НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"_____"_____2016 р.

Дипломна робота

_____першого (бакалаврського) рівня вищої освіти

зі спеціальності _____7.05010102, 8.05010102 Інформаційні технології проектування
_____7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Виявлення та усунення технічних конфліктів в контекстно-залежних системах

Виконав: студент _____4_____ курсу, групи ДА-22 _____
(шифр групи)

_____Піпіч Артем Андрійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник _____доцент, к.т.н. Кисельова А.Г._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний професор, д.е.н. Семенченко Н.В _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____доцент, к.т.н. Хижняк Т.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст. викладач Бритов О.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Київ – 2016 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

Рівень вищої освіти Перший(Бакалаврський)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"_____"_____2016 р.

ЗАВДАННЯ
на дипломний проект студенту

Піпічу Артему Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту: Виявлення та усунення технічних конфліктів в контекстно-залежних системах

   керівник проекту Кисельова Анна Геннадіївна, к.т.н., доцент
   (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

   затверджені наказом по університету від «12» травня 2016 р. № 50

2. Строк подання студентом проекту 08.06.2016

3. Вихідні дані до проекту

   Вводяться користувачем:

   - Початкові параметри середовища;
   - Дані про сенсори середовища;
   - Дані про ресурси середовища;
   - Дані про обробники даних з сенсорів;

   Встановлені за замовчуванням:

   - Методи усунення конфліктів, що виникають між компонентами системи;

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

   1. Аналітичний огляд
   2. Постановка завдання
   3. Проектні рішення
   4. Експериментальні дослідження

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)
    1. UML-діаграма класів додатку — плакат;
    2. UML-діаграма прецедентів додатку — плакат;
    3. UML-діаграма послідовності додатку — плакат;
6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Завдання видав		Завдання прийняв	
		Підпис	Дата	Підпис	Дата
Економічна частина	Семенченко Н.В., професор				
Основна частина	Хижняк Т.А., доцент				

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
1	Отримання завдання	25.11.2015	
2	Збір інформації	20.02.2016	
3	Дослідження взаємодії компонентів в контекстно-залежних системах	20.03.2016	
4	Дослідження методів усунення конфліктів в технічно-залежних системах	04.05.2016	
5	Розробка програмного додатку	22.05.2016	
6	Тестування додатку	24.05.2016	
7	Розробка наборів вхідних даних, їх обробка	28.05.2016	
8	Аналіз отриманих результатів	31.05.2016	
9	Оформлення дипломної роботи	05.06.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2016	

Студент _____ А.А. Піпіч
(підпис) (ініціали, прізвище)

Керівник проекту _____ А.Г. Кисельова
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Тема роботи: виявлення та усунення технічних конфліктів в контекстно-залежних системах

Виконавець: Піпіч Артем Андрійович

Пояснювальна записка містить: 80 сторінок, 15 рисунків, 15 таблиць, 1 додаток на сторінці 71, 13 посилань.

Ключові слова: контекстно-залежна система, технічний конфлікт, методи усунення конфліктів, моделювання системи з конфліктами, дослідження продуктивності системи.

Мета роботи: дослідження впливу застосування методів усунення конфліктів на ефективність роботи контекстно-залежної системи.

Результати: програмний продукт, що моделює контекстно-залежну систему та застосовує в ній методи усунення конфліктів; дані про доцільність застосування конкретних методів усунення конфліктів в залежності від структури системи, в якій вони застосовуються.

АННОТАЦИЯ

Тема работы: обнаружение и устранение технических конфликтов в контекстно-зависимых системах

Исполнитель: Пипич Артем Андреевич

Пояснительная записка содержит: 80 страниц, 15 рисунков, 15 таблиц, 1 приложение на странице 71, 13 ссылок.

Ключевые слова: контекстно-зависимая система, технический конфликт, методы устранения конфликтов, моделирование системы с конфликтами, исследование производительности системы.

Цель работы: исследование влияния использования методов устранения конфликтов на эффективность работы контекстно-зависимой системы.

Результаты: программный продукт, моделирующий контекстно-зависимую систему и применяющий в ней методы устранения конфликтов; данные про целесообразность применения конкретных методов устранения конфликтов в зависимости от структуры системы, в которой они применяются.

ABSTRACT

R&D: Technical conflict detection and resolution in context-aware systems

Author: Artem Pipich

Work contains: 80 pages, 15 images, 15 tables, 1 attachment on page 71, 13 references.

Key words: Context-Aware System, Technical Conflit, Conflict Resolution Methods, Conflict System Design, System Efficiency Research.

The purpose: to research the impact of conflict resolution methods application on efficiency of context-aware system.

Results: software product, which simulates context-aware system and applies conflict resolution methods on it; data about expediency of applience of conflict resolution methods depending on the structure of system, which the applies on.

ЗМІСТ

ВСТУП

Мета, актуальність розробки об’єкту проектування

В даний час значна кількість галузей діяльності людини є в певному обсязі автоматизованими. Як у повсякденному житті, так і в складних процесах виробництва, присутні механізми, що регулюють певні параметри підконтрольних елементів. Разом з ними вони утворюють систему, що існує в даній галузі. Контекстом системи є та її частина, що належить безпосередньо до галузі та є керованою механізмом, а також середовище, що оточує механізм та несе певну інформацію про систему. Для більш ефективної роботи системи механізм керування нею може перебувати в різних станах в залежності від контексту. Такі системи є контекстно-залежними.

Для деяких систем може бути створена така конфігурація контексту, для якої аналіз різних його частин дає суперечливий результат, виражений в потребі встановлення в різні стани в одній ситуації. Таку ситуацію можна вважати технічним конфліктом в системі. Технічні конфлікти можуть призвести до зниження ефективності системи чи до припинення її функціонування, тому їхнє виявлення та вирішення є достатньо важливою частиною керування системою.

Мета даної роботи полягає в дослідженні конфліктів, що виникають в контекстно-залежних системах, а також методів їх розв’язання. Таке дослідження дозволить на основі класифікації конфліктів та класифікації методів вирішення конфліктів визначити ефективність застосування певних груп методів до певних груп конфліктів; охарактеризувати методи вирішення конфліктів і оцінити їх ефективність незалежно від типу та кількості конфліктів.

Обґрунтування основних проектних рішень або напрямків

Для проведення описаних досліджень було опрацьовано ряд джерел, що надають інформацію щодо класифікації конфліктів та методів їх розв’язання. На основі класифікації було описано особливості груп конфліктів та методів вирішення, що надало змогу сформулювати основні положення щодо ефективності застосування певних груп методів до певних груп конфліктів, а також ряд припущень з цього приводу.

Розробка моделі контекстно-залежної системи, в якій є можливим виникнення технічних конфліктів, а також застосування методів їх усунення, дозволяє перевірити сформульовані припущення, вивести та підтвердити нові. Відповідно до ефективності роботи системи в залежності від типу методів вирішення технічних конфліктів може бути визначена ефективність розглянутих методів, як безпосередня, так і порівняно з іншими методами при розв’язанні конфліктів певного типу.

З цієї причини розробка програмного засобу, що моделює поведінку контекстно-залежної системи з внутрішніми технічними конфліктами, значно покращує якість результатів, отриманих в роботі.

Застосування об’єктно-орієнтованого підходу при розробці програмного засобу дозволяє наочно демонструвати результати обраного методу усунення конфліктів.

Представлення отриманих на виході програмного продукту даних у графічному вигляді спрощує їх аналіз людиною і надає можливість надати зразки отриманої інформації в роботі в зручному для сприйняття вигляді.

З цих причин було прийняте проектне рішення щодо розробки відповідного програмного продукту на основі об’єктно-орієнтованого підходу; надання програмному продукту графічного інтерфейсу для подання отриманих результатів в вигляді, що добре піддається аналізу.

Можливі галузі застосування результатів роботи

Результати, отримані в програмі уточнюють та доповнюють результати, отримані при виконанні теоретичного дослідження методів усунення конфліктів. Висновки, сформульовані з них, дозволяють визначати найбільш ефективні методи вирішення конфліктів для певної конфігурації системи.

Такі дані можуть покращити як моделювання контекстно-залежної системи, так і функціонування реальної системи. Застосування відповідних методів усунення конфліктів може як збільшити швидкодію системи, так і зменшити об’єм ресурсів, які вона використовує. Обладнання системи модулем, що динамічно застосовує різні методи розв’язання конфліктів в залежності від поведінки контексту, дозволяє значно збільшити її гнучкість.

Так, описані в роботі підходи можуть бути застосовані в автоматизованих системах керування будинком для усунення конфліктів між його компонентами, такими як датчики, пристрої впливу на мікроклімат, побутові прилади, підключені до загальної інформаційної мережі приміщення.

Можливою галуззю застосування є і розробка ПО для пристроїв, що підтримують багатозадачність, в тому числі і пристроїв, інформація до яких може вводитися користувачами паралельно з кількох джерел, або кількома потоками, прикладом чого є сенсорний екран, одночасні дотики користувача до частин якого можуть спричинити конфлікти програмних компонентів під час роботи пристрою.

ОСНОВНА ЧАСТИНА

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Галузь застосування об’єкту проектування

З появою і поширенням мобільних пристроїв, розповсюджені (повсюдні) системи стають все більш популярними. Термін «розповсюджені», вперше введений Вайзером в 1991 р, описує безшовну інтеграцію пристроїв в повсякденне життя користувачів. Техніка розчиняється в ному, працюючи в фоновому режимі, розміщуючи фокус на користувачі і його задачі, а не обчислювальних пристроях і технічних питаннях. Одним з різновидів в широкому діапазоні розповсюджених систем є так звані контекстно-залежні (чутливі) системи.

Контекстно-залежні системи здатні адаптувати свої операції до поточного контексту без явного втручання користувача і, таким чином, спрямовані на підвищення зручності та ефективності, беручи контекст середовища до уваги. Зокрема, коли мова заходить про використання мобільних пристроїв, то бажано, щоб програми і послуги реагували саме на місцезнаходження, час та інші атрибути оточуючого середовища і адаптували свою поведінку відповідно до обставин. Необхідна інформація про контекст може бути отримана в різних формах, таких як покази датчиків, мережева інформація, стан пристрою, перегляд профілів користувачів.

1.1.1 Принципи проектування контекстно-залежних систем

Надалі описано основні принципи проектування та впровадження концептуально розшарованої структури, з метою зв'язати функціональні

можливості реалізовані в існуючих рамках різних прошарків та зображено різні моделі контексту, використані для представлення, зберігання та обміну контекстною інформацією.

1.1.1.1 Архітектура

Контекстно-залежні системи можуть бути реалізовані у багатьох відношеннях.

Підхід залежить від особливих вимог і умов, таких як розташування датчиків (локального або віддаленого), кількість можливих користувачів (один користувач або багато), наявні ресурси використовуваних пристроїв (потужних комп'ютерів або невеликих мобільних пристроїв), засоби подальшого розширення системи тощо. Крім того, спосіб отримання контекстних-даних є дуже важливим при розробці контекстно-залежних систем, так як він визначає архітектуру системи, принаймні до певної міри. Чен 2004 р. представив три різних підходи до отримання контекстної інформації:

- Direct sensor access;
- Middleware infrastructure;
- Context server.

Підхід Direct sensor access часто використовується в пристроях з локально вбудованими датчиками. Програмне забезпечення клієнта збирає потрібну інформацію безпосередньо з цих датчиків, тобто немає додаткового шару для отримання і обробки даних датчиків. Драйвери для датчиків вшито в додаток, так що цей безпосередній метод можна застосовувати тільки в окремих випадках.

Саме тому, він не підходить для розподілених систем — через прямий характер доступу, в якому відсутній компонент, здатний до керування доступом до кількох паралельних датчиків.

Сучасна розробка програмного забезпечення використовує методи інкапсуляції для поділу, наприклад, бізнес-логіки та графічних користувацьких інтерфейсів. Підхід Middleware infrastructure являє собою багаторівневу архітектуру для контекстно-залежних систем з метою приховування деталей сприйняття низького рівня. У порівнянні з прямим виходом датчика цей метод полегшує розширюваність, так як клієнтський код більше не змінюється, і це спрощує повторне використання апаратно-залежного коду шляхом суворої інкапсуляції.

Context server - наступний логічний крок, що надає кільком клієнтам доступ до віддалених джерел даних. Цей розподілений підхід розширює Middleware infrastructure шляхом введення доступу, що керує віддаленим компонентом. Збір даних датчика переміщається до цього так званого контекстного сервера для полегшення паралельного доступу. Крім повторного використання датчиків, використання контекстного сервера позбавляє клієнтів ресурсномістких операцій. Цілком ймовірно, що більшість користувацьких пристроїв контекстно-залежних систем - мобільні пристрої з обмеженими потужностями обчислень, дискового простору тощо. Проте постає питання прийнятних протоколів, продуктивності мережі, якості параметрів сервісу тощо, у випадку, коли проектування контекстно-залежні системи спирається на клієнт-серверну архітектуру.

1.1.2 Конфлікти в контекстно-залежних системах

Контекстно-залежні системи в повсюдних середовищах стикаються з конфліктними ситуаціями під час збору даних датчиків, їх обробки для вилучення послідовної та узгодженої інформації про контекст високого рівня, і поширення цієї інформації контексту для надання допомоги в прийнятті рішень з адаптації до постійно мінливих ситуацій, не

відволікаючи людську увагу. Ці конфліктні ситуації створюють серйозні проблеми для проектування і розробки контекстно-залежних систем. Для їх усунення може бути застосовано підхід Quality of Context. Надалі розглянуто конфліктні ситуації, з якими контекстно-залежні системи можуть зіткнутися на різних рівнях концептуального проектування і політики розв'язання конфліктів, що базуються на підході Quality of Context, а також показано, як ці політики можуть бути використані в різних конфліктних ситуаціях для підвищення продуктивності і ефективності контекстно-залежних систем.

Повсюдні середовища характеризується великою кількістю обчислювально-комунікаційних сенсорних пристроїв, вбудованих в об'єкти повсякденного користування. Основна мета цих пристроїв полягає в наданні допомоги користувачеві.

В результаті численних досліджень, такі задачі як:

- збір необроблених даних з датчиків;
- вилучення з даних інформації про контекст високого рівня;
- агрегація і зберігання інформації після усунення надлишкового та суперечливого контексту;
- надання інформації додаткам і користувачам;
- пристосування додатків та користувачів до отриманої інформації

були розподілені по різним концептуальним рівням [1].

Різні конфліктні ситуації можуть виникнути під час виконання вищезазначених завдань. Ці конфліктні ситуації сильно впливають на здатність контекстно-залежних систем адаптуватися до ситуації в повсюдних середовищах. Деякі системи використовують ряд простих стратегій, таких як скасування всіх конфліктних процесів (drop all), скасування останнього з них (drop last), скасування першого (drop first)

[2], залучення користувача для вирішення конфліктів [3], або ж посередництво на основі деяких зумовлених статичних політик [4]. Такі стратегії можуть вповільнити процес прийняття рішень, відволікти користувача, вилучити важливі об'єкти контексту тощо. Інші не можуть бути застосовані, так як контекстні конфлікти не можуть бути вирішені під час проектування[5] і необхідна стратегія, яка може динамічно обробляти їх під час роботи системи, не відволікаючи користувача.

Таблиця 1.1 - Концептуальні межі прошарків

Концептуальні межі прошарків	Конфлікт	Приклад
Здобуття контексту	Конфлікти в виборі між датчиками, що використовують для здобуття контексту різні підходи [5, 6]	GPS і GSM визначають інформацію про місцезнаходження користувача мобільного пристрою з різним ступенем точності
Обробка	Конфлікти в добуванні інформації контексту високого рівня [8, 21, 22]	Дані датчика показують, що хтось одночасно присутній в двох різних місцях
Розподіл контексту	Конфлікти в агрегації інформації контексту [6]	Надлишкові і суперечливі дані, що приходять в вузол з різних маршрутів
Додаток	Суперечливі інтереси додатків [4]	Різні налаштування, встановлені користувачами, присутніми в кімнаті

Quality of context (QoC) визначається як "будь-яка інформація, що описує якість інформації, яка використовується в як контекстна" [7]; може використовуватися для розробки політик вирішення конфліктів на різних рівнях концептуальної основи контекстно-залежних систем (таблиця 1). QoC також визначається як "невід'ємна інформації, яка описує інформацію контексту, і може бути використана для визначення цінності інформації для конкретного застосування" [8]. QoC може бути

класифікована за параметрами і джерелами. Параметри QoC, такі як своєчасність, надійність, повнота і значущість, використовуються для позначення якості контекстної інформації. Джерела QoC, такі як початкове розташування, час заміру, початковий стан, і початкова категорія використовуються для оцінки цих параметрів QoC.

В результаті аналізу загальних конфліктних ситуацій, які можуть виникнути на різних прошарках контекстно-залежні систем можуть бути отримані політики розв'язання конфліктів, засновані на якості параметрів контексту. Також такий аналіз надає інформацію про те, яким чином ці політики можуть бути використані; опис прототипу реалізації системи, до якої застосовано політику. Отримані політики можуть бути досліджені екпериментально. Зазвичай, ті політики, які використовують комбінацію різних параметрів QoC, є більш ефективними відповідно до перспективи використання контекстної інформації.

Розглянуті нижче конфлікти можуть мати місце на різних прошарках CMS, і в залежності від цього по різному впливати на продуктивність контекстно-залежних додатків. Ці ж конфлікти надані в таблиці 2.1.

1.1.2.1 Здобуття контексту

У повсюдних середовищах обсяг даних, отриманих за допомогою датчиків робить аналіз контексту неможливим для людини [9]. Дані датчиків також можуть відрізнятися один від одного, з огляду на частоту оновлення контексту, здатність датчика збирати контекст певної сутності, точність методу, що використовується датчиками, формату представлення, а також цінність контекстної інформації [5, 6]. Наприклад, інформація про місцезнаходження користувача мобільного пристрою може бути зібрана за допомогою GPS і GSM методів. Постійна оцінка датчика для збору контексту певної сутності не є можливою. Таким чином, існує

необхідність в стратегії, яка може динамічно вирішити, який датчик є більш надійним, щоб зібрати контекст певної сутності в певний момент часу. Параметри QoC, які були динамічно оцінені з інформації про джерело контексту можуть бути використані для вирішення конфліктів в такій ситуації.

1.1.2.2 Обробка

На стадії обробки, контекст високого рівня витягується з даних датчика низького рівня. Дані датчиків не можуть бути представлені безпосередньо в додатку. Вони повинні бути відфільтровані, зібрані в цілісну структуру, корельовані, перекладені для вилучення даних контексту вищого рівня і виявлення зареєстрованих подій [10].

Ці показники використовуються, щоб прийняти рішення щодо добування інформації про контекст високого рівня. Так, заявлена достовірність джерела контексту може бути використана для прийняття рішення про витягання певної блоку інформації шляхом комбінування даних з різних джерел.

QoC параметри, які надають інформацію про своєчасність, надійність, повноту і значущість, можуть замінити ці метрики і зробити прийняття рішень з обробки даних більш змістовними та реалістичними для усунення конфліктів.

1.1.2.3 Розподіл контексту

Рухомість датчиків, ненадійні бездротові з'єднання, а також характер завдань в повсюдних середовищах призводить до накопичення великих обсягів надлишкового і суперечливого контексту, що не тільки призводить до витрат обмежених ресурсів, а й може призвести до небажаного поводження контекстно-залежних додатків. Прості політики усунення конфліктів (drop first, drop all) можуть привести до втрати цінної

інформації. У критичній ситуації, наприклад, функціонування контекстно-залежного додатку медицини на дому[11] і телемедицини [12], втрата інформації може призвести до серйозних наслідків. Рішення про відміну чи збереження сутності контексту доцільніше приймати відповідно до політик, визначених QoC параметрами.

1.1.2.4 Додатки

Контекстно-залежні додатки використовують контекстну інформацію, щоб адаптувати свою поведінку до потреб користувачів і змін в середовищі. Якщо конфлікти не будуть вирішені в контекстній інформації на більш ранніх стадіях, то вони виникнуть і в додатках, які приймають рішення відповідно до контекстної інформації. Контекстно-залежні додатки можуть також бути втягнуті в конфлікт відповідно до різних пріоритетів, встановлених користувачами. Для врегулювання роботи таких додатків використовуються різні стратегії [4]. Інформація про своєчасність, надійність, повноту і значущість контекстної інформації дозволяє легко вирішувати конфлікти і приймати рішення на її основі.

1.2 Існуючі методи усунення конфліктів

1.2.1 QoC-орієнтовані політики усунення конфліктів

Основною ідеєю цієї групи політик є вирішення конфліктів таким чином, що рішення має бути прийняте на користь об'єкт контексту, який містить контекстну інформацію найвищої якості. Ця якість контекстної інформації характеризується параметрами QoC. У таблиці 2 наведені вирази, які використовуються для їх оцінки.

Параметри QoC в діапазоні [0..1][13]. Користувач політики має можливість обрати кілька об’єктів контексту, встановити порогову якість, яку б задовільняла якість обраних об’єктів. Існує кілька основних

політик, заснованих на різних параметрах QoC.

Таблиця 1.2 - Оцінка QoC параметрів

QoC	Вираз для оцінки QoC параметрів об’єкта контексту O
Up-to-datedness	$\begin{cases} 1 - \frac{Age(O)}{Lifetime(O)}: & \text{if } Age(O) < Lifetime(O) \\ 0: & otherwise \end{cases}$
Trustworthiness	$\begin{cases} 1 - \frac{d(S,)}{d_{max}} * \delta \ : & if \ d(S,) < d_{max} \\ undefined \ : & otherwise \end{cases}$
Completeness	$\begin{cases} (\frac{\Sigma_{j=0}^{m} w_j(O)}{\Sigma_{i=0}^{n} w_i(O)} \ : & if \ m,n < \infty \\ 0 \ : & otherwise \end{cases}$
Significance	$\frac{CV(O)}{CV_{max}(O)}$

1.2.1.1 Up-to-datedness Based Policy

Своєчасність (up-to-datedness) показує, наскільки раціонально використовувати об'єкт контексту в певний момент часу. Своєчасність об'єкта контексту може бути розрахована як відношення між віком цього контекстного об'єкта і періоду життєвого циклу інформації того типу, що міститься в даному об'єкті контексту. Цей показник може бути корисним у вирішенні конфлікту в об'єкті, який змінюється дуже швидко, наприклад, місце розташування рухомого транспортного засобу. У цьому випадку доцільнішим буде використання об'єкта контексту з найвищим значенням своєчасності. Однак, своєчасність не матиме вагомого значення в разі конфлікту в статичній інформації, що містилась в системі, наприклад, в інформації про структуру розумного будинку.

1.2.1.2 Trustworthiness Based Policy

Достовірність (trustworthiness) це ступінь придатності датчика до збору контексту певного типу. Достовірність контекстного об'єкта може бути розрахована на основі концепції просторової роздільної здатності та

точності датчика в процесі вимірювання такого роду інформації. Ця концепція особливо корисна при вирішенні конфлікту, коли ми маємо більше ніж один датчик для збору контексту тієї ж сутності або події. Наприклад, є датчики температури в різних місцях в вітальні розумного будинку, який побудований, щоб забезпечити комфортне життя літнім людям. Датчики, встановлені поруч з електричним нагрівачем будуть посилати більш високе значення температури в порівнянні з датчиками в інших місцях. Для того, щоб забезпечити комфортну температуру в приміщенні, ми будемо більше покладатися на показники датчиків, які знаходяться ближче до області сидіння, ніж датчиків в кутах вітальні і датчиків біля нагрівача.

1.2.1.3 Completeness Based Policy

Повнота (Completeness) інформації про контекст вказує на те, що всі аспекти контекстної інформації були представлені контекстним об'єктом. Повноту об'єкта може бути оцінена як відношення суми ваг доступних атрибутів об'єкта до суми ваг загальної кількості атрибутів об'єкта. Повнота об'єкта є особливо важливим параметром для того, щоб отримати повне уявлення про поточну ситуацію в реальному світі. Відповідно до даної політики доцільніше приймати рішення, орієнтовно до об’єкта з більш повною інформацією про поточну ситуацію.

1.2.1.4 Significance Based Policy

Значущість (Significance) вимірює цінність або вартість контекстного об'єкт. Особливо важливо враховувати дану метрику, коли наявний критично важливий контекстний об'єкт. Наприклад, якщо датчики диму виявити густий дим в спальні. Цей показник можна використовувати для генерації подій, яким необхідна негайна обробка додатком. Додатки можуть вказувати, що про контекстні об'єкти з високим

рівнем значущості слід повідомляти в пріоритетному порядку.

Крім зазначених вище основних стратегій, політики також можуть бути визначені на основі двох або більше параметрів QoC, в залежності від вимог конкретного додатку. Наприклад, політика може бути також визначена шляхом об'єднання параметрів QoC, таких як своєчасність і достовірністю. У такій політиці середнє (або більше/менше — політика оптимізму/песимізму) значення зазначених параметрів QoC використовується для прийняття рішень.

Користувач політик усунення конфлікту встановлює порогове значення відповідно до власних вимог з урахуванням перспективи використання контекстної інформації.

1.3 Висновки

Для покращення роботи існуючих методів усунення конфліктів є доцільним дослідження їх ефективності в системах певної структури та складу, інтенсивності взаємодії її компонентів та періоду їх життєвого циклу.

Таке дослідження може бути проведене шляхом моделювання контекстно-залежної системи з варійованими параметрами.

Модель має надавати можливість застосувати певний підхід до усунення конфліктів при попередньо встановленій конфігурації компонетів, що почали взаємодію, і, можливо, утворили ряд конфліктів.

Така модель може бути реалізована засобами мови програмування Java, з використанням парадигм об’єктно-орієнтованого програмування.

Подальший аналіз результатів роботи програми надасть детальну інформацію про доцільність застосування відповідних методів в відповідних типах систем.

2 ПОСТАНОВКА ЗАВДАННЯ

2.1 Задача дипломного проектування

2.1.1 Тема

Виявлення та вирішення технічних конфліктів в контекстно-залежних системах

2.1.2 Мета

Дослідження впливу застосування методів усунення конфліктів на ефективність роботи контекстно-залежної системи.

2.1.3 Методи рішення

- Моделювання складної системи з варійованими параметрами;
- Test-driven development;
- Статистичний аналіз отриманих результатів;

2.2 Вхідна інформація

Набір даних по компоненти контекстно-залежної системи: сенсори, ресурси, обробники; методи усунення конфліктів, що застосовують в системі.

2.3 Вихідна інформація

Ефективність роботи системи при заданій конфігурації: кількість енергії, що споживає система; ступінь відповідності параметрів системи параметрам, встановленим користувачем.

2.4 Засоби, використані в роботі

- Пакети програм:
  - Libre Office 5, ProjectLibre, IntelliJ IDEA 2016.1.2, JVM: OpenJDK;
- Типові проектні рішення:
  - Застосування шаблонів проектування (MVC, Command, Abstract Factory);
  - Застосування елементів scrum-методології (спрінти);
- Операційна ситема:
  - Linux Mint 17.3 "Rosa" - Cinnamon (64-bit);

2.5 Етапи проектування

1. Розглянути методи усунення конфліктів в контекстно-залежних системах;
2. Розробити UML-діаграму класів додатку для дослідження розглянутих методів;
3. Розробити Java-додаток, що дозволяє дослідити розглянуті методи на моделі контекстно-залежної системи;
4. Протестувати розроблений додаток;
5. Провести дослідження розглянутих методів за допомогою додатку;
6. Зробити висновки щодо застосування різних методів усунення конфліктів;

2.6 Висновки

Сформульовано мету роботи, окреслено її основні етапи, визначено засоби її виконання. Визначено основні характеристики додатку, який має бути розроблено в ході виконання роботи.

3 ПРОЕКТНІ РІШЕННЯ

3.1 Функціонально-вартісний аналіз програмного продукту

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для перевірки усунення конфліктів у контекстно-залежних системах. Програмний продукт був розроблений за допомогою мови програмування Java.

Програмний продукт є крос-платформенним але рекомендується для використання на персональних комп'ютерах під управлінням операційної системи Linux.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.
- для кожної функції визначаються повні річні витрати й кількість робочих часів.
- для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

3.1.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи для дослідження засобів усунення технічних конфліктів в контекстно-залежних системах. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для дослідження засобів усунення технічних конфліктів в контекстно-залежних системах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на персональних комп'ютерах із стандартним набором компонент;
- забезпечувати високу швидкість обробки даних у реальному часі;
- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

3.1.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який будує набір моделей прогнозу та перевіряє їх точність на певному наборі даних. Виходячи з конкретної мети, можна виділити основні функції ПП:

- F_2 – вибір IDE;
- F_1 – вибір мови програмування;
- F_3 – вибір бібліотеки графічного інтерфейсу користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

- Функція F_1:
  a) Qt Creator;
  b) Intellij IDEA/CLion;
- Функція F_2:
  a) мова програмування C++;
  b) мова програмування Java;
- Функція F_3:
  a) Qt libraries;
  b) JavaFX

3.1.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 3.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 3.1).

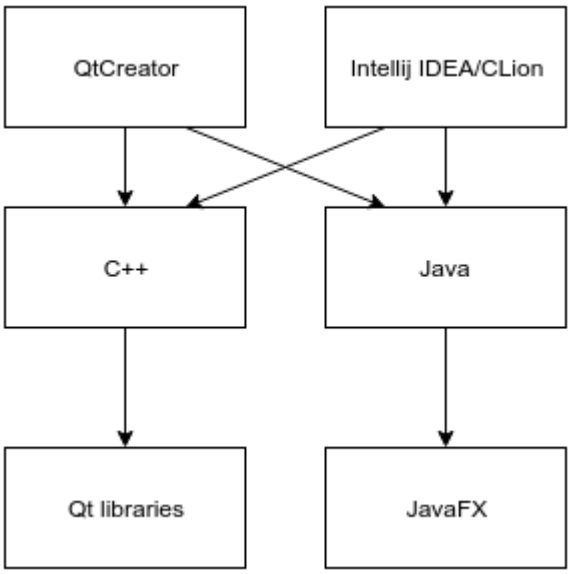

Рисунок 3.1 - Морфологічна карта

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 3.1 - Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	a	Детальна довідка	Менш потужні інструменти рефакторингу
	b	Потужні інструменти рефакторингу	Помилки при підключенні ряду розширень
F2	a	Економне використання пам’яті	Необхідність керування пам’яттю власноруч
	b	Автоматичне керування пам’яттю	Ресурсоємність JVM
F3	a	Краща підтримка	Відсутність гнучкості
	b	Гнучкість	Мінливість з версіями, відсутність однозначності в підтримці

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

- Функція F1:

  Обидва варіанти можуть бути розглянуті як ті, що відповідають поставленій задачі.
- Функція F2:
  - Варіант a) може бути використано для a) F1, але разом з b) F1 стає несумісним з a) F3 і b) F3
  - Варіант b) може бути використаний для b) F1, але виникають складнощі з встановленням розширень при використанні a) F1
- Функція F3:
- Варіант a) F3 сумісний лише з a) F1
- Варіант b) сумісний лише з b) F2

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1a – F2a – F3a
2. F1b – F2b – F3b

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

3.1.2 Обґрунтування системи параметрів ПП

3.1.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм пам’яті для збереження даних;
- X3 – час обробки даних;
- X4 – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

3.1.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 3.2.

Таблиця 3.2 - Основні параметри ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	2000	11000	19000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	800	420	60
Потенційний об’єм програмного коду	X4	кількість рядків коду	2000	1500	1000

За даними таблиці 3.2 будуються графічні характеристики параметрів – рис. 3.2 – рис. 3.5.

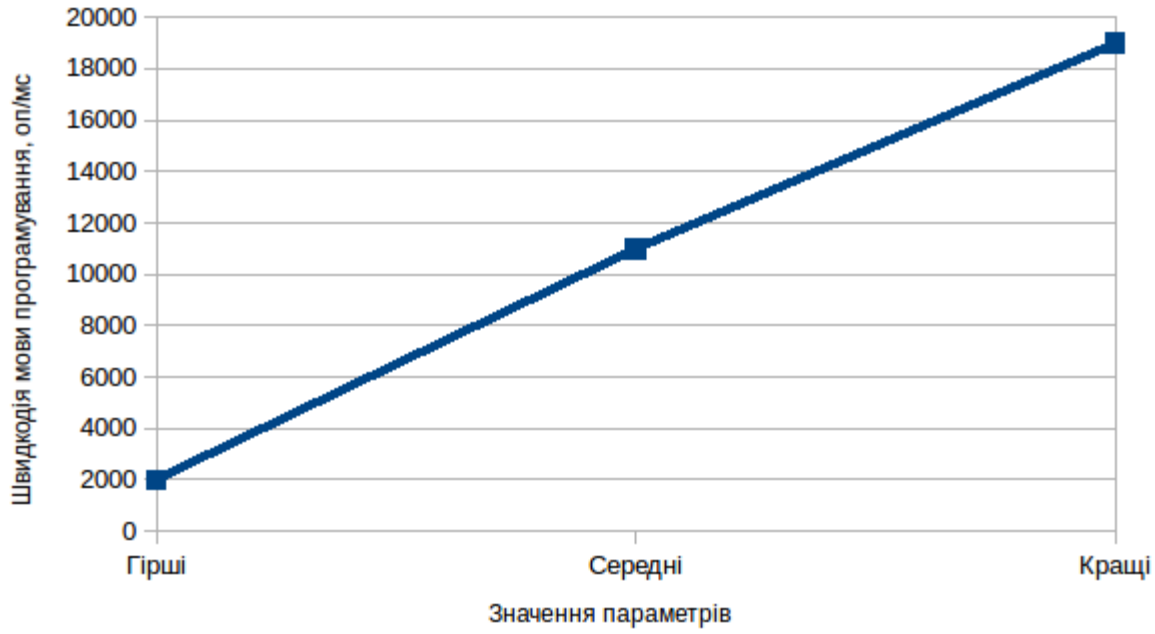

Рисунок 3.2 - Графік швидкодії мови програмування

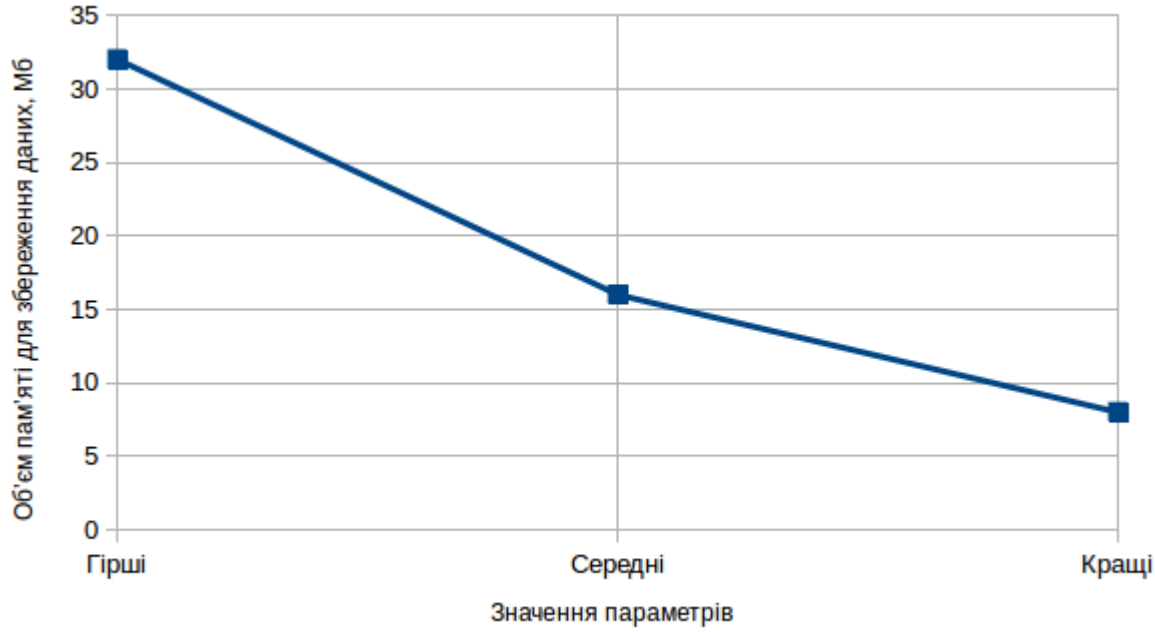

Рисунок 3.3 - Графік об’єму пам’яті для збереження даних

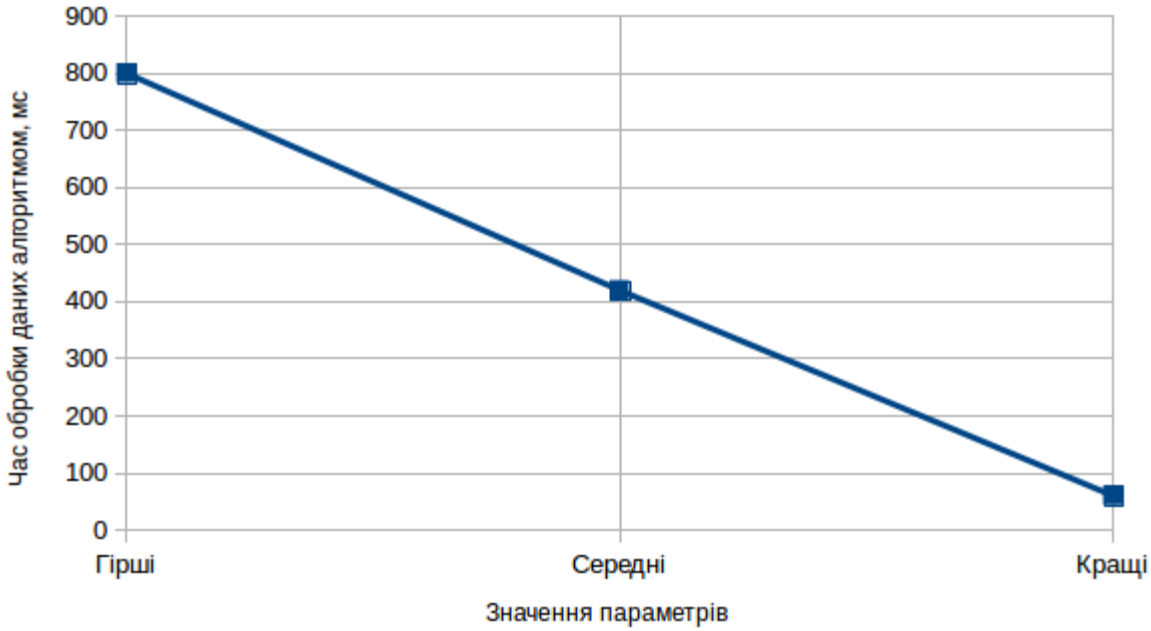

Рисунок 3.4 - Графік часу обробки даних алгоритмом

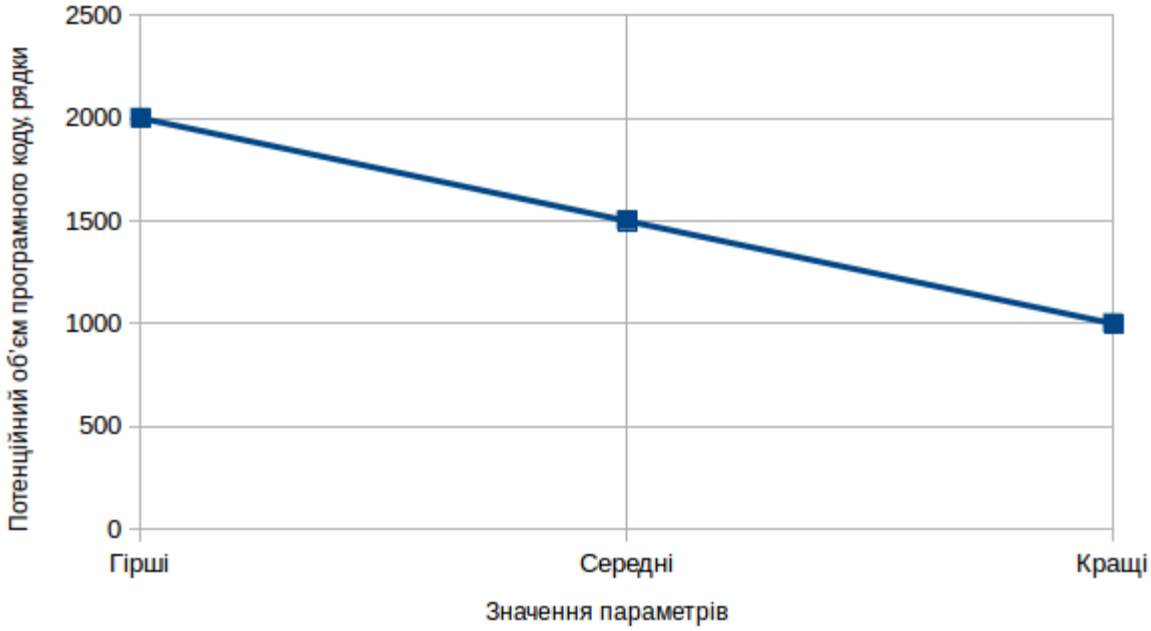

Рисунок 3.5 - Графік потенційного об’єму програмного коду

3.1.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 3.3.

Таблиця 3.3 - Результати ранжування параметрів

Позначення параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
		1	2	3	4	5	6	7			
X1	Оп/мс	4	3	4	4	4	4	4	27	0.75	0.56
X2	Мб	4	4	4	3	4	3	3	25	-1.25	1.56
X3	Мс	2	2	1	2	1	2	2	12	-14.25	203.06
X4	Рядки коду	5	6	6	6	6	6	6	41	14.75	217.56
$\sum X_i$	-	15	15	15	15	15	15	15	105	0	420.75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105 \quad ,$$

де N – число експертів, n – кількість параметрів;

b) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26.25$$

c) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

d) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420.75}{7^2(5^3 - 5)} = 1,03 > W_k = 0.67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 3.4.

Таблиця 3.4 - Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	>	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 & : X_i > X_j \\ 1.0 & : X_i = X_j \\ 0.5 & : X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю

$$A = ||a_{ij}||$$

кожного параметра зробимо розрахунок вагомості K_{wi} за наступними формулами:

$$K_{wi} = \frac{b_i}{\Sigma_{i=1}^{n} b_i}, \tag{3.1}$$

де

$$b_i = \Sigma_{i=1}^{N} a_{ij}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулою 3.1, де

$$b_i = \Sigma_{i=1}^{N} b_j$$

Як видно з таблиці 3.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 3.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	K_{wi}	b_i^1	K_{wi}^1	b_i^2	K_{wi}^2
X1	1.0	0.5	0.5	1.5	3.5	0.219	22.25	0.216	100	0.215
X2	1.5	1.0	0.5	1.5	4.5	0.281	27.25	0.282	124.25	0.283
X3	1.5	1.5	1.0	1.5	5.5	0.344	34.25	0.347	156	0.348
X4	0.5	0.5	0.5	1.0	2.5	0.156	14.25	0.155	64.75	0.154
Всього:					16	1	98	1	445	1

3.1.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2 (об’єм пам’яті для збереження даних) X1 (швидкодія мови програмування) та X3 відповідають технічним вимогам умов функціонування даного ПП.1

Абсолютне значення параметра X4 (потенційний об’єм програмного коду) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту a) 1800 або варіанту b) 1200.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 3.6):

$$K_K(j) = \Sigma_{i=1}^{n} K_{wi,j} B_{i,j}$$

де n – кількість параметрів; K_{wi} – коефіцієнт вагомості і–го параметра; B_i – оцінка і–го параметра в балах.

Таблиця 3.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3.6	0.215	0.774
F2(X3)	А, Б	800	2.4	0.348	0.835
F2(X4)	А	1800	2	0.154	0.308
	Б	1200	8	0.154	1.232
F3(X2)	Б	16	3.4	0.283	0.962

За даними з таблиці 3.6 за формулою

$$K_K = \Sigma_{i=1}^{z} K_{TY}[F_{ik}]$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 0.774 + 0.835 + 0.308 + 0.962 = 2.879$$

$$K_{K2} = 0.774 + 0.835 + 1.232 + 0.962 = 3.803$$

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

3.1.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

При цьому варіант 3 має додаткове завдання:

3. Реалізація методів аналізу;

А варіант 4 має інше додаткове завдання:

4. Обробка інтерфейсу готових бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P * K_P * K_{CK} * K_M * K_{CT} * K_{CT.M} \tag{3.2}$$

де T_P – трудомісткість розробки ПП; K_P – поправочний коефіцієнт; K_{CK} – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; K_{CT} – коефіцієнт використання стандартних модулів і прикладних програм; $K_{CT.M}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: КП = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $K_{CK} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{CT} = 0.8$. Тоді, за формулою 3.2, загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 * 1.7 * 0.8 = 122.4 \text{ людино-днів}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто $T_P = 27$ людино-днів, $K_P = 0.9$, $K_{CK} = 1$, $K_{CT} = 0.8$:

$$T_2 = 27 * 0.9 * 0.8 = 19.44 \text{ людино-днів}$$

Для третього завдання (використовується алгоритм другої групи складності, ступінь новизни Г з використанням перемінної інформації):

$$T_P = 12 \text{ людино-днів}$$

$$K_P = 0.72$$

$$K_{CT} = 0.8$$

$$T_O = 12 * 0.72 * 0,8 = 6.91$$

Для четвертого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$$T_P = 8 \text{ людино-днів}$$

$$K_P = 0.6$$

$$K_{CT} = 1$$

$$T_O = 8 * 0.6 * 1 = 4.8$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122.4 + 19.44 + 6.91)8 = 1190 \text{ людино-годин}$$

$$T_{II} = (122.4 + 19.44 + 4.8)8 = 1173.12 \text{ людино-годин}$$

Найбільш високу трудомісткість має варіант I.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000 грн. Визначимо зарплату за годину за

формулою:

$$C_h = \frac{M}{T_m * t} \text{ грн},$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$C_h = \frac{6000 + 6000 + 9000}{3 * 21 * 8} = 41,67 \text{ грн}$$

Тоді, розрахуємо заробітну плату за формулою

$$C_{Cl} = C_h * T_i * K_D$$

де C_h – величина погодинної оплати праці програміста; T_i – трудомісткість відповідного завдання; K_D – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $C_{Cl} = 41.67 * 1190 * 1.2 = 59504.76$ грн.

II. $C_{Cl} = 41.67 * 1173.12 * 1.2 = 58660.69$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $C_{-} = C_{Cl} * 0.22 = 59504.76 * 0.22 = 13091.05$ грн.

II. $C_{-} = C_{Cl} * 0.22 = 58660.69 * 0.22 = 12905.35$ грн.

Тепер визначимо витрати на оплату однієї машино-години (C_M)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_G = 12 * M * K = 12 * 6000 * 0.2 = 14400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{Cl} = C_G * (1 + K_{Cl}) = 14400 * (1 + 0.2) = 17280 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$C_{-} = C_{Cl} * 0.22 = 17280 * 0.22 = 3801.6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$$C_A = K_{TM} * K_A * C_{PC} = 1.15 * 0.25 * 8000 = 2300 \text{ грн.},$$

де K_{TM}– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; K_A – річна норма амортизації; C_{PC} – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} * C_{PC} * K_P = 1.15 * 8000 * 0.05 = 460 \text{ грн.},$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу P_K за рік розраховуємо за формулою:

$$T_{EF} = (D_C - D_H - D_{Cl} - D_F) * t * C_U =$$

$$= (365 - 104 - 8 - 16) * 8 * 0.9 = 1706.4 \text{ годин},$$

де D_C – календарна кількість днів у році; D_H, D_{Cl} – відповідно кількість вихідних та святкових днів; D_F – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; C_U– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{EL} = T_{EF} * N_C * C_B * T_E = 1706.4 * 0.156 * 0.2 * 2.0218 = 107.6 \text{ грн.},$$

де N_C – середньо-споживча потужність приладу; C_B– коефіцієнтом зайнятості приладу; T_E – тариф за 1 КВт-годин електроенергії.

Розраховуємо накладні та річні експлуатаційні витрати:

$$C_N = C_{PP} * 0.67 = 8000 * 0.67 = 5360 \text{ грн.}$$

$$_ex = C_{Cl} + C_- + C_A + C_P + C_{EL} + C_N$$

$$_ex = 17280 + 3801.6 + 2300 + 460 + 107.64 + 5360 = 29309.24 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{MH} = {}_{ex}/T_{EF} = 29309.24/1706.4 = 17.18 \text{ грн/час.}$$

Роботи, пов‘язані з розробкою ПП ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від варіанта реалізації, складають:

$$C_M = C_{MH} * T$$

I. $$C_M = 17.18 * 1190 = 16444.2 \text{ грн.;}$$

II. $$C_M = 17.18 * 1173.12 = 16154.2 \text{ грн.;}$$

Накладні витрати складають 67% від заробітної плати:

$$C_N = C_{Cl} * 0.67$$

I. $$C_N = 53504.76 * 0.67 = 35868.19 \text{ грн.;}$$

II. $$C_N = 52660.69 * 0.67 = 35302.66 \text{ грн.;}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{PP} = C_{Cl} + C_- + C_M + C_N$$

I. $$C_{PP} = 59504.76 + 13091.05 + 16444.2 + 35868.19 = 124908.2 \text{ грн}$$

II. $$C_{PP} = 58660.69 + 12905.35 + 16154.2 + 35302.66 = 123022.9 \text{ грн}$$

3.1.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{TELj} = K_{Kj}/C_{Fj}$$

$$K_{TEL1} = 2.879/124908.2 = 2.30e-5$$

$$K_{TEL2} = 3.803/123022.9 = 3.09e-5$$

Більш ефективним є другий варіант реалізації: $K_{TEL2} = 3,09e-5$.

3.2 Висновки

Проведено функціонально-вартісний аналіз ПП, розробленого в рамках дипломного проекту. Процес аналізу можна розділити на частини.

В першій проведено дослідження ПП з технічної точки зору: визначено функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, експертних оцінок їх важливості обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

В другій обрано найбільш економічно обґрунтовану реалізацію. Порівняння варіантів виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання ФВА, можна зробити висновок, що оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $3,09e-5$.

Цей варіант реалізації програмного продукту має такі параметри:

- IDE — Intellij IDEA
- мова програмування – Java;
- використання бібліотек JavaFX;

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Опис моделі

Для дослідження ефективності роботи контекстно-залежної системи при застосуванні в ній ряду методів усунення технічних конфліктів було побудовано модель відповідної системи.

На об’єктно-орієнтованій мові програмування Java було розроблено програмний засіб, що моделює поведінку контекстно-залежної системи. Програму було складено з наступних модулів:

- MVC шаблон проектування;
  - модель;
    - середовище;
    - компонентний модуль;
      - модуль ресурсів;
      - модуль сенсорів;
      - модуль обробників;
    - модуль усунення конфліктів між компонентами;
  - графічний модуль;
    - засоби відображення стану середовища та системи в цілому;
    - засоби модифікації та відображення даних про компоненти системи, що містяться в середовищі;
  - контролер моделі та графічного модулю.

Шаблон mvc було використано для відокремлення графічного модулю від безпосередньо моделі. Завдяки цьому програмний засіб може бути портований на іншу платформу без додаткових модифікацій в моделі, тобто принципи її функціонування при тих самих введених параметрах

змінено не буде. Окрім цього графічний модуль може бути замінено на інтерфейс командного рядку чи веб-інтерфейс. В результаті застосування шаблону отриманий програмний продукт може бути охарактеризовано як кросплатформенний.

Модуль моделі складається з класу середовища, компонентів контекстно-залежної системи та модулю усунення конфліктів.

4.1.1 Середовище

При налаштування програми середовищу може бути надано ряд параметрів, які є екземплярами окремого класу. Кожний параметр системи може бути інерційним або миттєвим.

Значення інерційних параметрів змінюються поступово, залежать від їх значень в попередній момент часу. Прикладами інерційних характеристик можуть бути температура, вологість, швидкість певних частин системи.

Значення миттєвих параметрів характеризують конкретний момент часу і не залежать від попередніх значень, а визначаються тільки початковим станом середовища та параметрами ресурсів системи. Прикладами миттєвих характеристик можуть бути освітленість, атмосферний тиск (лише в деяких системах), спад напруги на компоненті електромережі.

Саме середовище володіє властивістю інерційності, що означає, що значення параметрів середовища будуть намагатися повернутися до початкових.

Для інерційних параметрів це відбувається із швидкістю пропорційною різниці початкового значення параметру та значення цього параметру в поточний момент часу. Інтенсивність цього процесу може бути змінена в налаштуваннях програми.

Для миттєвих параметрів їх значення буде повертатися в початкове одразу після зупинки ресурсів, що ці значення змінювали. Поточне значення миттєвих параметрів розраховується як сума їх початкових значень та їх сума значень для всіх активних ресурсів системи.

4.1.2 Компоненти

Компоненти системи поділяються на три типи:

- ресурси;
- сенсори;
- обробники;

Кожен з них представлений в програмі екземпляром відповідного класу. Кожен з класів компонентів наслідує або використовує клас, що містить в собі набір параметрів, що відповідає набору параметрів середовища.

4.1.2.1 Ресурси

Ресурсом системи є такий компонент, який може змінювати значення параметрів середовища протягом часу. Ресурс має унікальний ідентифікатор, в якості якого використовується його ім’я, що задає користувач програми, коли додає новий ресурс в систему.

Ресурс може знаходитися в одному з двох станів: активному та неактивному. В активному стані ресурс використовує енергію системи та змінює значення параметрів середовища. Кількість енергії, що була використана ресурсом в подальшому використовується для оцінки ефективності роботи системи.

В активному стані ресурс змінює параметри середовища на певну величину, яка задається для кожного параметра користувачем перед запуском системи. Набір параметрів ресурсу відповідає набору параметрів

системи. Значення інерційних параметрів середовища змінюються на значення відповідного параметру ресурсу за кожним сигналом часу, що генерується контролером. Значення миттєвих параметрів середовища змінюється на значення відповідних параметрів ресурсу при його активації та на протилежне значення при його деактивації.

Ресурси можуть створюватися та налаштовуватися лише до запуску системи. Активація та деактивація ресурсів виконується обробниками, що мають доступ до цих ресурсів відповідно до рішень, прийнятих на основі показників сенсорів.

4.1.2.2 Сенсори

Сенсором є такий компонент системи, який в кожен момент часу визначає значення параметрів середовища та передає їх обробникам, які мають доступ до даного сенсору.

Набір параметрів, значення яких в середовищі визначає сенсор, відповідає набору параметрів самого середовища.

Сенсор має унікальний ідентифікатор, в якості якого було використано його ім’я, яке задає користувач при створенні сенсору.

Сенсор може бути розміщено так, що якийсь з ресурсів буде чинити влив на покази сенсору, що впливатиме й на рішення, що приймають обробники на основі показів сенсору. Ресурс та ступінь його впливу задаються користувачем при створенні нового сенсору в системі.

В покази сенсора вноситься випадкова похибка, яка залежить від точності сенсору. Точність задається у вигляді максимального значення похибки користувачем при створенні сенсору.

Сенсори можуть створюватися та налаштовуватися лише до початку роботи системи. Сенсор продовжує функціонувати весь час, не використовуючи при цьому енергії системи. Сенсор не може бути

деактивовано відповідно до якогось з методів усунення конфліктів, так як при цьому він припинить знімати покази з середовища і не існуватиме підстав для подальшої активації сенсору, внаслідок чого конфігурацію системи буде незворотньо змінено, що зменшить її ефективність.

4.1.2.3 Обробники

Обробником є такий компонент системи, який має доступ до ряду сенсорів та ресурсів системи. Обробник має доступ принаймні до одного сенсору та одного ресурсу.

Обробник має унікальний ідентифікатор, в якості якого використовується його ім’я, що задається користувачем при створенні нового обробника.

Обробник може бути створено лише до запуску системи.

Обробник має набір параметрів, що відповідають параметрам середовища. Набір задає вимогу до значень відповідних параметрів середовища; значення параметрів набору встановлюється користувачем при створенні обробника.

На основі значень параметрів вимог до середовища та показів сенсорів, до яких обробник має доступ, він визначає конфігурацію ресурсів, які слід активувати або деактивувати для встановлення значень параметрів середовища відповідно до вимоги, заданій при створенні обробника.

Обробник не використовує енергії системи та продовжує роботу протягом всього часу роботи системи.

Саме з обробником пов’язані основні типи технічних конфліктів, що виникають в системі.

4.1.3 Модуль усунення конфліктів

4.1.3.1 Джерела конфліктів

Джерелами конфліктів є набори компонентів системи, результати роботи який суперечить один одному.

Серед технічних конфліктів в змодельовані системі можна виділити основні групи класифікації за джерелом виникнення:

- конфлікт показів сенсорів, до яких має доступ один з обробників;
- конфлікт параметрів ресурсів, до яких має доступ один з обробників;
- конфлікт параметрів ресурсів, до яких не має одночасного доступу жоден з обробників;
- конфлікт вимог обробників, що мають спільні ресурси;
- конфлікт вимог обробників, що не мають спільних ресурсів;

Конфлікт показів сенсорів, до яких має доступ один з обробників виникає всередині спільного обробника. Конфлікт виникає через те, що різні сенсори можуть бути розміщені біля різних ресурсів, що можуть бути активними, що систематично викривлятиме покази. Окрім цього на покази сенсорів впливає випадкова похибка, що змінює значення в кожен момент часу. В результаті цього покази сенсорів відрізнятимуться і виникне конфлікт, в результаті якого не існувати однозначності щодо того, які покази сенсорів використовуватимуться в обробнику. Для даного конфлікту усунення полягає в виключенні з набору тих сенсорів, які конфлікт спричиняють. Після цього в наборі залишаються сенсори , між показами яких конфлікти відсутні, і покази для прийняття рішення обробником можуть бути обрані.

Конфлікт параметрів ресурсів, до яких має доступ один з обробників, виникає кожного разу при прийняття рішення обробником,

якщо хоча б в двох ресурсів, до яких має доступ цей обробник, відрізняється знак значення хоча б одного параметра. При активації обох ресурсів обробником, вплив кожного з ресурсів на параметр середовища буде компенсуватися іншим ресурсом.

Для усунення конфлікту не застосовується той вид методів, що розглядаються в роботі. Натомість, в самому обробнику реалізовано алгоритм оцінки ефективності роботи певної комбінації ресурсів. При цьому, в деяких випадках в результаті роботи алгоритму можуть бути активовані обидва ресурси для послаблення впливу на параметр середовища, значення якого в ресурсів мають різний знак, і для посилення впливу на інші параметри середовища.

Конфлікт параметрів ресурсів, до яких не має одночасного доступу жоден з обробників виникає в тому випадку, коли в моделі наявні два ресурси, що були активовані різними обробниками, параметри яких значно відрізняються. При цьому вплив на середовище одного з ресурсів буде компенсуватися іншим. Як результат, мета, з якою ресурси були активовані обробниками, досягнута не буде, я активовані ресурси використовуватимуть енергію системи, знижуючи ефективність її роботи. Для усунення конфлікту один з ресурсів має бути деактивовано та тимчасово заблоковано на час роботи іншого ресурсу. Обробник при наступному прийнятті рішення почне пошук заміни для деактивованого ресурсу.

Конфлікт вимог обробників, що мають спільні ресурси виникає в тому випадку, коли обробники, приймають різне рішення щодо активності спільного ресурсу на основі різного набору датчиків або різних вимог до параметрів середовища. Подальший стан ресурсу залежить від методу усунення конфлікту, що було застосовано до обробників.

Конфлікт вимог обробників, що не мають спільних ресурсів виникає при паралельній роботі двох обробників, що мають різні вимоги до параметрів середовища. Якщо конфлікт не буде розв’язано, він призведе до виникнення ряду конфліктів параметрів ресурсів, до яких не має одночасного доступу жоден з обробників, кожен з яких вимагатиме окремого застосування одного з методів усунення конфлікту, що може збільшити ефективність роботи системи, але вимагатиме більшої кількості застосування модулю усунення конфліктів.

4.1.3.2 Усунення конфліктів

В модулі було застосовано дві основні групи методів усунення конфліктів:

- прості методи;
- методи Quolity of Context;

Застосування простих методів передбачає вилучення елементів з конфліктної групи на основі їх порядку надходження до неї і незалежно від інформації, яку елементи несуть.

Методи Quolity of Context працюють з компонентами контекстно-залежної системи в залежності від інформації про контекст, яку вони несуть. Детально методи цієї групи були розглянуті в розділі роботи 1.2.1QoC-орієнтовані політики усунення конфліктів.

Реалізовані в моделі методи усунення конфліктів наведено в таблицях 4.1 та 4.2.

Таблиця 4.1 - Реалізовані прості методи усунення конфліктів

Назва методу	Drop last	Drop firts	Drop all	User action*
Відбір компонентів	Послідовне вилучення, FILO	Послідовне вилучення, FIFO	Вилучення всіх	Всі
Умова завершення роботи	Група втрачає ознаку конфліктної		Група порожня	Користувач ввів рішення для всіх компонентів
Опис	Вилучені компоненти деактивуються або ігноруються; інші - усереднюються або виконуються/використовуються послідовно			Рішення приймає користувач

* метод User action не застосовувався при порівнянні конфліктів в даній роботі через значний час, що необхідний для вирішення конфліктів, але він може бути обраний при використанні програмного засобу для моделювання реальної системи.

Таблиця 4.2 - Реалізовані Quolity of Context методи усунення конфліктів

Назва методу	QoC Completeness	QoC Significance	QoC Trustworthiness	QoC UpToDatedness
Відбір компонентів	За спаданням значення відповідного виразу таблиці 1.2			
Умова завершення роботи	Вибрано один або кілька компонентів			
Опис	Всі відібрані компоненти виконуються/використовуються послідовно			

Для застосування описаних методів усунення конфліктів в моделі в програмному засобі було реалізовано шаблон проектування Command. Було створено абстрактний клас Policy, з методом resolve, що має бути перевизначено для класів-нащадків даного класу. Метод приймає на вході масив посилань на компоненти системи, утворює з них конфліктні групи та застосовує метод усунення конфліктів, визначений для даного класу.

Окремі методи представлені унікальними екземплярами анонімних класів, що наслідуються від абстрактного класу Policy, та зберігаються в

структурі даних TreeMap, що забезпечує доступ до об’єктів методів за ключем, яким є назва методу.

4.1.4 Графічний модуль

Основним об’єктом, що моделюється за допомогою програмного засобу є контекстно-залежна система з середовищем, яке характеризується рядом параметрів. Через це постало питання про зручне відображення проміжних та остаточних результатів роботи програмного засобу в поточній сесії.

Програмний засіб використовується в даній роботі як інструмент для вивчення ефективності роботи системи при застосуванні в ній різних методів усунення конфліктів, отже було прийняте рішення про надання програмному засобу графічного інтерфейсу, що дозволить представити результати моделювання в роботі в зручному для сприйняття вигляді.

Графічний інтерфейс користувача було реалізовано у вигляді програмного модулю, що використовує засоби бібліотеки javafx.

Основним об’єктом графічного інтерфейсу є діаграма, на якій відображені значення параметрів середовища змодельованої системи в усі моменти часу, від початку роботи системи, закінчуючи поточним моментом, для якого розраховані параметри середовища.

При виборі кількох параметрів середовища на діаграмі будуть відображені всі вони; кожному відповідатиме набір точок певного кольору.

Графіки будуються на спільних координатних осях, що дозволяє порівняти значення параметрів системи в певний момент часу; співставити стрибки відповідних графіків в момент зміни стану активності ресурсів системи.

В кожен момент часу моделі на графік параметра наноситься нова точка, що відповідає значенню даного параметра в середовищі системи. При призупиненні відліку часу в моделі нові точки не наносяться на графік, що надає можливість проаналізувати поточний стан середовища.

Панель керування системою розміщено в лівій частині вікна програми. Панель включає в себе

- кнопку виходу з програми;
- кнопку запуску системи;
  - після запуску системи: кнопка призупинення/продовження відліку часу в моделі;
- панель виведення інформації про ефективність системи:
  - дисплей використання енергії системою;
  - дисплей ступені відповідності стану середовища характеристикам, введеним користувачами системи;
- панелі редагування та відображення компонентів системи:
  - панель ресурсів;
  - панель сенсорів;
  - панель обробників;

Кожна з панелей компонентів може бути прихована. Панелі містять список відповідних компонентів, по кожному з яких може бути отримана детальна інформація шляхом розгортання елементу списку. Верхнім елементом кожного списку компонентів є панель для введення параметрів нового компоненту системи; панель може бути прихована.

Елементи списку ресурсів мають червоний колір, якщо ресурс деактивовано, і зелений, якщо ресурс активовано.

4.1.5 Контролер

Для керування модулями графічного інтерфейсу та моделі, їх ініціалізації та забезпечення зв’язку між ними було створено модуль контролеру.

В модулі наявний метод ініціалізації моделі початковими значеннями, що було використано для отримання результатів моделювання системи з кожним з розглянутих в роботі методів усунення конфліктів для різних параметрів конфігурації системи.

В контролері задаються обробники подій елементів керування графічного інтерфейсу. При введені користувачем параметрів нового компоненту та натисканні на кнопку його створення, контролер отримує введені параметри, виконує їх перевірку та перетворення в дані для моделі та передає їх в її відповідний метод, де ініціюється збірка нового компоненту системи. Після створення компоненту контролер робить запит списку компонентів системи в моделі та передає його в відповідний метод графічного модулю, де дані компоненту відображаються на відповідній панелі у вигляді інформації про компонент.

Після запуску системи контролер генерує сигнали часу, кожен з яких переводить модель в новий стан, для якого розраховуються нові значення параметрів системи. Після виконання розрахунків контролер робить в модель запит параметрів середовища та передає отримані дані у відповідний метод графічного інтерфейсу, де вони відображаються на діаграмі у вигляді графіків, які відповідають переданим параметрам, а також у вигляді поточного стану кожного з ресурсів.

4.2 Тестування моделі

Для перевірки функціонування моделі було виконано ряд тестів. Результати виконання зі знімками екрану, що їх підтверджують, наведено в таблиці 4.3

Таблиця 4.3 - Результати виконання тестів

Тест	Знімок	Результат
Успішний запуск програмного продукту	рисунок Error: Reference source not found	Пройдено
Наявність та коректність розташування елементів графічного інтерфейсу	рисунок 4.2	Пройдено
Розгортання та приховування панелі компонентів	рисунок 4.3	Пройдено
Додавання нового компоненту до системи;	рисунок 4.4	Пройдено
Відображення та приховування детальної інформації про компонент;	рисунок 4.5	Пройдено
Запуск описаної системи;	рисунок 4.6	Пройдено
Призупинення та продовження відліку часу системи;	рисунок 4.7	Пройдено
Зміна статусу ресурсу протягом часу;	рисунок 4.8	Пройдено
Відповідність між статусом ресурсу та графіком параметру середовища, на який ресурс впливає;	рисунок 4.9	Пройдено
Утримання обробником параметрів середовища на рівні, найбільш наближеному до встановлених вимог;	рисунок 4.10	Пройдено

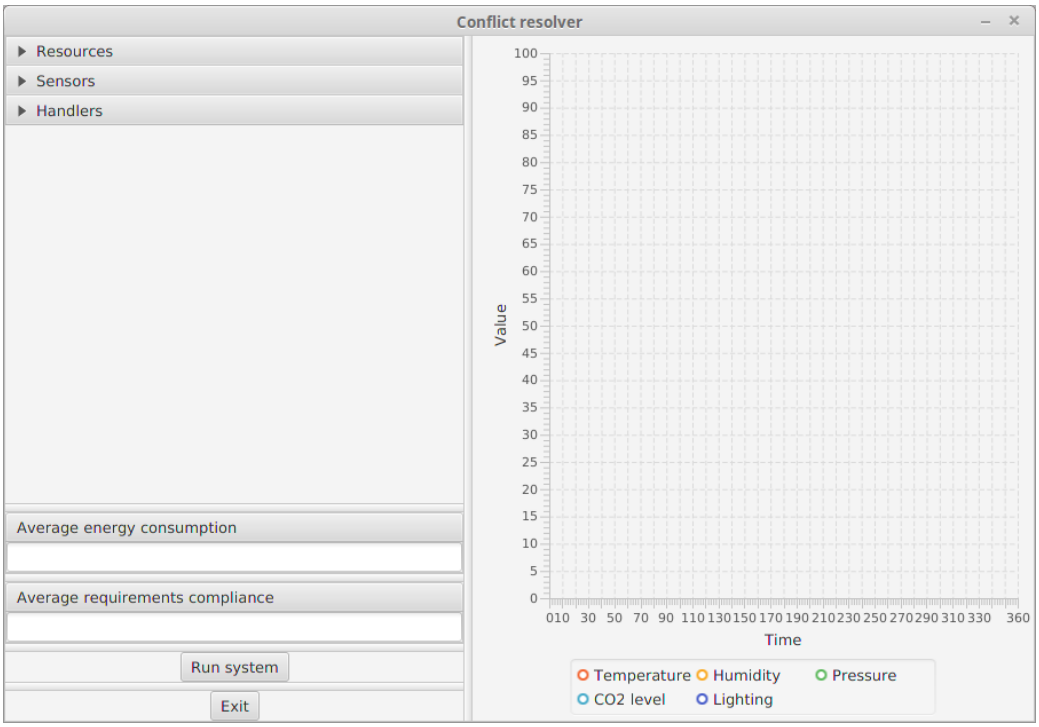

Рисунок 4.1 - Успішний запуск програмного продукту

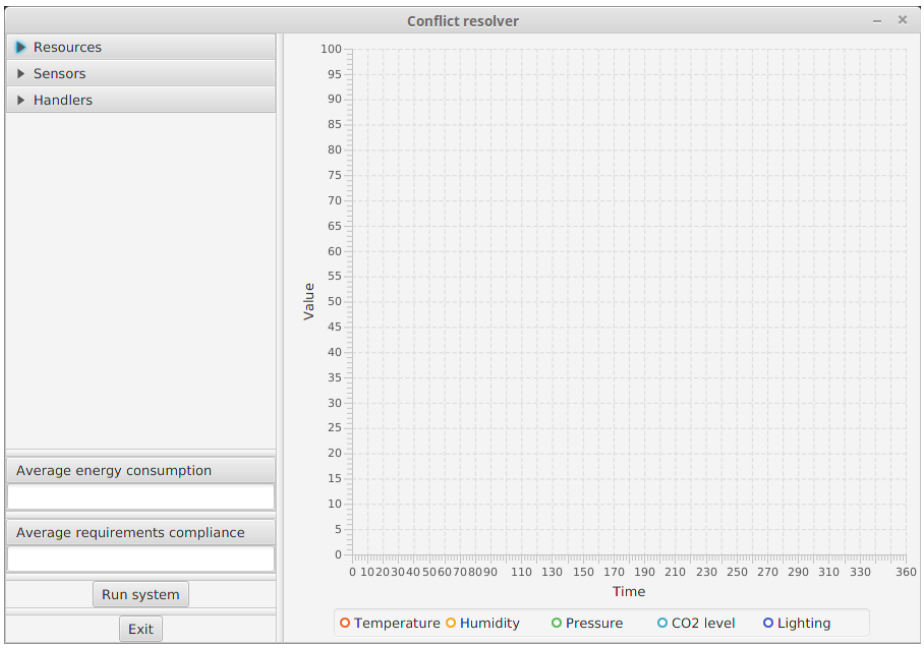

Рисунок 4.2 - Наявність та коректність розташування елементів графічного інтерфейсу

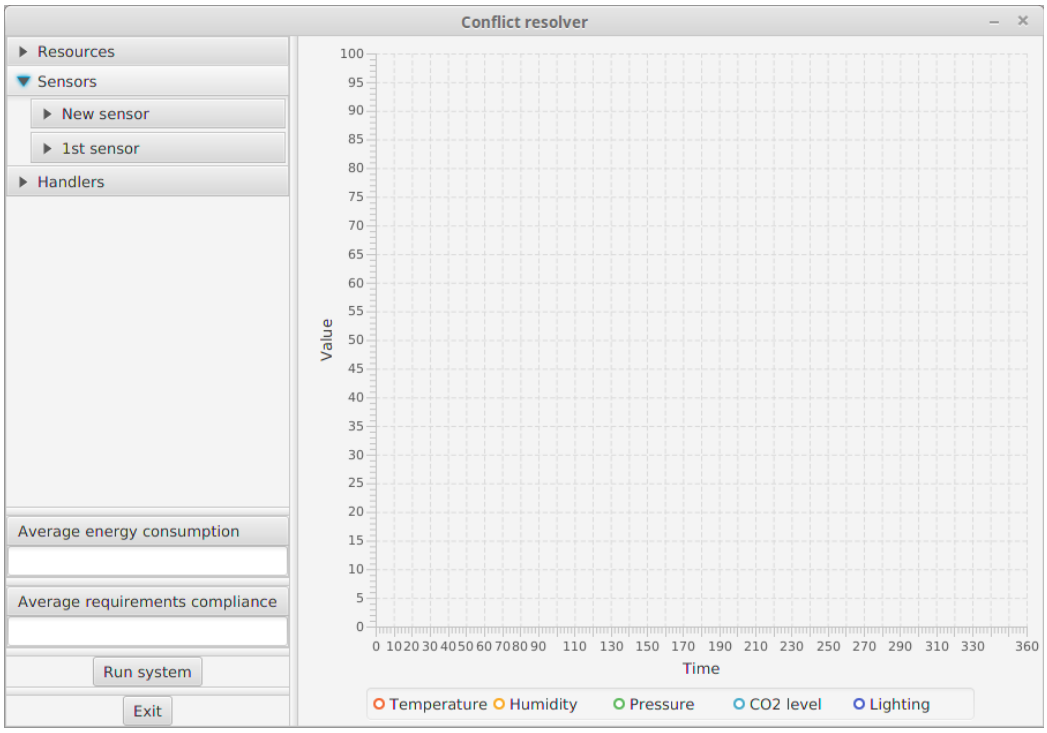

Рисунок 4.3 - Розгортання та приховування панелі компонентів

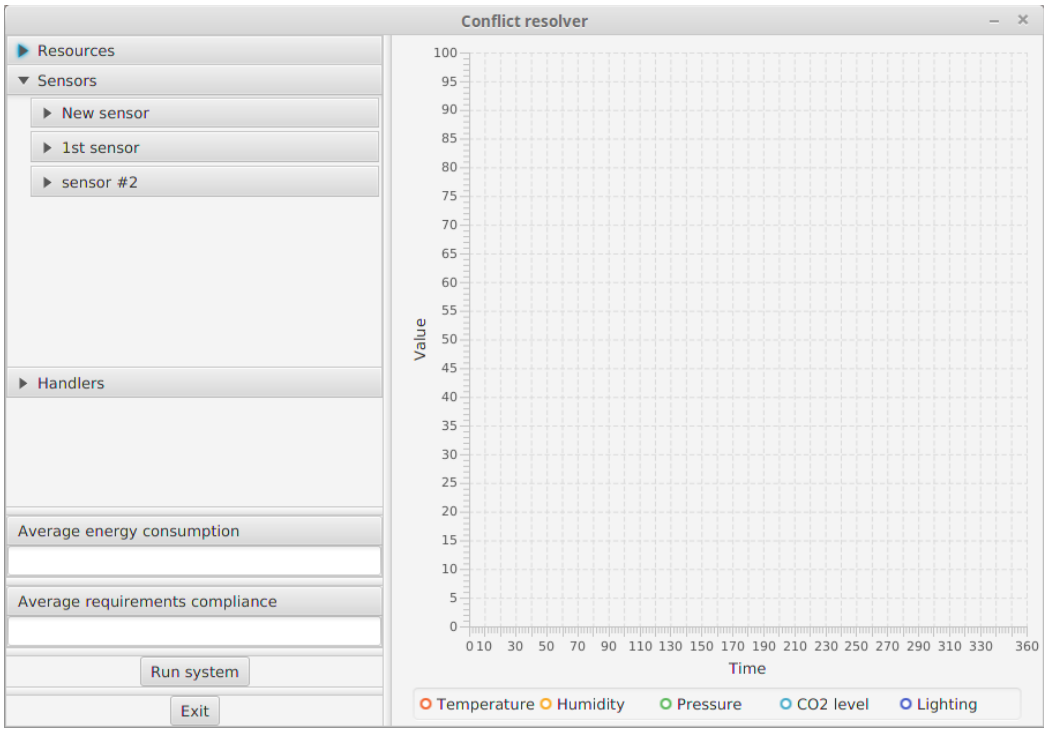

Рисунок 4.4 - Додавання нового компоненту до системи

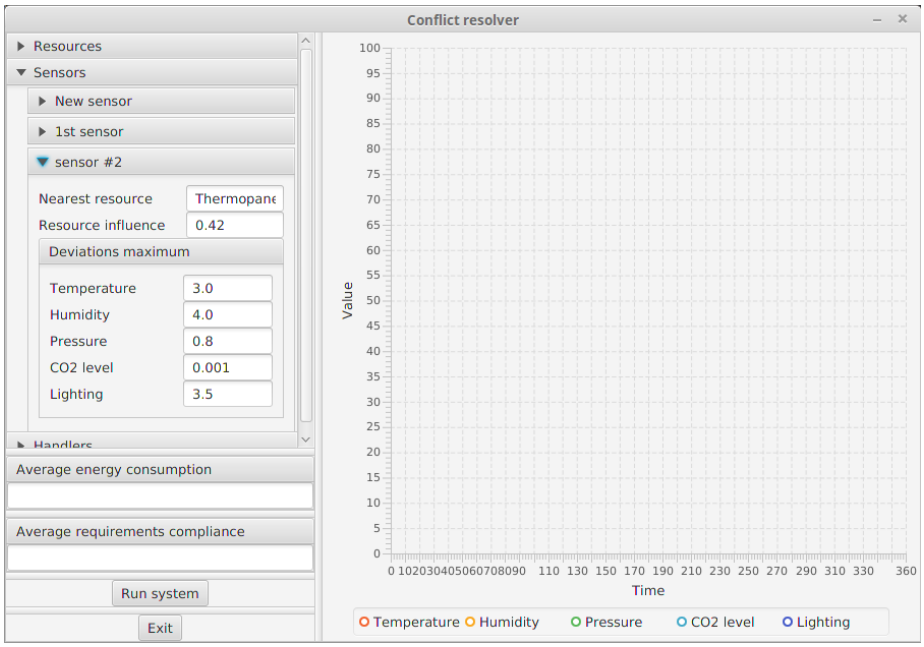

Рисунок 4.5 - Відображення та приховування детальної інформації про компонент

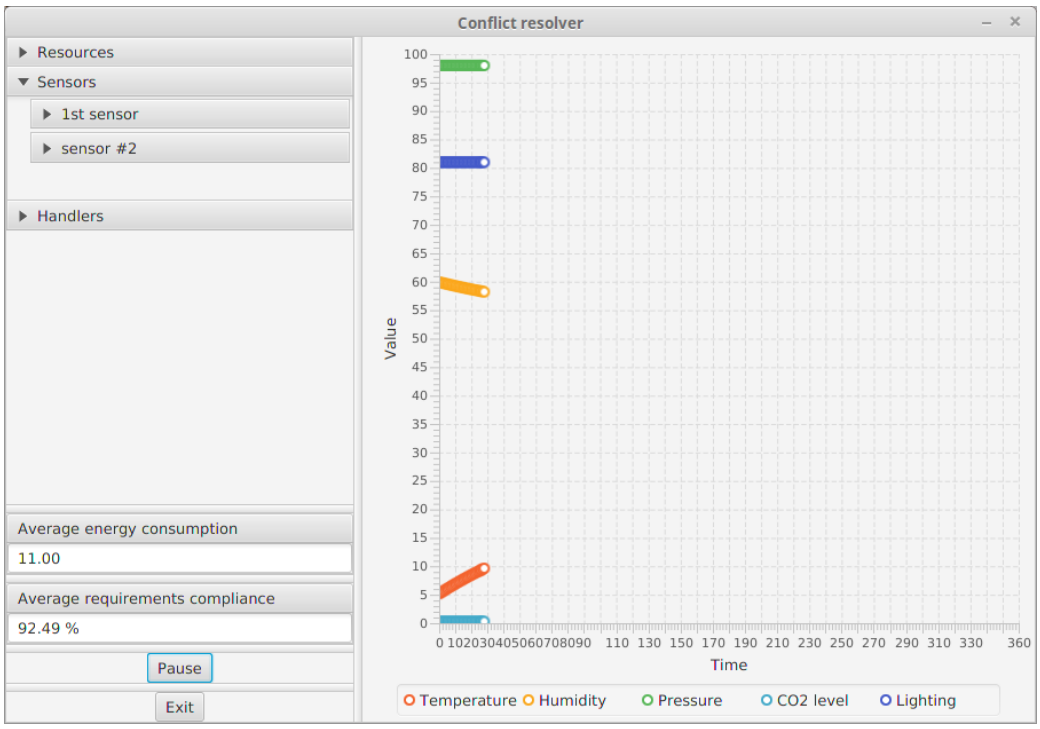

Рисунок 4.6 - Запуск описаної системи

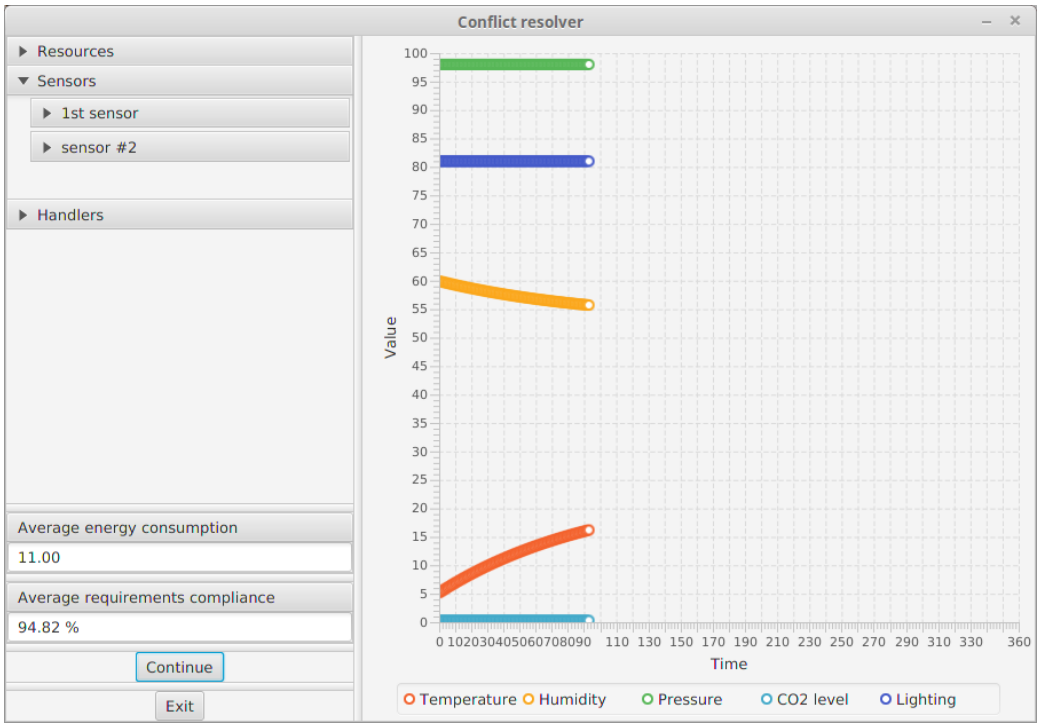

Рисунок 4.7 - Призупинення та продовження відліку часу системи

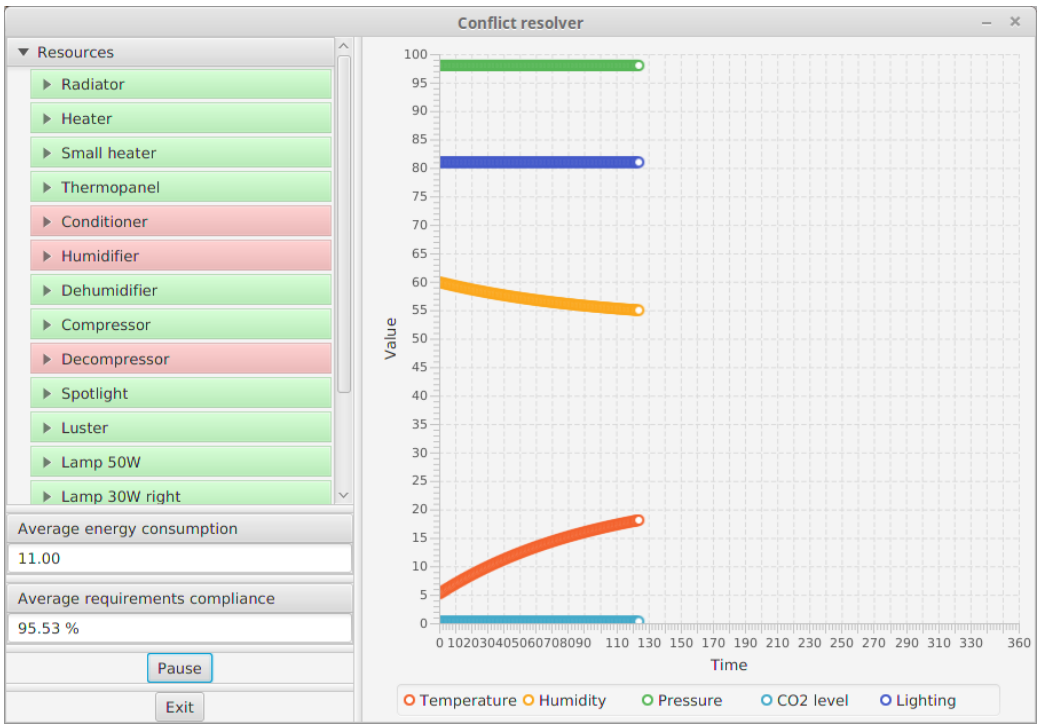

Рисунок 4.8 - Зміна статусу ресурсу протягом часу

Рисунок 4.9 - Відповідність між статусом ресурсу та графіком параметру середовища, на який ресурс впливає

Рисунок 4.10 - Утримання обробником параметрів середовища на рівні, найбільш наближеному до встановлених вимог

4.3 Результати роботи з моделлю

За допомогою розробленого програмного засобу було застосовано методи усунення конфліктів, розглянуті в пункті 4.1.3.2 до системи, параметри компонентів якої було встановлено таким чином, щоб переважав один з типів конфліктів, розглянутих в пункті 4.1.3.1.

Результати проведення кожної з серій дослідів наведено в таблицях 4.4 - 4.7.

Таблиця 4.4 - Конфлікт показів сенсорів, до яких має доступ один з обробників

Метод	Більш впливовий тип помилки сенсора	Відповідність середовища вимогам при максимальній кількості параметрів, які оцінює сенсор	
		2 параметри	4 параметри
Use all	Вплив ресурсу	43%	36%
	Випадкова помилка	46%	53%
Drop all	Вплив ресурсу	52%	47%
	Випадкова помилка	41%	44%
Drop last	Вплив ресурсу	58%	52%
	Випадкова помилка	53%	59%
Drop firts	Вплив ресурсу	56%	50%
	Випадкова помилка	56,00%	57%
QoC Completeness	Вплив ресурсу	62%	67%
	Випадкова помилка	68%	76%
QoC Significance	Вплив ресурсу	67%	61%
	Випадкова помилка	63%	56%
QoC Trustworthiness	Вплив ресурсу	54%	51%
	Випадкова помилка	83%	87%
QoC UpToDatedness	Вплив ресурсу	49%	46%
	Випадкова помилка	53%	43%

Таблиця 4.5 - Конфлікт параметрів ресурсів, до яких не має одночасного доступу жоден з обробників

Метод	Максимальна кількість параметрів системи, що змінюються ресурсом	Оцінка продуктивності модельованої системи	
		Використання енергії	Відповідність встановленим вимогам
Use all	4	16	23%
	2	16	32%
Drop all	4	2	31%
	2	3	37%
Drop last	4	4	35%
	2	5	41%
Drop firts	4	4	31%
	2	5	39%
QoC Completeness	4	2	81%
	2	4	72%
QoC Significance	4	2	73%
	2	4	69%
QoC Trustworthiness	4	2	34%
	2	4	38%
QoC UpToDatedness	4	2	61%
	2	4	65%

Для кожної комбінації методу усунення конфліктів та переважного типу конфліктів в системі проведено серію з 5 дослідів із незначною варіацією параметрів компонентів системи. Так як дисперсія результатів однієї серії не виявилася значною, результати кожної серії усереднено.

Аналізуючи отримані результати, можна зробити висновки, що QoC методи усунення конфліктів не можна вважати більш ефективними за найпростіші методи, так як в загальному випадку об’єкти, між якими

виникає конфлікт, можуть не володіти параметрами за якими виконується оцінка об’єкту QoC методами.

Проте в частинних випадках QoC зазвичай виявляється більш ефективним, так як для певного типу конфлікту, об’єкти, що його створюють, можуть бути оцінені за одним або кількома критеріями, що піддаються аналізу в методі QoC.

Таблиця 4.6 - Конфлікт вимог обробників, що мають спільні ресурси

Метод	Максимальна кількість параметрів системи, що змінюються ресурсом	Оцінка продуктивності модельованої системи	
		Використання енергії	Відповідність встановленим вимогам
Use all	4	14	25%
	2	19	49%
Drop all	4	5	67%
	2	7	72%
Drop last	4	7	61%
	2	8	63%
Drop firts	4	7	59%
	2	8	62%
QoC Completeness	4	3	86%
	2	5	91%
QoC Significance	4	2	64%
	2	2	69%
QoC Trustworthiness	4	3	36%
	2	3	59%
QoC UpToDatedness	4	3	75%
	2	5	86%

В даному випадку було визначено, що при виникненні конфлікту між показами сенсорів, найкращі результати дають такі методи, як QoC Completeness та QoC Trustworthiness, в той час як при конфлікті між ресурсами системи більш ефективними виявляються методи QoC Completeness та QoC Significance.

Звичайно, це пов’язано і з тим, які параметри було надано компонентам при ініціалізації системи. Так, наприклад, при введенні в сенсор підвищеної затримки, актуальним був би метод QoC UpToDatedness.

Таблиця 4.7 - Конфлікт вимог обробників, що не мають спільних ресурсів

Метод	Максимальна кількість параметрів , що змінюються ресурсом	Оцінка продуктивності модельованої системи	
		Використання енергії	Відповідність встановленим вимогам
Use all	4	17	21%
	2	23	35%
Drop all	4	7	71%
	2	11	74%
Drop last	4	9	64%
	2	13	67%
Drop firts	4	9	62%
	2	11	66%
QoC Completeness	4	4	91%
	2	7	93%
QoC Significance	4	3	71%
	2	3	74%
QoC Trustworthiness	4	4	51%
	2	5	63%
QoC UpToDatedness	4	3	83%
	2	4	89%

Таким чином, є сенс у застосуванні в системі, як найпростіших методів усунення конфліктів, так і QoC методів. Перші дають показник ефективності системи, вищий за середній незалежно від типу конфлікту. Другі дають вищий показник ефективності системи, але лише для певних типів конфліктних компонентів. Тому в контекстно-залежній системі, що має різні типи потенційно конфліктних компонентів, що добре піддаються оцінюванню за одним з параметрів QoC, можуть бути застосовані відповідні методи. В інших об’єктах доцільно буде застосувати методи загального спрямування, так як метод QoC за параметром, не характерним для об’єкту, знизить ефективність системи.

4.4 Висновки

В розділі описано структуру програмного продукту, який було створено під час виконання дипломної роботи. Описано типи конфліктів, які можуть бути спричинені в контекстно-залежній системі за допомогою програмного продукту. Проведено досліди із застосуванням в системі методів усунення конфліктів і досліджено ефективність роботи системи в залежності від переважного типу конфліктів в системі. Результати дослідів проаналізовано, підведено підсумки.

ВИСНОВКИ

Виконаний в роботі аналітичний огляд дозволяє визначити особливості контекстно-залежних систем та методів усунення конфліктів, які в них застосовуються.

Було розглянуто класифікацію конфліктів за характеристиками контексту, в якому конфлікти виникають та приведено приклади таких конфліктів в широко використовуваних контекстно-залежних системах.

Було розглянуто ряд методів усунення конфліктів, серед яких прості методи та методи, засновані на політиках Quolity of Context. Кожен з методів було описано та наведено приклади конфліктів, для яких застосування методу дає помітний результат.

На етапі постановки завдання було сформульовано мету роботи, окреслено її основні етапи, визначено засоби її виконання. Було визначено основні характеристики додатку, який має бути розроблено в ході виконання роботи.

Для дослідження впливу різних методів на ефективність роботи системи в залежності від її структури, було спроектовано програмний продукт. Було проведено дослідження програмного продукту з технічної точки зору: було визначено основні функції програмного продукту та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій програмного продукту.

Відповідно до прийнятих рішень, було затверджено основні кроки розробки програмного продукту з використанням відповідних

інструментів. Спроектований програмний продукт було реалізовано на об’єктно-орієнтованій мові програмування Java, що дозволяє використовувати його на різних платформах; для кращої взаємодії з користувачем та більш зручного представлення результатів, отриманих з програмного продукту, в роботі програмному продукту було надано графічний інтерфейс, побудований за допомогою засобів графічної бібліотеки JavaFX.

При проектуванні програмного продукту, в його основу було закладено шаблон проектування MVC, який розділяє модель та інтерфейс. В результаті застосування шаблону в програмному продукті може бути використано інший інтерфейс користувача без внесення змін в модель. Програмний продукт було розроблено за використанням методології Test-driven development. Було підтверджено його придатність до використання для досягнення поставлених в роботі цілей.

За допомогою програмного продукту було виконано ряд дослідів, результати яких представлено в роботі. Аналіз дослідів дозволив зробити висновки щодо ефективності використання різних груп методів усунення конфліктів в контекстно-залежній системі в залежності від її структури; доцільності закладення в системі певних методів усунення конфліктів для підвищення її продуктивності. Програмний продукт може бути використаний для моделювання контекстно-залежної системи певної структури та перевірки її функціонування до побудови, що дозволить внести в неї відповідні зміни на етапі проектування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Matthias Baldauf A survey on context-aware systems / Matthias Baldauf, Schahram Dustdar, Florian Rosenberg // Int. J. Ad Hoc Ubiquitous Comput.. – 2007. – № 2. – С. 263–277
2. Chang Xu On impact-oriented automatic resolution of pervasive context inconsistency / Chang Xu, S. C. Cheung, W. K. Chan, Chunyang Ye // In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. – 2007. – № 6. – С. 569–572
3. Anind K. Dey Designing mediation for context-aware applications / Anind K. Dey, Jennifer Mankoff // ACM Trans. Comput.-Hum. Interact.. – 2005. – № 12. – С. 53–80
4. Insuk Park A dynamic context-conflict management scheme for group-aware ubiquitous computing environments / Insuk Park, Dongman Lee, Soon J. Hyun // Proceedings of the 29th Annual International Computer Software and Applications Conference. – 2005. – № 1. – С. 359–364
5. Licia Capra Carisma: Context-aware reflective middleware system for mobile applications / Licia Capra, Wolfgang Emmerich, Cecilia Mascolo // IEEE Transactions on Software Engineering. – 2003. – № 29. – С. 929–945
6. Filip Perich On data management in pervasive computing environments / Filip Perich, Anupam Joshi, Timothy W. Finin, Yelena Yesha // IEEE Trans. Knowl. Data Eng.. – 2004. – № 16. – С. 621–634
7. Thomas Buchholz Quality of context: What it is and why we need it / Thomas Buchholz, Axel Kupper, Michael Schiffers // In Proceedings of the 10th International Workshop of the HP OpenView University

Association. – 2003. – № 10. – С. 1–14

8. Michael Krause Challenges in modeling and using quality of context (qoc) / Michael Krause, Iris Hochstatter // In Proceedings of Mobility Aware Technologies and Applications. – 2005. – № 3744. – С. 324–333
9. Diane J. Cook Making sense of sensor data / Diane J. Cook // IEEE Pervasive Computing. – 2007. – № 6. – С. 105–108
10. Alex Wun A system for semantic data fusion in sensor networks / Alex Wun, Milenko Petrovi, Hans-Arno Jacobsen // Proceedings of the 2007 inaugural international conference on Distributed event-based systems. – 2007. – № 7. – С. 75–79
11. Younghee Kim A quality measurement method of context information in ubiquitous environments / Younghee Kim, Keumsuk Lee // Proceedings of the 2006 International Conference on Hybrid Information Technology. – 2006. – № 6. – С. 576–581
12. Nadjia Kara Reasoning with contextual data in telehealth applications / Nadjia Kara, O. Andrei Dragoi // In Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. – 2007. – № 3. – С. 69
13. Atif Manzoor On the evaluation of quality of context / Atif Manzoor, Hong Linh Truong, Schahram Dustdar // In Proceedings of Third European Conference on Smart Sensing and Context, EuroSSC. – 2008. – № 5279. – С. 140–153

ДОДАТОК А

ЛІСТИНГ ВИХІДНОГО КОДУ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

```
import mvc.Controller;
import mvc.Model;
import mvc.ViewFX;

/**
 * Main class of Context Aware System
 *
 * @author pipich3
 * @version 1.0
 * @since 5/17/16
 */
public class ContextAwareSystem {

    public static void main(String[] args) throws InterruptedException {
        new Thread(() -> ViewFX.launch(ViewFX.class)).start();
        Model model = new Model();

        Controller controller = new Controller(model);
        controller.process();
    }
}
```

```
package mvc;

import component.evaluable.Evaluable;
import javafx.animation.Animation;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.util.Duration;
import mvc.fx_component.FXHandler;
import mvc.fx_component.FXResource;
import mvc.fx_component.FXSensor;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * ${NAME} class of mvc
 *
 * @author pipich3
 * @version 1.0
 * @since 5/23/16
 */
public class Controller {

    private static final double TIMER_PERIOD = 100;

    private Model model;
    private ViewFX view;

    boolean systemRuns = false;

    public Controller(Model model) throws InterruptedException {
        while (!ViewFX.isReady) {
            TimeUnit.MILLISECONDS.sleep(10);
        }
        this.model = model;
        this.view = ViewFX.currentView;

        view.buttonExit.setOnMousePressed(event -> System.exit(0));
        view.buttonRun.setOnMousePressed(event -> onButtonRun());
        view.buttonAddResource.setOnMousePressed(event ->
onButtonAddResource());
        view.buttonAddSensor.setOnMousePressed(event ->
onButtonAddSensor());
        view.buttonAddHandler.setOnMousePressed(event ->
onButtonAddHandler());
        view.timeline = new Timeline(new KeyFrame(
                Duration.millis(TIMER_PERIOD), new
EventHandler<ActionEvent>() {
            int time = 0;

            @Override
            public void handle(ActionEvent event) {
                view.updateEnvironment(model.onTimer(time), time);
                view.updateResourcesState();
                view.updateInfo(model.getEnergyConsumption() / ++time,
model.getRequirementsCompliance());
            }
        }
        ));
        view.timeline.setCycleCount(Animation.INDEFINITE);
    }

    private void onButtonAddSensor() {
        model.addSensor(
                FXSensor.newSensor.getName(),
                FXSensor.newSensor.getResource(),
                Double.valueOf(FXSensor.newSensor.getInfluence()),
                new Evaluable(
                        FXSensor.newSensor.getEvaluableFields().stream().map(
                                Double::valueOf
                        ).collect(Collectors.toList()).toArray(new Double[0])
                )
        );
        updateViewComponents();
    }

    private void updateViewComponents() {
        view.updateComponents(model.getResources(), model.getSensors(),
model.getHandlers(), systemRuns);
    }

    private void onButtonAddResource() {
        model.addResource(
                FXResource.newResource.getName(),
                FXResource.newResource.getEvaluableFields().stream().map(
                        Double::valueOf).collect(Collectors.toList()).toArray(new
Double[0])
        );
        updateViewComponents();
    }

    private void onButtonAddHandler() {
        model.addHandler(
                FXHandler.newHandler.getName(),
                FXHandler.newHandler.getResourceList(),
                FXHandler.newHandler.getSensorList(),
```

```
            new Evaluable(
                    FXHandler.newHandler.getRequirement().stream().map(
Double::valueOf).collect(Collectors.toList()).toArray(new Double[0])
            )
        );
        updateViewComponents();
    }

    private void onButtonRun() {
        if (!systemRuns) {
            systemRuns = true;
            updateViewComponents();
        }
        if (view.buttonRun.getText().equals("Pause")) {
            view.timeline.pause();
            view.buttonRun.setText("Continue");
        } else {
            view.timeline.play();
            view.buttonRun.setText("Pause");
        }
    }

    public void process() {
        initModel();
        updateViewComponents();
    }

    private void initModel() {
        List<EvaluableInitialParameters> eipList = new
ArrayList<EvaluableInitialParameters>() {{
            add(new EvaluableInitialParameters(0, "Radiator", 0.07d, 0d, 0d, 0d,
0d));
            add(new EvaluableInitialParameters(0, "Heater", 0.05d, 0d, 0d, 0d,
0d));
            add(new EvaluableInitialParameters(0, "Small heater", 0.04d, 0d, 0d,
0d, 0d));
            add(new EvaluableInitialParameters(0, "Thermopanel", 0.02d, 0d,
0d, 0d, 0d));
            add(new EvaluableInitialParameters(0, "Conditioner", -0.02d, 0d, 0d,
0d, 0d));
            add(new EvaluableInitialParameters(0, "Humidifier", 0d, 0.07d, 0d,
0d, 0d));
            add(new EvaluableInitialParameters(0, "Dehumidifier", 0d, -0.07d,
0d, 0d, 0d));
            add(new EvaluableInitialParameters(0, "Compressor", 0d, 0d, 15d,
0d, 0d));
            add(new EvaluableInitialParameters(0, "Decompressor", 0d, 0d,
-15d, 0d, 0d));
            add(new EvaluableInitialParameters(0, "Spotlight", 0.0001d, 0d, 0d,
0d, 30d));
            add(new EvaluableInitialParameters(0, "Luster", 0.0001d, 0d, 0d, 0d,
20d));
            add(new EvaluableInitialParameters(0, "Lamp 50W", 0.001d, 0d, 0d,
0d, 5d));
            add(new EvaluableInitialParameters(0, "Lamp 30W right", 0.001d,
0d, 0d, 0d, 3d));
            add(new EvaluableInitialParameters(0, "Lamp 30W left", 0.001d, 0d,
0d, 0d, 3d));

        }};

        model.setEnvironment(5, 60, 83, 0.3, 20);
        eipList.stream().filter(eip -> eip.checkType(0)).forEach(
                eip -> model.addResource(eip.name, eip.v1, eip.v2, eip.v3, eip.v4,
eip.v5)
        );

        model.addSensor("1st sensor", eipList.get(11).name, 0.5,
                new Evaluable(2d, 5d, 0.5, -0.01, 5d));

        model.addHandler("1st handler",
                eipList.stream().filter(eip -> eip.checkType(0)).map(eip ->
eip.name).collect(Collectors.toList()),
                new ArrayList<>(Arrays.asList(new String[]{"1st sensor"})),
                new Evaluable(20d, 40d, 98d, 0.3, 100d));
    }

    class EvaluableInitialParameters {
        public int type;
        public String name;
        public double v1;
        public double v2;
        public double v3;
        public double v4;
        public double v5;

        public EvaluableInitialParameters(int type, String name, double v1,
double v2, double v3, double v4, double v5) {
            this.type = type;
            this.name = name;
            this.v1 = v1;
            this.v2 = v2;
            this.v3 = v3;
            this.v4 = v4;
            this.v5 = v5;
        }

        public boolean checkType(int type) {
            return this.type == type;
        }
    }

}
```

```
package mvc;

import component.Handler;
import component.Sensor;
import component.evaluable.Environment;
import component.evaluable.Evaluable;
import component.evaluable.Resource;
import component.parameter.Parameter;
import component.parameter.ParameterValue;
import reasoner.Policy;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Model class of mvc
 *
 * @author pipich3
 * @version 1.0
 * @since 5/23/16
 */
public class Model {
    private static final double MODEL_ALTER_PERIOD = 10;

    public static final double
COEFFICIENT_ENVIRONMENT_STABILIZATION = 0.01;
    public static final double CONFLICTABLE_DEVIATION = 0.05;
    public static final String EXCEPTION_INCOMPARABLE_TYPES =
"Incomparable types: %s, %s";

    private Environment environment;
    private List<Handler> handlers = new ArrayList<>();
    private Policy handlerPolicy;

    public static List<Parameter> parameters = new ArrayList<Parameter>()
{{
        add(new Parameter("Temperature", -10, 60, false));
        add(new Parameter("Humidity", 0, 100, false));
        add(new Parameter("Pressure", 90, 110, true));
        add(new Parameter("CO2Level", 0, 10, false));
        add(new Parameter("Lighting", 0, 400, true));
    }};

    void setEnvironment(double temperature, double humidity, double
atmosphericPressure, double CO2Level,
                        double lighting) {
        environment =
                new Environment(temperature, humidity, atmosphericPressure,
CO2Level, lighting);
        handlerPolicy = Policy.getPolicy("Drop last");
    }
```

```
    void addResource(String name, Double... parameters) {
        environment.addResource(new Resource(name, parameters));
    }

    void addSensor(String name, String nearestResource, double
resourceInfluence, Evaluable maxError) {
        environment.addSensor(
                new Sensor(name, environment,
environment.getResource(nearestResource), resourceInfluence, maxError));
    }

    void addHandler(String name, List<String> resources, List<String>
sensors, Evaluable requirement) {
        handlers.add(new Handler(name,
                resources.stream().map(resource ->
environment.getResource(resource)).collect(Collectors.toList()),
                sensors.stream().map(sensor ->
environment.getSensor(sensor)).collect(Collectors.toList()),
                requirement));
    }

    List<ParameterValue> onTimer(int time) {
        environment.influentOuterEnvironment();
        if (time % MODEL_ALTER_PERIOD == 0) {
            environment.updateSensors();
            handlers.forEach(Handler::checkSensors);
        }
        environment.matchResources();
        return environment.getParameters();
    }

    List<Resource> getResources() {
        return environment.getResources();
    }

    List<Sensor> getSensors() {
        return environment.getSensors();
    }

    List<Handler> getHandlers() {
        return handlers;
    }

    double getRequirementsCompliance() {
        double totalCompliance = 0;
        for (Handler handler : handlers) {
            double compliance = 0;
            for (int j = 0; j < handler.getRequirement().getParameters().size(); j+
+) {
                compliance += Math.abs(
                        handler.getRequirement().getParameters().get(j).getValue() -
                                environment.getParameters().get(j).getValue()
                ) / (

environment.getParameters().get(j).getParameter().getMaxValue() -

environment.getParameters().get(j).getParameter().getMinValue()
                );
            }
            totalCompliance += compliance /
handler.getRequirement().getParameters().size();
        }
        return 1 - totalCompliance / handlers.size();
    }

    public double getEnergyConsumption() {
        return environment.getEnergyConsumption();
    }

}
```

```
package mvc;

import component.Handler;
import component.Sensor;
import component.evaluable.Resource;
import component.parameter.ParameterValue;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.geometry.Orientation;
import javafx.scene.Scene;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.*;
import javafx.stage.Stage;
import mvc.fx_component.FXEvaluable;
import mvc.fx_component.FXHandler;
import mvc.fx_component.FXResource;
import mvc.fx_component.FXSensor;

import java.util.List;
import java.util.stream.Collectors;

/**
 * ViewFX class of mvc
 *
 * @author pipich3
 * @version 1.0
 * @since 5/29/16
 */
public class ViewFX extends Application {

    private static final String ADD = "Add";
    public static ViewFX currentView = null;
    static boolean isReady = false;

    Button buttonExit;
    Button buttonRun;
    private SplitPane mainPane;

    public Accordion paneResource;
    public Accordion paneSensor;
    private Accordion paneHandler;

    public Button buttonAddResource;
    public Button buttonAddSensor;
    public Button buttonAddHandler;

    private List<XYChart.Series> environmentSeries;

    Timeline timeline;

    private TextField energyConsumption;
    private TextField requirementsSatisfied;

    @Override
    public void start(Stage primaryStage) throws Exception {
        initScene();
        isReady = true;

        primaryStage.setTitle("Conflict resolver");
        primaryStage.setScene(new Scene(mainPane, 900, 600));
        primaryStage.setResizable(false);
        primaryStage.show();
    }

    private void initScene() {
        buttonRun = new Button("Run system");
        buttonExit = new Button("Exit");
        buttonAddResource = new Button(ADD);
        buttonAddSensor = new Button(ADD);
        buttonAddHandler = new Button(ADD);

        currentView = this;

        paneResource = new Accordion(new FXResource());
        paneSensor = new Accordion();
        paneHandler = new Accordion();

        Accordion accordionComponents = new Accordion(
                new TitledPane("Resources", paneResource),
                new TitledPane("Sensors", paneSensor),
                new TitledPane("Handlers", paneHandler)
        );

        energyConsumption = new TextField();
```

```
        requirementsSatisfied = new TextField();

        TitledPane t1 = new TitledPane("Average energy consumption",
energyConsumption);
        TitledPane t2 = new TitledPane("Average requirements compliance",
requirementsSatisfied);
        t1.setCollapsible(false);
        t2.setCollapsible(false);

        ScrollPane scrollPane = new ScrollPane(accordionComponents);
        scrollPane.setFitToWidth(true);

        SplitPane paneControl = new SplitPane(scrollPane,
                t1, t2, buttonRun, buttonExit);

        paneControl.setOrientation(Orientation.VERTICAL);
        paneControl.setDividerPositions(0.6,0.8);

        LineChart<Number, Number> chart = new LineChart<>(
                new NumberAxis("Time", 0, 360, 10),
                new NumberAxis("Value", 0, 100, 5)
        );

        environmentSeries = FXEvaluable.textFieldNames.stream().map(
                s -> {
                    XYChart.Series series = new XYChart.Series<>();
                    series.setName(s);
                    return series;
                }
        ).collect(Collectors.toList());


        for (XYChart.Series series : environmentSeries) {
            chart.getData().add(series);
        }

        mainPane = new SplitPane(paneControl, chart);
        mainPane.setDividerPositions(0.3);



    }

    void updateInfo(double energy, double requirements) {
        energyConsumption.setText(String.format("%.2f", energy));
        requirementsSatisfied.setText(String.format("%.2f %%", requirements
* 100));
    }

    void updateResourcesState() {
        paneResource.getPanes().forEach(titledPane ->
((FXResource)titledPane).updateResourceState());
    }

    void updateComponents(List<Resource> resources, List<Sensor> sensors,
List<Handler> handlers, boolean systemRuns) {
        paneResource.getPanes().clear();
        paneSensor.getPanes().clear();
        paneHandler.getPanes().clear();

        if (!systemRuns) {
            paneResource.getPanes().add(new FXResource());
            paneSensor.getPanes().add(new FXSensor(

resources.stream().map(Resource::getName).collect(Collectors.toList())
            ));
            paneHandler.getPanes().add(new FXHandler(

resources.stream().map(Resource::getName).collect(Collectors.toList()),

sensors.stream().map(Sensor::getName).collect(Collectors.toList())
            ));
        }

        resources.forEach(c -> paneResource.getPanes().add(new
FXResource(c)));
        sensors.forEach(c -> paneSensor.getPanes().add(new FXSensor(c)));
        handlers.forEach(c -> paneHandler.getPanes().add(new
FXHandler(c)));
    }

    void updateEnvironment(List<ParameterValue> environmentParameters,
int tick) {
        int seriesCounter = 0;
        for (XYChart.Series series : environmentSeries) {
            series.getData().add(new XYChart.Data<>(tick,
environmentParameters.get(seriesCounter).getValue()));
            seriesCounter++;
        }
    }
}
```

```
package mvc.fx_component;

import javafx.geometry.Insets;
import javafx.scene.control.TitledPane;

import java.util.List;

/**
 * FXResource class of mvc.fx_component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/29/16
 */
abstract class FXComponent extends TitledPane {

    FXComponent(String title) {
        setText(title);
        setPadding(new Insets(2, 2, 2, 20));
    }

    abstract List<String> getTextFields();

}
```

```
package mvc.fx_component;

import component.evaluable.Evaluable;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.TextInputControl;
import javafx.scene.control.TitledPane;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import mvc.Model;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

/**
 * FXEvaluable class of mvc.fx_component
 *
 * @author pipich3
 * @version 1.0
 * @since 6/1/16
 */
public class FXEvaluable extends TitledPane {
    private static final Integer[] COL_WIDTH = new Integer[]{60, 40};
    private List<TextField> textFields = new ArrayList<>();

    public static final List<String> textFieldNames = new
ArrayList<String>() {{
        add("Temperature");
        add("Humidity");
        add("Pressure");
        add("CO2 level");
        add("Lighting");
    }};

    FXEvaluable(String title) {
        this(title, false);
    }
```

```
    public FXEvaluable(String title, boolean collapsible) {

        textFields = Model.parameters.stream().map(s -> {
            TextField textField = new TextField();
            textField.setPromptText(s.getName());
            return textField;
        }).collect(Collectors.toList());

        setText(title);
        setCollapsible(collapsible);
        setContent(new VBox(textFields.toArray(new TextField[0])));
    }

    public FXEvaluable(String title, Evaluable evaluable) {
        this(title, evaluable, false);
    }

    FXEvaluable(String title, Evaluable evaluable, boolean collapsible) {
        GridPane gridPane = new GridPane();
        gridPane.getColumnConstraints().setAll(new
ArrayList<>(Arrays.asList(COL_WIDTH)).stream().map(
                d -> {
                    ColumnConstraints columnConstraints = new
ColumnConstraints();
                    columnConstraints.setPercentWidth(d);
                    return columnConstraints;
                }
        ).collect(Collectors.toList()));

        setText(title);
        setContent(gridPane);
        setCollapsible(collapsible);

        textFields = evaluable.getParameters().stream().map(
                d -> {
                    TextField textField = new
TextField(Double.toString(d.getValue()));
                    textField.setEditable(false);
                    return textField;
                }).collect(Collectors.toList());

        gridPane.addColumn(0, textFieldNames.stream().map(
                Label::new).collect(Collectors.toList()).toArray(new Node[]{}));
        gridPane.addColumn(1, textFields.toArray(new Node[]{}));

        this.getChildren().add(gridPane);
        gridPane.setAlignment(Pos.BASELINE_RIGHT);
    }

    List<String> getTextFields() {
        return
textFields.stream().map(TextInputControl::getText).collect(Collectors.toList(
));
    }
}
```

```
package mvc.fx_component;

import component.Handler;
import javafx.collections.FXCollections;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import mvc.ViewFX;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * FXHandler class of mvc.fx_component
 *
 * @author pipich3
 * @version 1.0
 * @since 6/3/16
 */
public class FXHandler extends FXComponent {
    public static FXHandler newHandler = null;

    private TextField nameField;

    private List<String> resourceList = new ArrayList<>();
    private List<String> sensorList = new ArrayList<>();

    private FXEvaluable requirement;

    public FXHandler(List<String> resources, List<String> sensors) {
        super("New handler");
        newHandler = this;

        nameField = new TextField();
        nameField.setPromptText("Name");

        List<List<Node>> nodes = new ArrayList<>();
        nodes.addAll(
                new FXComponentList(
                        "Resources",
                        resources,
                        resourceList).getNodes()
        );
        nodes.addAll(
                new FXComponentList(
                        "Sensors",
                        sensors,
                        sensorList).getNodes()
        );

        GridPane gridPane = new GridPane();
        gridPane.addRow(0, nameField,
ViewFX.currentView.buttonAddHandler);

        int i = 1;
        for (List<Node> l : nodes) {
            gridPane.addRow(i++, l.toArray(new Node[0]));
        }

        requirement = new FXEvaluable("New requirement", true);

        setContent(new VBox(gridPane, requirement));
    }

    public FXHandler(Handler handler) {
        super(handler.getName());

        this.setContent(new VBox(
                new TitledPane("Resources", new VBox(
                        handler.getResources().stream().map(resource ->
                                new
Label(resource.getName())).collect(Collectors.toList()).toArray(new
Label[0])
                )),
                new TitledPane("Sensors", new VBox(
                        handler.getSensors().stream().map(sensor ->
                                new
Label(sensor.getName())).collect(Collectors.toList()).toArray(new Label[0])
                )),
                new FXEvaluable("Requirement", handler.getRequirement(), true)
        ));
    }

    private class FXComponentList extends TitledPane {
        List<String> components;
        List<String> componentsSelected;
        ComboBox<String> componentsBox;
        ComboBox<String> componentsSelectedBox;
        Button buttonAdd;
        Button buttonRemove;

        public FXComponentList(String title, List<String> components,
List<String> componentsSelected) {
            this.components = components;
            this.componentsSelected = componentsSelected;

            buttonAdd = new Button("Add");
```

```
        buttonRemove = new Button("Remove");

        componentsBox = new ComboBox<>();
        componentsSelectedBox = new ComboBox<>();
        componentsBox.setPromptText(title);
        componentsSelectedBox.setPromptText("Selected " + title);
        updateComboBoxes();

        setText(title);
        setContent(new VBox(
              new HBox(componentsBox, buttonAdd),
              new HBox(componentsSelectedBox, buttonRemove)
        ));

        buttonAdd.setOnMousePressed(event -> {
           if (componentsBox.getValue() == null ||

componentsBox.getValue().equals(componentsBox.getPromptText())) {
              return;
           }
           components.remove(componentsBox.getValue());
           componentsSelected.add(componentsBox.getValue());
           updateComboBoxes();
        });

        buttonRemove.setOnMousePressed(event -> {
           if (componentsSelectedBox.getValue() == null ||

componentsSelectedBox.getValue().equals(componentsSelectedBox.getPro
mptText())) {
              return;
           }
           componentsSelected.remove(componentsSelectedBox.getValue());
           components.add(componentsSelectedBox.getValue());
           updateComboBoxes();
        });
     }

     public List<List<Node>> getNodes() {
        return new ArrayList<List<Node>>(){{
           add(new ArrayList<Node>(){{
              add(componentsBox);
              add(buttonAdd);
           }});
           add(new ArrayList<Node>(){{
              add(componentsSelectedBox);
              add(buttonRemove);
           }});
        }};
     }


     private void updateComboBoxes() {

componentsBox.setItems(FXCollections.observableList(this.components));

componentsSelectedBox.setItems(FXCollections.observableList(component
sSelected));
     }

     public List<String> getComponentsSelected() {
        return componentsSelected;
     }
  }

  @Override
  List<String> getTextFields() {
     return null;
  }

  public String getName() {
     return nameField.getText();
  }

  public List<String> getResourceList() {
     return resourceList;
  }

  public List<String> getSensorList() {
     return sensorList;
  }

  public List<String> getRequirement() {
     return requirement.getTextFields();
  }
}
```

```
package mvc.fx_component;

import component.evaluable.Resource;
import javafx.geometry.Pos;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import mvc.ViewFX;

import java.util.List;

/**
 * FXResource class of mvc.fx_component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/29/16
 */
public class FXResource extends FXComponent{
  private static final String PARAMETERS = "Parameters";
  public static FXResource newResource;

  private FXEvaluable evaluable;
  private TextField nameField;

  private Resource resource;

  public FXResource() {
     super("New resource");

     evaluable = new FXEvaluable(PARAMETERS);
     nameField = new TextField();
     nameField.setPromptText("Name");

     GridPane gridPane = new GridPane();
     gridPane.addRow(0, nameField,
ViewFX.currentView.buttonAddResource);
     gridPane.setAlignment(Pos.BASELINE_RIGHT);

     this.setContent(new VBox(gridPane, evaluable));
     newResource = this;
  }

  public FXResource(Resource resource) {
     super(resource.getName());
     this.resource = resource;
     setStyle(resource.isActive() ?
           "-fx-color: rgb(200, 255, 200, 0.8)" :
           "-fx-color: rgb(255, 200, 200, 0.8)"
     );
     evaluable = new FXEvaluable(PARAMETERS, resource);
     this.setContent(new VBox(evaluable));
  }

  public String getName() {
     return nameField.getText();
  }

  @Override
  public List<String> getTextFields() {
     List<String> textFields = evaluable.getTextFields();
     textFields.add(0, nameField.getText());
     return textFields;
  }

  public List<String> getEvaluableFields() {
     return evaluable.getTextFields();
  }

  public void updateResourceState() {
     setStyle(resource != null && resource.isActive() ?
```

```
            "-fx-color: rgb(200, 255, 200, 0.8)" :
            "-fx-color: rgb(255, 200, 200, 0.8)"
      );
   }

}
```

```
package mvc.fx_component;

import component.Sensor;
import javafx.collections.FXCollections;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import mvc.ViewFX;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

/**
 * FXSensor class of mvc.fx_component
 *
 * @author pipich3
 * @version 1.0
 * @since 6/1/16
 */
public class FXSensor extends FXComponent {
   public static final String DEVIATIONS_MAXIMUM = "Deviations
maximum";
   public static FXSensor newSensor;

   TextField nameField = new TextField();
   ComboBox<String> resourceBox;
   TextField resourceInfluence;
   TextField resourceField;
   FXEvaluable evaluable;

   public FXSensor(List<String> resoureList) {
      super("New sensor");
      evaluable = new FXEvaluable(DEVIATIONS_MAXIMUM);

//       List<String> list =
//
ViewFX.currentView.paneResource.getPanes().stream().map(Labeled::getTe
xt).collect(Collectors.toList());
////       list =
//       list.remove(0);
      resourceBox = new
ComboBox<>(FXCollections.observableList(resoureList));
      resourceBox.setPromptText("Nearest resource");

      resourceInfluence = new TextField();
      resourceInfluence.setPromptText("Resource influence");

      nameField.setPromptText("Name");

      GridPane gridPane = new GridPane();
      gridPane.addRow(0, nameField,
ViewFX.currentView.buttonAddSensor);

      this.setContent(new VBox(gridPane, resourceBox, resourceInfluence,
evaluable));

      newSensor = this;
   }

   public FXSensor(Sensor sensor) {
      super(sensor.getName());
      evaluable = new FXEvaluable(DEVIATIONS_MAXIMUM,
sensor.getError());
      resourceField = new TextField(sensor.getNearestResourceName());
      resourceField.setEditable(false);
      resourceInfluence = new
TextField(Double.toString(sensor.getResourceInfluencePower()));
      resourceInfluence.setEditable(false);

      GridPane gridPane = new GridPane();
      gridPane.getColumnConstraints().addAll(
            new ArrayList<>(Arrays.asList(new Integer[]{60,
40})).stream().map(
                  d -> {
                     ColumnConstraints columnConstraints = new
ColumnConstraints();
                     columnConstraints.setPercentWidth(d);
                     return columnConstraints;
                  }
            ).collect(Collectors.toList())
      );
      gridPane.addRow(0, new Label("Nearest resource"), resourceField);
      gridPane.addRow(1, new Label("Resource influence"),
resourceInfluence);

      this.setContent(new VBox(gridPane, evaluable));
   }

   @Override
   public List<String> getTextFields() {
      List<String> textFields = new ArrayList<>();
      textFields.add(nameField.getText());
      textFields.add(resourceBox.getValue());
      textFields.add(resourceInfluence.getText());
      textFields.addAll(evaluable.getTextFields());
      return textFields;
   }

   public String getName() {
      return nameField.getText();
   }

   public String getResource() {
      return resourceBox.getValue();
   }

   public String getInfluence() {
      return resourceInfluence.getText();
   }

   public List<String> getEvaluableFields() {
      return evaluable.getTextFields();
   }
}
```

```
package component.evaluable;

import component.parameter.ParameterValue;
import mvc.Model;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Resource class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/17/16
 */
public class Evaluable implements Cloneable{
   List<ParameterValue> parameters = new ArrayList<>();

   public Evaluable(Double ... parameters) {
      final int[] parameterCounter = {0};
      this.parameters = Model.parameters.stream().map(
            parameter -> new ParameterValue(parameter,
parameters[parameterCounter[0]++])
      ).collect(Collectors.toList());
   }

   private static Double[] doubleToDouble(double[] parameters) {
      Double[] D = new Double[parameters.length];
      for (int i = 0; i < parameters.length; i++) {
         D[i] = parameters[i];
```

```
        }
        return D;
    }

    public Evaluable clone() {
        return new Evaluable(parameters.stream().map(
                ParameterValue::getValue
        ).collect(Collectors.toList()).toArray(new Double[0]));
    }

    public List<ParameterValue> getParameters() {
        return new ArrayList<>(parameters);
    }

    public boolean isConflict(Evaluable evaluable) {
        return isConflict(evaluable, false);
    }

    public boolean isConflict(Evaluable evaluable, boolean isResource) {
        int parameterCounter = 0;
        for (ParameterValue parameter : parameters.stream().filter(
                parameterValue -> ((!isResource) ||
parameterValue.getParameter().isInstant())
        ).collect(Collectors.toList())) {
            if (Math.abs(parameter.getValue() -
evaluable.parameters.get(parameterCounter++).getValue()) >
                    (parameter.getParameter().getMaxValue() -
parameter.getParameter().getMinValue())
                            * Model.CONFLICTABLE_DEVIATION) {
                return true;
                }
        }
        return false;
    }

}
```

```
package component.evaluable;

import component.ComponentException;
import component.Sensor;
import component.parameter.ParameterValue;
import mvc.Model;

import java.util.ArrayList;
import java.util.List;

/**
 * Environment class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/17/16
 */
public class Environment extends Evaluable {

    private static final String SOURCE_ENVIRONMENT_CONDITIONS =
"source environment conditions";
    private static final String EXCEPTION_RESOURCE_USE =
"Environment can not use resource ";
    private static final String EXCEPTION_UNKNOWN_RESOURCE =
"Unknown resource ";
    private static final String EXCEPTION_UNKNOWN_SENSOR =
"Unknown sensor ";

    private Resource sourceConditions;
    private List<Resource> resources = new ArrayList<>();
    private List<Sensor> sensors = new ArrayList<>();

    private int energyConsumption = 0;

    public Environment(Double ... parameters) {
        super(parameters);
        sourceConditions =
                new Resource(SOURCE_ENVIRONMENT_CONDITIONS,
parameters);
        sourceConditions.setActivity(true);
    }

    public void addResource(Resource resource) {
        resources.add(resource);
    }

    public Resource getResource(String name) {
        for (Resource resource : resources) {
            if (name.equals(resource.getName())) {
                return resource;
            }
        }
        throw new
ComponentException(EXCEPTION_UNKNOWN_RESOURCE + name);
    }

    public void addSensor(Sensor sensor) {
        sensors.add(sensor);
    }

    public Sensor getSensor(String name) {
        for (Sensor sensor : sensors) {
            if (name.equals(sensor.getName())) {
                return sensor;
            }
        }
        throw new
ComponentException(EXCEPTION_UNKNOWN_SENSOR + name);
    }

    public void matchResources() {
        resources.stream().filter(Resource::isActive).forEach(resource ->
resource.influence(this, 1));

    }

    public void influentOuterEnvironment() {
        int parameterCounter = 0;
        for (ParameterValue parameter : parameters) {
            double sourceConditionsValue =
sourceConditions.getParameters().get(parameterCounter++).getValue();
            if (parameter.getParameter().isInstant()) {
                parameter.setValue(sourceConditionsValue);
            } else {
                parameter.increaseValue(
                        (sourceConditionsValue - parameter.getValue())
                                *
Model.COEFFICIENT_ENVIRONMENT_STABILIZATION
                );
            }
        }
    }

    public void updateSensors() {
        sensors.forEach(Sensor::update);
    }

    public List<Resource> getResources() {
        return resources;
    }

    public List<Sensor> getSensors() {
        return sensors;
    }

    public void useEnergy() {
        energyConsumption++;
    }

    public int getEnergyConsumption() {
        return energyConsumption;
    }
}
```

```
package component.evaluable;

import component.Component;
import component.parameter.ParameterValue;
import mvc.Model;
import reasoner.Policy;
```

```
import java.util.List;
import java.util.stream.Collectors;

/**
 * ResourceImpl class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/23/16
 */
public class Resource extends Evaluable implements Component {
    private String name;
    private boolean activity = false;
    private Policy handlerPolicy;

    public Resource(String name, Double... parameters) {
        super(parameters);
        this.name = name;
        handlerPolicy = Policy.getPolicy("Drop last");
    }

    public Resource(Evaluable evaluable) {
        super(evaluable.getParameters().stream().map(

ParameterValue::getValue).collect(Collectors.toList()).toArray(new
Double[0]));
    }

    public String getName() {
        return name;
    }

    public boolean isActive() {
        return activity;
    }

    public void setActivity(boolean activity) {
        this.activity = activity;
    }

    public void influence(Evaluable characteristics, double influencePower) {
        final int[] parameterCounter = new int[] {0};
        characteristics.getParameters().forEach(parameterValue ->

parameterValue.increaseValue(parameters.get(parameterCounter[0]+
+).getValue() * influencePower));
        if (characteristics instanceof Environment) {
            ((Environment) characteristics).useEnergy();
        }
        }

    @Override
    public boolean isConflict(Component component) {
        if (component instanceof Resource) {
            return super.isConflict((Resource)component);
        }
        throw new IllegalArgumentException(String.format(
                Model.EXCEPTION_INCOMPARABLE_TYPES, "Resource",
component.getClass().getSimpleName()));
    }

    public static Evaluable getResourcesSummary(List<Resource> resources,
int resourceCombination) {
        Evaluable resourceSummary = new Evaluable(new
Double[Model.parameters.size()]);
        final int[] finalShifter = {resourceCombination};

        resources.forEach(resource -> {
            if ((finalShifter[0] & 0x1) == 1) {
                resource.influence(resourceSummary, 1);
            }
            finalShifter[0] >>=1;
        });

        return resourceSummary;
    }
}
```

```
package component;

/**
 * Component class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/29/16
 */
public interface Component {
    String getName();
    boolean isConflict(Component component);

}
```

```
package component;

/**
 * ComponentException class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/24/16
 */
public class ComponentException extends IllegalArgumentException {
    public ComponentException(String message) {
        super(message);
    }
}
```

```
package component;

import component.evaluable.Evaluable;
import component.evaluable.Resource;
import javafx.util.Pair;
import mvc.Model;
import reasoner.Policy;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Handler class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/17/16
 */
public class Handler implements Component {

    private String name;
    private List<Resource> resources = new ArrayList<>();
    private List<Sensor> sensors = new ArrayList<>();
    private Evaluable requirement;

    private Policy sensorPolicy;
    private Policy resourcePolicy;

    public Handler(String name, List<Resource> resources, List<Sensor>
sensors, Evaluable requirement) {
        this.name = name;
        this.requirement = requirement;
        this.resources.addAll(resources);
        this.sensors.addAll(sensors);
        this.sensorPolicy = Policy.getPolicy("Drop last");
        this.resourcePolicy = Policy.getPolicy("Drop last");
    }

    public List<Resource> getResources() {
        return resources;
    }

    public List<Sensor> getSensors() {
        return sensors;
    }

    public Evaluable getRequirement() {
        return requirement;
    }
```

```
    public void checkSensors() {
        List<Sensor> inconflictedSensors =
sensorPolicy.resolve(sensors).stream().map(
                component -> (Sensor)component).collect(Collectors.toList());

        Evaluable sensorsSummary =
Sensor.getSensorsSummary(inconflictedSensors);
        final int[] parametersCounter = new int[]{0};
        sensorsSummary.getParameters().forEach(parameterValue ->

parameterValue.setValue(requirement.getParameters().get(parametersCounte
r[0]++).getValue()
                - parameterValue.getValue()));

        Pair<Integer, Double> bestResourceCombination = new Pair<>(0,
getDeviation(sensorsSummary, 0));
        for (int i = 1; i < (1 << resources.size()); i++) {
            double deviation = getDeviation(sensorsSummary, i);
            if (deviation < bestResourceCombination.getValue()) {
                bestResourceCombination = new Pair<>(i, deviation);
            }
        }

        final int[] resourceCounter = {0};
        final int combination = bestResourceCombination.getKey();
        resources.forEach(resource ->
                resource.setActivity((0x1 & (combination >>
(resourceCounter[0]++))) == 1));

    }

    private Double getDeviation(Evaluable sensorsSummary, int
resourceCombination) {
        Evaluable resourceSummary =
Resource.getResourcesSummary(resources, resourceCombination);
        final double[] deviation = {0};
        final int[] parameterCounter = {0};

        sensorsSummary.getParameters().forEach(parameterValue ->
deviation[0] +=
                Math.abs(parameterValue.getValue() -
                        resourceSummary.getParameters().get(parameterCounter[0]+
+).getValue()) /
                        (parameterValue.getParameter().getMaxValue() -
parameterValue.getParameter().getMinValue()) /
                        (parameterValue.getParameter().isInstant() ? 1000 : 1)
        );
        return deviation[0];
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public boolean isConflict(Component component) {
        if (component instanceof Handler) {
            return requirement.isConflict(((Handler) component).requirement);
        }
        throw new IllegalArgumentException(String.format(
                Model.EXCEPTION_INCOMPARABLE_TYPES, "Handler",
component.getClass().getSimpleName()));
    }
}
```

```
package component;

import component.evaluable.Environment;
import component.evaluable.Evaluable;
import component.evaluable.Resource;
import mvc.Model;

import java.util.List;
import java.util.Random;

/**
 * Sensor class of component
 *
 * @author pipich3
 * @version 1.0
 * @since 5/17/16
 */
public class Sensor implements Component {
    private Environment environment;
    private Evaluable characteristics;

    private Resource nearestResource;
    private Resource error;

    private double resourceInfluencePower;
    private String name;

    public Sensor(String name, Environment environment, Resource
nearestResource, double resourceInfluencePower,
                  Evaluable maxError) {
        this.environment = environment;
        this.nearestResource = nearestResource;
        this.error = new Resource(maxError);
        this.resourceInfluencePower = resourceInfluencePower;
        this.name = name;
    }

    public void update() {
        characteristics = environment.clone();
        nearestResource.influence(characteristics, resourceInfluencePower);
        error.influence(characteristics, getErrorInfluence());
    }

    private double getErrorInfluence() {
        return new Random().nextDouble() - 0.5;
    }

    public String getName() {
        return name;
    }

    public String getNearestResourceName() {
        return nearestResource.getName();
    }

    public double getResourceInfluencePower() {
        return resourceInfluencePower;
    }

    public Evaluable getCharacteristics() {
        return characteristics;
    }

    public Evaluable getError() {
        return error.clone();
    }

    @Override
    public boolean isConflict(Component component) {
        if (component instanceof Sensor){
            return characteristics.isConflict(((Sensor)
component).characteristics);
        }
        throw new IllegalArgumentException(String.format(
                Model.EXCEPTION_INCOMPARABLE_TYPES, "Sensor",
component.getClass().getSimpleName()));
    }

    public static Evaluable getSensorsSummary(List<Sensor>
inconflictedSensors) {
        Evaluable sensorsSummary = new Evaluable(new
Double[Model.parameters.size()]);
        inconflictedSensors.forEach(sensor -> {
            final int[] parametersCounter = {0};
            sensor.getCharacteristics().getParameters().forEach(parameterValue
->
                    sensorsSummary.getParameters().get(parametersCounter[0]+
+).increaseValue(parameterValue.getValue()));
        });
        return sensorsSummary;
    }
}
```