УДК 004.67

ІНТЕЛЕКТУАЛІЗОВАНА БІБЛІОТЕКА РЕФАКТОРИНГУ ДЛЯ IDE VISUAL STUDIO

Степанюк А.С.[1)], Манжула В.І.[2)]
Західноукраїнський національний університет
[1)] магістрант; [2)] к.т.н., доцент

І. Актуальність проблеми

З кожним роком важливість програмного забезпечення у всіх галузях зростає, а разом з нею і складність розробки. Сучасні умови вимагають ефективних стратегій розробки та гнучких методик управління процесами. Однак, з часом накопичуються технічні борги, що можуть суттєво вплинути на подальший розвиток програмного продукту. Вирішенням цього виклику є покращення якості коду шляхом рефакторингу.Рефакторинг, за визначенням Мартіна Фаулера, – це процес зміни внутрішньої структури програми, що не впливає на її зовнішню поведінку, має на меті полегшити розуміння коду та покращити його еволюцію. Основою рефакторингу є послідовність невеликих, еквівалентних перетворень, які не змінюють поведінку програми[1-6].

Аналізуючи наявні інструменти, такі як ReSharper та CodeRush для IDE Microsoft Visual Studio, можна виокремити кілька проблем [7,8]:

- обмеження комерційних продуктів –деякі інструменти є комерційними, обмежуючи доступність для широкого кола користувачів;
- вплив на продуктивність –деякі розширення можуть негативно впливати на продуктивність та швидкість роботи IDE;
- відсутність спеціалізації –деякі інструменти не спеціалізуються чітко на рефакторингу, пропонуючи розширення інших компонентів IDE.

Враховуючи вищезазначені недоліки, актуальною є розробка нових компонент для рефакторингу в IDE Visual Studio. З використанням API платформи компіляторів .Net, новий компонент повинен мати мінімальний вплив на продуктивність IDE,бути відкритим продуктом та призначатися для широкого кола розробників [9-12].

ІІ. Мета дослідження

Метою роботи є створення інструменту для автоматизованого рефакторингу коду в IDE Visual Studio.

Завдання включають:

- аналіз існуючих засобів та генераторів коду;
- дослідження метрик коду та їх актуальності;
- розробку ефективного інструменту для автоматизованого рефакторингу.

ІІІ. Особливості реалізаціїбібліотеки рефакторингу

При розробці програмного продукту великою увагою був обраний вибір інструментів, які допомагають спростити роботу програміста та забезпечують всі необхідні засоби для успішної реалізації поставлених завдань. Основними засобами розробки були середовище Visual Studio, мова програмування C#, пакет Visual Studio SDK, платформа компіляторів .NET (“Roslyn”) SDK та експериментальний екземпляр Visual Studio.

Важливим етапом був вибір операційної системи для розробки програмного продукту. З урахуванням популярності, багатофункціональності, сумісності та надійності операційної системи Microsoft Windows, а також зручності роботи з нею, було вирішено використовувати середовище Microsoft Windows 10.Важливо відзначити, що можливість використання розробленого програмного продукту не залежить від операційної системи користувача. Продукт може успішно функціонувати на комп'ютерах з операційною системою Windows, Mac чи UNIX, на яких встановлено відповідну версію Visual Studio.

Програмне розширення надає можливість реалізації різновидів рефакторингу, які можна розділити на три основні категорії:

- генерація нового коду –включає в себе функціонал, спрямований на автоматичне створення нових фрагментів програмного коду;
- усунення помилок, що можуть виникнути на етапі виконання –забезпечує інструменти для виявлення та виправлення помилок ще на етапі розробки, що сприяє покращенню якості програми;

- трансформація коду до сучасної форми запису –включає функції, спрямовані на оптимізацію та адаптацію існуючого коду до сучасних стандартів та практик програмування.

До першої категорії належить можливість автоматичного створення події на зміну властивості, яка часто виникає, коли потрібно реагувати на зміни властивостей об'єкта. Для автоматизації цього процесу вирішено використовувати однаковий синтаксис для всіх подій цього типу. Користувачу достатньо поставити курсор на назві властивості, і система автоматично згенерує відповідну подію типу EventHandler.

Друга категорія рефакторингу дозволяє оптимізувати код, який часто може викликати помилки на етапі виконання. У розробці іноді забувають, що типи посилань можуть мати значення за замовчуванням null. Це може спричинити неочікувані виключення NullReferenceException. Щоб уникнути таких проблем, запропоновано рефакторинг, який додає перевірку на null до вхідних параметрів методу та змінних всередині методу, які належать до типу посилань або до Nullable типу.

Рефакторинг "To lambda expression" надає зручний спосіб конвертації анонімного методу до лямбда-виразу. Хоча для створення анонімних функцій можна використовувати і анонімні методи, лямбда-вирази забезпечують більш оптимізований синтаксис, що відповідає трансформації коду до сучасної форми запису.

Програмне розширення забезпечує рефакторинг, який сприяє підвищенню продуктивності розробки за допомогою генерації нового коду, виправлення помилок, які можуть виникнути на етапі виконання, та трансформації коду до сучасної форми запису. Методи рефакторингу інтегруються в інтерфейс IDE Visual Studio, надаючи користувачу змогу переглядати доступні опції рефакторингу для конкретного блоку коду.

Висновки

У даній роботі розглянуто питання необхідності створення інструменту для автоматизованого рефакторингу коду.У процесі аналізу наявного програмного забезпечення для автоматичного рефакторингу виявлено недоліки існуючих рішень. Дослідження показало, що компоненти рефакторингу, інтегровані в IDE Visual Studio, не лише займаються рефакторингом, але також вирішують інші завдання, такі як підтримка юніт-тестування, створення синтаксичного аналізатора та додавання нових функцій до редактора. Це призводить до негативного впливу на продуктивність IDE.

На основі проведеного аналізу було вирішено створити новий компонент для розширення функціоналу рефакторингу в IDE Visual Studio. Використання API, яке надає платформа компіляторів .Net, дозволяє мінімізувати вплив на продуктивність IDE. Новий компонент рефакторингу в IDE Visual Studio є відкритим продуктом для загального використання.Розроблений програмний продукт автоматизує процес розпізнавання фрагментів коду, які можна піддати рефакторингу, і сам процес рефакторингу через додаткові опції у меню "Quick Actions".У роботі проведено огляд методів та засобів розробки програмної системи. Обгрунтовано вибір створення програмної системи на основі SDK та .Net Compiler Platform [9-10].

Список використаних джерел

1. Fowler, M. (1999). "Refactoring: Improving the Design of Existing Code." Addison-Wesley.
2. S.Krepych, I.Spivak, “Improvement of SVD algorithm to increase the efficiency of recommendation systems”. Advanced Information Systems. 2021. – Num.5. Vol.4. pp.55-59
3. Martin, R. C. (2008). "Clean Code: A Handbook of Agile Software Craftsmanship." Prentice Hall.
4. Крепич С.Я. Програмний комплекс оцінювання функціональної придатності пристроїв при заданих допустимих значеннях вихідних характеристик та допусків на параметри їх елементів. Сучасні комп‘ютерні інформаційні технології: Матеріали V Всеукраїнської школи-семінару молодих вчених і студентів АСІТ‘2015. – Тернопіль: ТНЕУ, 2015. – С. 23-25.
5. The official website dedicated to refactoring.[Електронний ресурс]. –Режим доступу до ресурсу:https://Refactoring.com
6. Feathers, M. C. (2004). "Working Effectively with Legacy Code." Prentice Hall.
7. refactoring.guru(Include the specific URL or publication details if available on the repository).[Електронний ресурс]. –Режим доступу до ресурсу:https://refactoring.guru/
8. IDEVisual Studio.[Електронний ресурс]. –Режим доступу до ресурсу:https://visualstudio.microsoft.com/
9. Code Rush.[Електронний ресурс]. –Режим доступу до ресурсу:https://marketplace.visualstudio.com/items?itemName=DevExpress.CodeRushforRoslyn
10. The .NET Compiler Platform SDK.[Електронний ресурс]. –Режим доступу до ресурсу:https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/
11. Томас Кормен, Чарльз Лейзерсон, Рональд Рівест, Кліффорд Стайн. Вступ до алгоритмів. 2019. р 190-194
12. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms, Fourth Edition. 2022. p. 76-90.