colorList.add(new ColorName("Lime", 0x00, 0xFF, 0x00));
        colorList.add(new ColorName("LimeGreen", 0x32, 0xCD, 0x32));
        colorList.add(new ColorName("Linen", 0xFA, 0xF0, 0xE6));
        colorList.add(new ColorName("Magenta", 0xFF, 0x00, 0xFF));
        colorList.add(new ColorName("Maroon", 0x80, 0x00, 0x00));
        colorList.add(new ColorName("MediumAquaMarine", 0x66, 0xCD, 0xAA));
        colorList.add(new ColorName("MediumBlue", 0x00, 0x00, 0xCD));
        colorList.add(new ColorName("MediumOrchid", 0xBA, 0x55, 0xD3));
        colorList.add(new ColorName("MediumPurple", 0x93, 0x70, 0xDB));
        colorList.add(new ColorName("MediumSeaGreen", 0x3C, 0xB3, 0x71));
        colorList.add(new ColorName("MediumSlateBlue", 0x7B, 0x68, 0xEE));
        colorList.add(new ColorName("MediumSpringGreen", 0x00, 0xFA, 0x9A));
        colorList.add(new ColorName("MediumTurquoise", 0x48, 0xD1, 0xCC));
        colorList.add(new ColorName("MediumVioletRed", 0xC7, 0x15, 0x85));
        colorList.add(new ColorName("MidnightBlue", 0x19, 0x19, 0x70));
        colorList.add(new ColorName("MintCream", 0xF5, 0xFF, 0xFA));
        colorList.add(new ColorName("MistyRose", 0xFF, 0xE4, 0xE1));
        colorList.add(new ColorName("Moccasin", 0xFF, 0xE4, 0xB5));
        colorList.add(new ColorName("NavajoWhite", 0xFF, 0xDE, 0xAD));
        colorList.add(new ColorName("Navy", 0x00, 0x00, 0x80));
        colorList.add(new ColorName("OldLace", 0xFD, 0xF5, 0xE6));
        colorList.add(new ColorName("Olive", 0x80, 0x80, 0x00));
        colorList.add(new ColorName("OliveDrab", 0x6B, 0x8E, 0x23));
        colorList.add(new ColorName("Orange", 0xFF, 0xA5, 0x00));
        colorList.add(new ColorName("OrangeRed", 0xFF, 0x45, 0x00));
        colorList.add(new ColorName("Orchid", 0xDA, 0x70, 0xD6));
        colorList.add(new ColorName("PaleGoldenRod", 0xEE, 0xE8, 0xAA));
        colorList.add(new ColorName("PaleGreen", 0x98, 0xFB, 0x98));
        colorList.add(new ColorName("PaleTurquoise", 0xAF, 0xEE, 0xEE));
        colorList.add(new ColorName("PaleVioletRed", 0xDB, 0x70, 0x93));
        colorList.add(new ColorName("PapayaWhip", 0xFF, 0xEF, 0xD5));
        colorList.add(new ColorName("PeachPuff", 0xFF, 0xDA, 0xB9));
        colorList.add(new ColorName("Peru", 0xCD, 0x85, 0x3F));
        colorList.add(new ColorName("Pink", 0xFF, 0xC0, 0xCB));
        colorList.add(new ColorName("Plum", 0xDD, 0xA0, 0xDD));
        colorList.add(new ColorName("PowderBlue", 0xB0, 0xE0, 0xE6));
        colorList.add(new ColorName("Purple", 0x80, 0x00, 0x80));
        colorList.add(new ColorName("Red", 0xFF, 0x00, 0x00));
        colorList.add(new ColorName("RosyBrown", 0xBC, 0x8F, 0x8F));
        colorList.add(new ColorName("RoyalBlue", 0x41, 0x69, 0xE1));
        colorList.add(new ColorName("SaddleBrown", 0x8B, 0x45, 0x13));
        colorList.add(new ColorName("Salmon", 0xFA, 0x80, 0x72));
        colorList.add(new ColorName("SandyBrown", 0xF4, 0xA4, 0x60));
        colorList.add(new ColorName("SeaGreen", 0x2E, 0x8B, 0x57));
        colorList.add(new ColorName("SeaShell", 0xFF, 0xF5, 0xEE));
        colorList.add(new ColorName("Sienna", 0xA0, 0x52, 0x2D));
        colorList.add(new ColorName("Silver", 0xC0, 0xC0, 0xC0));
        colorList.add(new ColorName("SkyBlue", 0x87, 0xCE, 0xEB));
        colorList.add(new ColorName("SlateBlue", 0x6A, 0x5A, 0xCD));
```

```
        colorList.add(new ColorName("SlateGray", 0x70, 0x80, 0x90));
        colorList.add(new ColorName("Snow", 0xFF, 0xFA, 0xFA));
        colorList.add(new ColorName("SpringGreen", 0x00, 0xFF, 0x7F));

colorList.add(new ColorName("SteelBlue", 0x46, 0x82, 0xB4));
        colorList.add(new ColorName("Tan", 0xD2, 0xB4, 0x8C));
        colorList.add(new ColorName("Teal", 0x00, 0x80, 0x80));
        colorList.add(new ColorName("Thistle", 0xD8, 0xBF, 0xD8));
        colorList.add(new ColorName("Tomato", 0xFF, 0x63, 0x47));
        colorList.add(new ColorName("Turquoise", 0x40, 0xE0, 0xD0));
        colorList.add(new ColorName("Violet", 0xEE, 0x82, 0xEE));
        colorList.add(new ColorName("Wheat", 0xF5, 0xDE, 0xB3));
        colorList.add(new ColorName("White", 0xFF, 0xFF, 0xFF));
        colorList.add(new ColorName("WhiteSmoke", 0xF5, 0xF5, 0xF5));
        colorList.add(new ColorName("Yellow", 0xFF, 0xFF, 0x00));
        colorList.add(new ColorName("YellowGreen", 0x9A, 0xCD, 0x32));
return colorList;
    }

/**
     * Get the closest color name from our list
     *
     * @param r
     * @param g
     * @param b
     * @return
     */
public String getColorNameFromRgb(int r, int g, int b) {
        ArrayList<ColorName> colorList = initColorList();
        ColorName closestMatch = null;
int minMSE = Integer.MAX_VALUE;
int mse;
for (ColorName c : colorList) {
            mse = c.computeMSE(r, g, b);
if (mse < minMSE) {
                minMSE = mse;
closestMatch = c;
            }
        }

if (closestMatch != null) {
return closestMatch.getName();
} else {
return"No matched color name.";
        }
    }

/**
     * Convert hexColor to rgb, then call getColorNameFromRgb(r, g, b)
     *
     * @param hexColor
     * @return
     */
public String getColorNameFromHex(int hexColor) {
int r = (hexColor & 0xFF0000) >> 16;
int g = (hexColor & 0xFF00) >> 8;
int b = (hexColor & 0xFF);
return getColorNameFromRgb(r, g, b);
    }

publicintcolorToHex(Color c) {
```

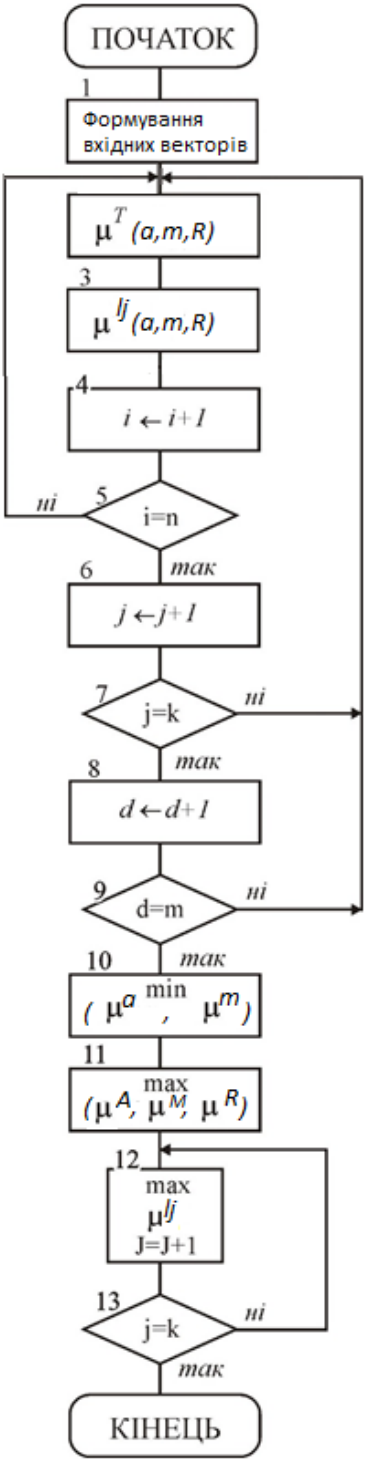
```
return Integer.decode("0x"
                + Integer.toHexString(c.getRGB()).substring(2));
}


public String getColorNameFromColor(Color color) {
return getColorNameFromRgb(color.getRed(), color.getGreen(),
color.getBlue());
    }

/**
     * SubClass of ColorUtils. In order to lookup color name
     *
     *
     *
     */
publicclassColorName {
publicint r, g, b;
public String name;
publicColorName(String name, int r, int g, int b) {
this.r = r;
this.g = g;
this.b = b;
this.name = name;
        }
publicintcomputeMSE(int pixR, int pixG, int pixB) {
return (int) (((pixR - r) * (pixR - r) + (pixG - g) * (pixG - g) + (pixB - b)
                    * (pixB - b)) / 3);
        }
publicintgetR() {
return r;
        }
publicintgetG() {
return g;
        }
publicintgetB() {
return b;
        }
public String getName() {
return name;
        }
    }
return (int);
}
```

Додаток В

Графічна частина



					08-22.МКР 010.19.000.ПЛ	
					Прогнозування трансформації автомобільних фарб	Літ. / Маса / Масштаб: 1:1
Змн.	Арк.	№ докум.	Підпис	Дата		Арк. 1 / Аркушів 6
Розроб.		Сілагін Є.О.			Алгоритм логічного виведення рішення	1КН-19м
Перевір.		Перевозніков С.І.				
Т. Контр.						
Реценз.		Романюк О.В.				
Н. Контр.		Іванчук Я.В.				
Затверд.		Яровий А.А.				

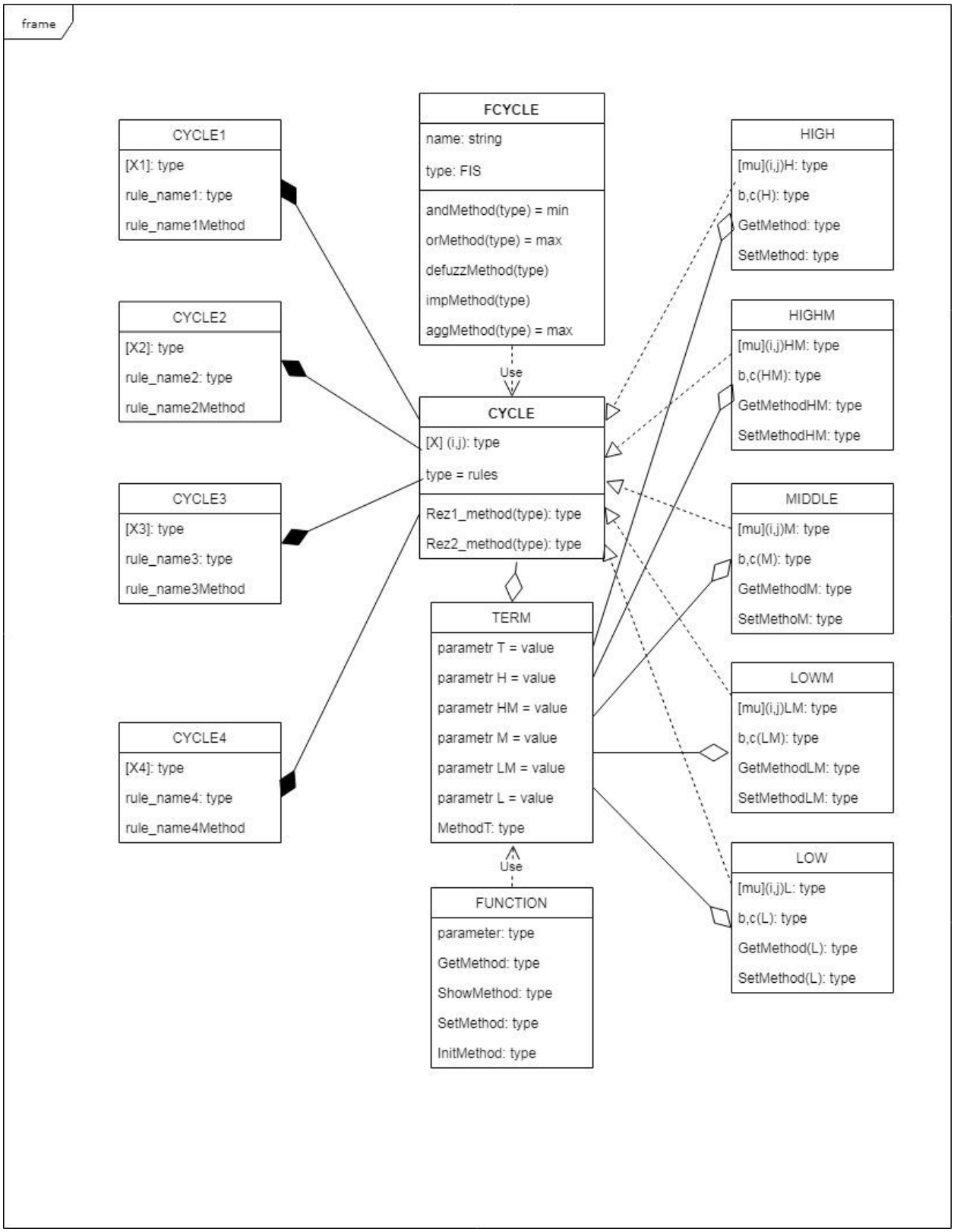


Рисунок 1 – Діаграма класів модуля виведення частичного показника

					08-22.МКР 010.19.000.ПЛ		
Змн.	Арк.	№ докум.	Підпис	Дата	Прогнозування трансформації автомобільних фарб	Літ.	Маса / Масштаб
Розроб.		Сілагін Є.О.					
Перевір.		Перевозніков С.І.				Арк. 2	Аркушів 6
Т. Контр.							
Реценз.		Романюк О.В.			Діаграма класів модуля виведення частичного показника	1КН-19м	
Н. Контр.		Іванчук Я.В.					
Затверд.		Яровий А.А.					

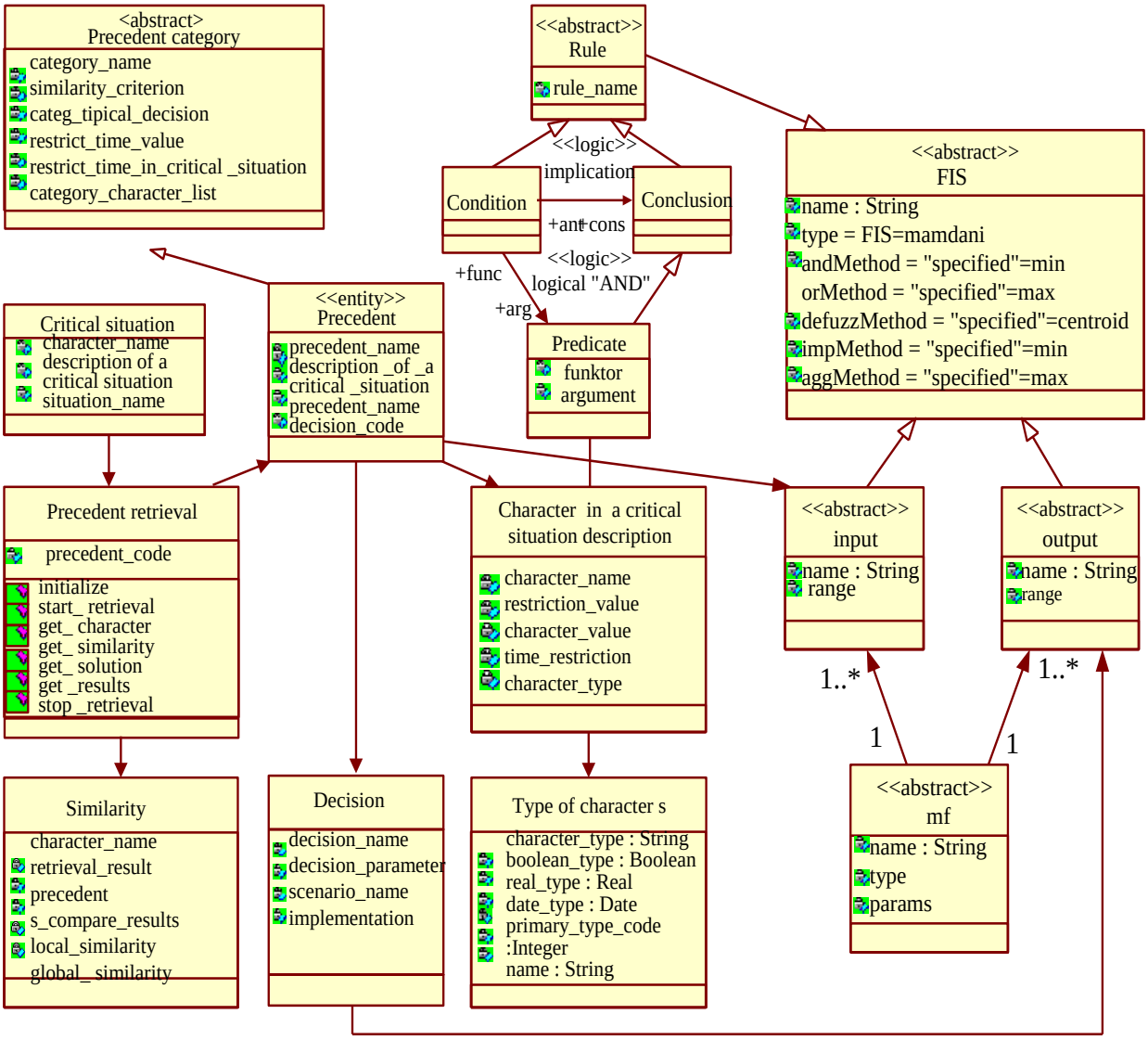


Рисунок 2 – шаблон “FUZZY CONCLUSION”

					08-22.МКР 010.19.000.ПЛ		
Змн.	Арк.	№ докум.	Підпис	Дата	Прогнозування трансформації автомобільних фарб	Літ.	Маса	Масштаб
Розроб.		Сілагін Є.О.						
Перевір.		Перевозніков С.І.				Арк. 3	Аркушів 6	
Т. Контр.								
Реценз.		Романюк О.В.			Шаблон “FUZZY CONCLUSION”	1КН-19м		
Н. Контр.		Іванчук Я.В.						
Затверд.		Яровий А.А.						

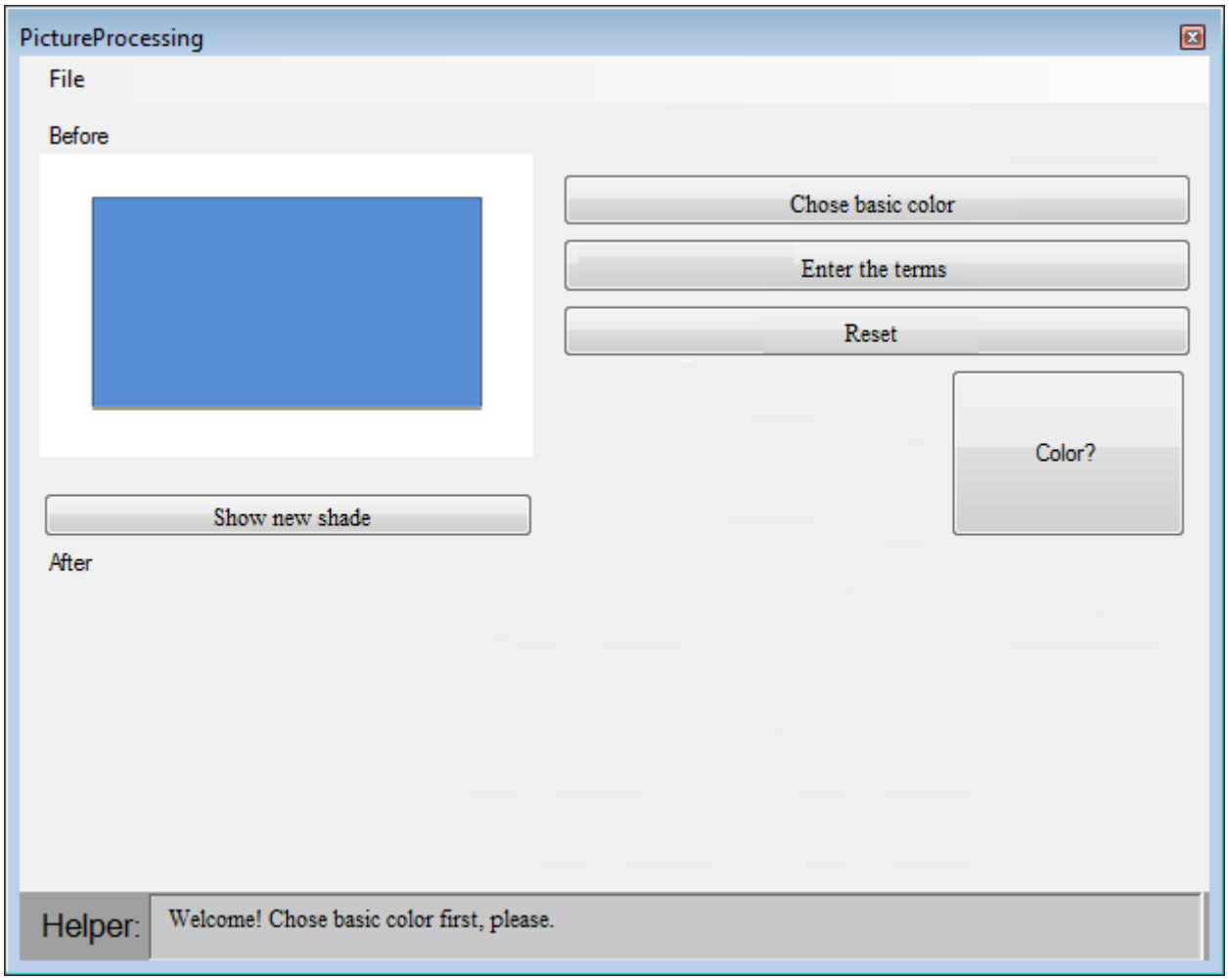


Рисунок 3 – загальний вигляд інтерфейсу користувача

					08-22.МКР 010.19.000.ПЛ	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Сілагін Є.О.			Прогнозування трансформації автомобільних фарб	Літ. / Маса / Масштаб
Перевір.		Перевозніков С.І.				Арк. 4 / Аркушів 6
Т. Контр.						
Реценз.		Романюк О.В.			Загальний вигляд інтерфейсу користувача	1КН-19м
Н. Контр.		Іванчук Я.В.				
Затверд.		Яровий А.А.				

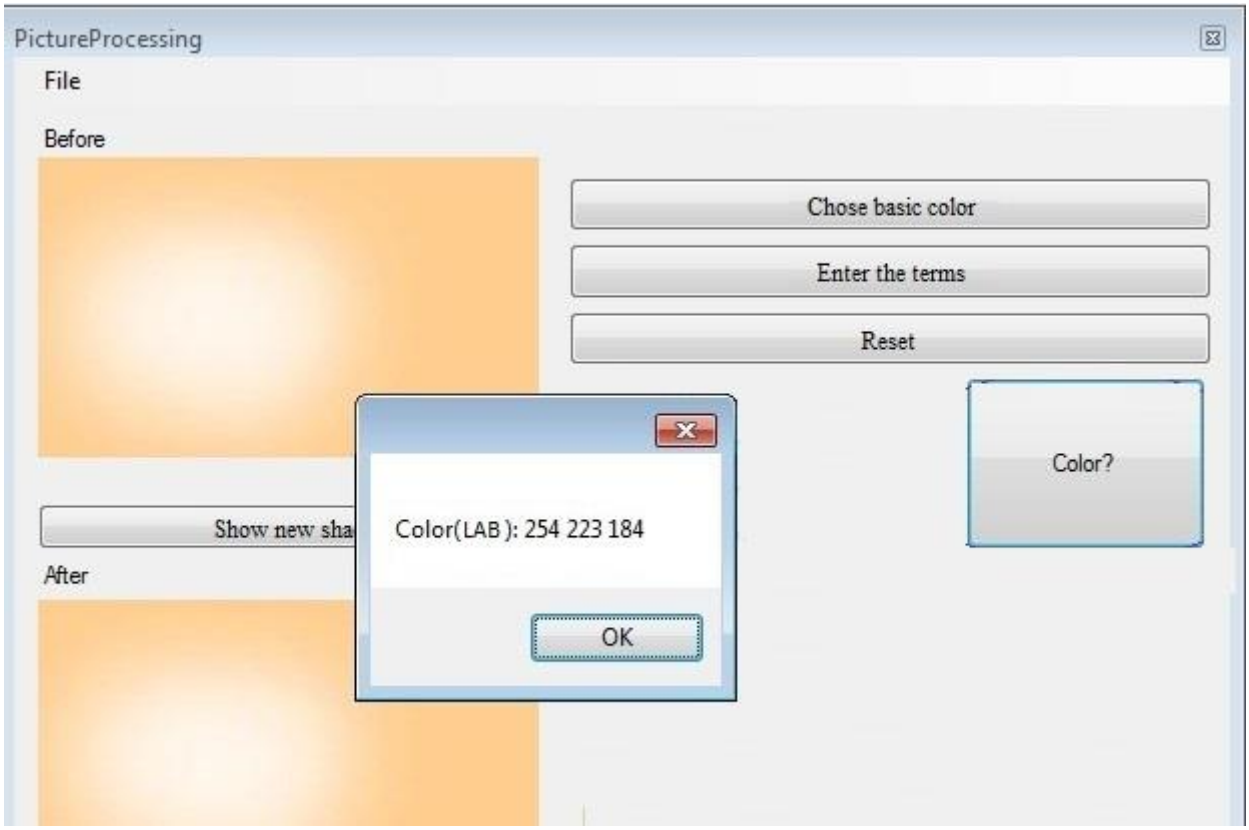


Рисунок 4 – приклад виконання програми

					08-22.МКР 010.19.000.ПЛ		
Змн.	Арк.	№ докум.	Підпис	Дата	Прогнозування трансформації автомобільних фарб	Літ.	Маса	Масштаб
Розроб.		Сілагін Є.О.						
Перевір.		Перевозніков С.І.				Арк. 5	Аркушів 6	
Т. Контр.								
Реценз.		Романюк О.В.			Приклад виконання програми	1КН-19м		
Н. Контр.		Іванчук Я.В.						
Затверд.		Яровий А.А.						

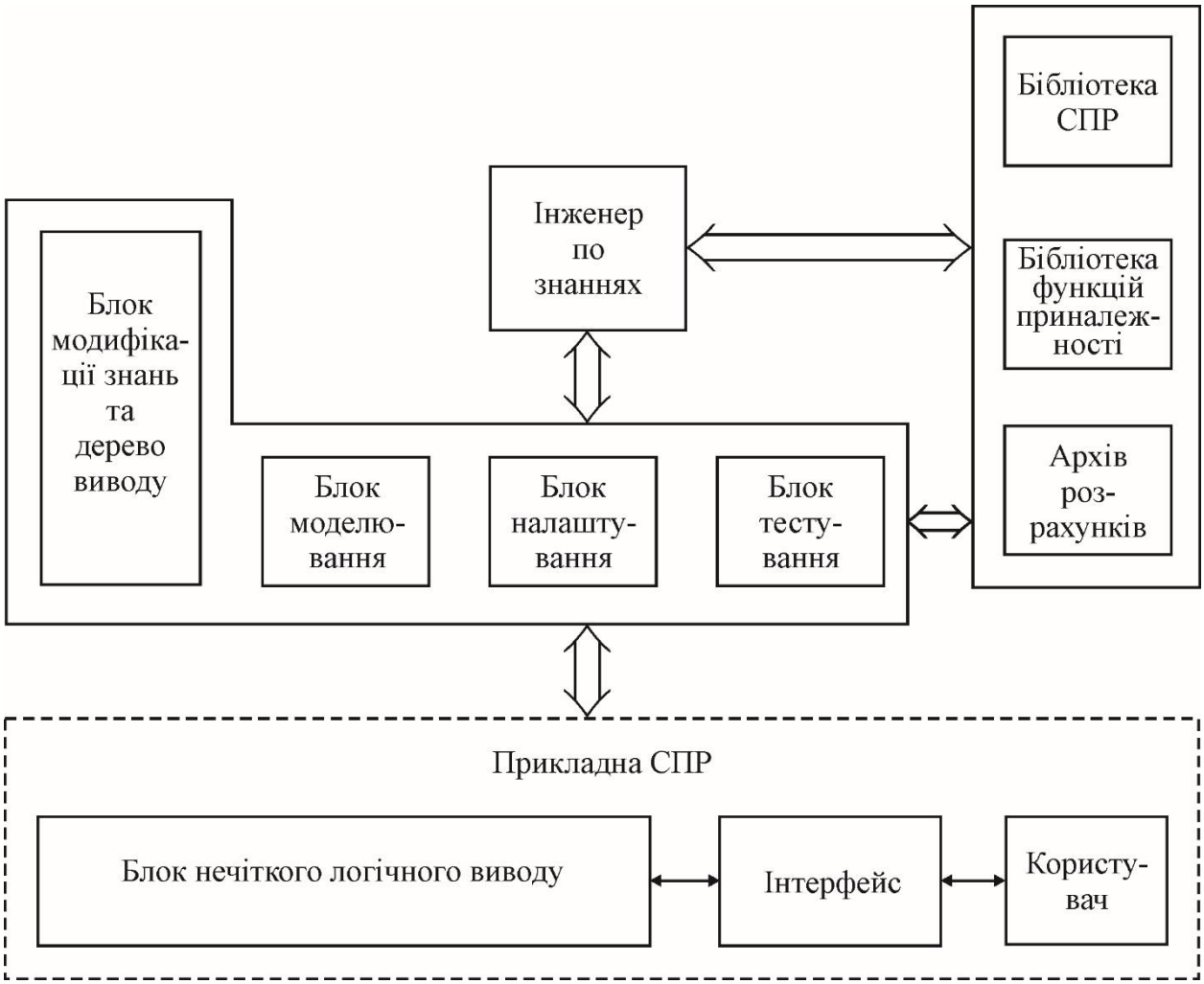


Рисунок 5 – Структурна схема середовища прийняття рішень

					08-22.МКР 010.19.000.ПЛ		
Змн.	Арк.	№ докум.	Підпис	Дата	Прогнозування трансформації автомобільних фарб	Літ.	Маса	Масштаб
Розроб.		Сілагін Є.О.						
Перевір.		Перевозніков С.І.						
Т. Контр.						Арк. 6	Аркушів 6	
Реценз.		Романюк О.В.			Структурна схема середовища прийняття рішень	1КН-19м		
Н. Контр.		Іванчук Я.В.						
Затверд.		Яровий А.А.						

Додаток Г

Д О В І Д К А

Дана Сілагіну Єгору Олексійовичу, магістранту ВНТУ (гр.1КН-19м) в тому, що результати, одержані ним в ході виконання магістерської кваліфікаційної роботи, а саме алгоритми та програмні засоби плануються до впровадження в розробки ТОВ «ІТІ».

Зам директора ТОВ «ІТІ» Бодяк В.М.