МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**Звіт**

**з наукової практики**

магістра І курсу групи ІПЗм-21-2

спеціальності 121 «Інженерія програмного забезпечення»

\_Ліщинського Олександра Анатолійовича\_

(прізвище, ім’я та по-батькові)

Керівник професор кафедри ІПЗ ,

, д.п.н., проф. Вакалюк Т.А. ,

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 2022 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Морозов А.В. \_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанчикова Н.М. .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Антонюк Д.С. \_\_\_

(підпис) (прізвище та ініціали)

Житомир – 2022

ІНДИВІДУАЛЬНИЙ ГРАФІК ПРОХОДЖЕННЯ

НАУКОВОЇ ПРАКТИКИ

студентом 1 року навчання групи \_\_\_\_\_\_\_

денної/заочної форми навчання

спеціальності 121 «Інженерія програмного забезпечення»

(ОПП «Інженерія програмного забезпечення»)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Термін проходження практики: з 24.01.2022 по 04.02.2022 року.

| Завдання за планом | Термін виконання | Фактичне виконання | Підписи  наукового керівника та керівника від кафедри |
| --- | --- | --- | --- |
| 1. Інструктаж з техніки безпеки. Розробка індивідуального графіку проходження практики. Узгодження його з науковим керівником магістерської роботи та керівником практики від університету. |  |  |  |
| 1. Формулювання теми магістерських досліджень, визначення предмету та об’єкту дослідження. |  |  |  |
| 1. Ознайомлення з науковими напрямами роботи кафедри, на якій проходять практику. |  |  |  |
| 1. Ознайомлення з іноземними та вітчизняними науково-інформаційними джерелами за спеціалізацією, обрання наукової проблематики та формування бібліографії. |  |  |  |
| Збір та обробка відповідними методами фактичного, фактологічного та статистичного матеріалу щодо стану об’єкту дослідження. |  |  |  |
| Ознайомлення з нормативно-правовою документацією за обраною проблематикою та формування напрямів удосконалення щодо стану об’єкту дослідження. |  |  |  |
| Виконання індивідуального завдання з обраної проблеми досліджень. |  |  |  |
| 1. Оформлення звіту з практики. |  |  |  |

Тема магістерської кваліфікаційної роботи:

\_\_\_Алгоритмічно-програмний комбінований метод генерації коду шаблонів веб-застосунків\_

Узгоджено: «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

**Науковий керівник магістерської роботи:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(науковий ступінь, вчене звання керівника, прізвище, ім'я, по-батькові) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Керівник практики від університету:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(науковий ступінь, вчене звання керівника,прізвище, ім'я, по-батькові) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЕФЕРАТ**

**Актуальність теми.** На сьогоднішній день постійне зростання обсягів розроблюваного програмного забезпечення та попиту на створення нових програм є причиною того, що підвищується рівень вимог до швидкості написання коду та якості отримуваних застосунків.

Для того, щоб код програми було легко читати та модифікувати, від розробників вимагають дотримання у роботі принципу модульності, винесення основної функціональності застосунку в окремі утиліти, використання об’єктно-орієнтованого програмування і т.д. Як відомо, для підвищення швидкості написання коду у переважній більшості випадків програмісти вдаються до пошуку готових рішень та подальшої модифікації знайденого фрагменту програмного коду з точки зору наявної задачі. Але такий підхід з копіюванням чужого коду містить потенційний ризик того, що в додатку залишаться зайві дії, що не мають відношення до розв’язуваної ним задачі.

Також відомо, що задля пришвидшення створення програмного продукту розробники або тестувальники досить часто використовують генерування структури проєкту із заготовками коду деяких класів, методів та конфігураційних файлів відповідно до обраного типу проєкту. Звісно, така допомога у написанні коду зменшує час на створення програмного забезпечення та дозволяє розробникам зосереджуватися на більш творчих задачах. Але крім того хотілося б мати можливість автоматизації створення шаблонів окремих програмних компонентів (наприклад, кастомізованих React-компонентів для веб-застосунків), а також мати можливість більш тонкого налаштування генератора заготовок коду: щоб брати до уваги не тільки тип проєкту, а і деякі вхідні дані, складені формалізованою природною мовою.

**Практичне значення** отриманих результатів визначається тим, що запропонований метод генерування програмного коду складових вебзастосунків дозволив підвищити ефективність та швидкість написання коду та розробити відповідне програмне забезпечення, придатне до використання у вигляді бібліотеки.

**Апробація роботи.** Основні положення і результати роботи доповідалися та обговорювалися на XIII науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2020 та IX Міжнародній науково-практичній конференції «Проблеми інформатики та комп’ютерної техніки» ПІКТ-2020.

**Структура та обсяг роботи.** Магістерська дисертація складається з вступу, п’яти розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У першому розділі проаналізовано існуючі інструменти для генерування програмного коду, а також розглянуто на проаналізовано методи генерування програмного

ЗМІСТ

[ВСТУП 6](#_Toc108566732)

[1. АНАЛІЗ ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ ГЕНЕРУВАННЯ ПРОГРАМНОГО КОДУ 7](#_Toc108566733)

[1.1. Поняття генеративного програмування 7](#_Toc108566734)

[1.2. Аналіз існуючих інструментів генерування коду 8](#_Toc108566735)

[1.3. Методи та алгоритми генерування програмного коду 17](#_Toc108566736)

[1.4. Висновки до розділу 1 18](#_Toc108566737)

[СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 21](#_Toc108566738)

## ВСТУП

Постійне зростання обсягів розроблюваного програмного забезпечення та попиту на створення нових програм є причиною того, що підвищується рівень вимог до швидкості написання коду та якості отримуваних застосунків. Для того, щоб код програми було легко читати та модифікувати, від розробників вимагають дотримання у роботі принципу модульності, винесення основної функціональності застосунку в окремі утиліти, використання об’єктно-орієнтованого програмування і т.д.

Для підвищення швидкості написання коду у переважній більшості випадків програмісти вдаються до пошуку готових рішень та подальшої модифікації знайденого фрагменту програмного коду з точки зору наявної задачі. Але такий підхід з копіюванням чужого коду містить потенційний ризик того, що в додатку залишаться зайві дії, що не мають відношення до розв’язуваної ним задачі. Іншим використовуваним підходом до пришвидшення створення програмного продукту є генерування структури проєкту із заготовками коду деяких класів, методів та конфігураційних файлів відповідно до обраного типу проєкту.

Звісно, така допомога у написанні коду зменшує час на створення програмного забезпечення та дозволяє розробникам зосереджуватися на більш творчих задачах. Але окрім формування заготовок інфраструктури проєктів в цілому цінною була б автоматизація створення шаблонів окремих програмних компонентів (наприклад, кастомізованих React-компонентів для веб-застосунків).

Крім того, доцільним видається більш тонке налаштування генератора заготовок коду: хотілося б брати до уваги не тільки тип проєкту, а і деякі вхідні дані, складені формалізованою природною мовою. Аналіз існуючих середовищ розроблення програмного забезпечення показав, що на даний момент вони не надають такої функціональності, а, отже, розроблення відповідного методу генерування коду програмних компонентів та його наступна програмна реалізація є актуальною задачею.

## 1. АНАЛІЗ ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ ГЕНЕРУВАННЯ ПРОГРАМНОГО КОДУ

### 1.1. Поняття генеративного програмування

В інформатиці термін автоматичне програмування [1] визначає тип комп’ютерного програмування, при якому якийсь механізм генерує комп’ютерну програму, що дозволяє програмістам писати код на вищому рівні абстракції.

Існує декілька варіантів точного визначення автоматичного програмування, здебільшого тому, що його значення змінювалось з часом. Девід Парнас, простежуючи історію "автоматичного програмування" в опублікованих дослідженнях, зазначив, що в 1940-х роках в ній описувалася автоматизація ручного процесу штампування паперової стрічки. Пізніше мова йшла про переклад мов програмування високого рівня, таких як Fortran та ALGOL. Парнас дійшов висновку, що "автоматичне програмування завжди було скоріше евфемізмом для програмування мовами вищого рівня" [2].

У той же час Мілдред Косс, раніше відомий як програміст UNIVAC, писав наступне: "Написання машинного коду включало кілька нудних етапів – розбиття процесу на дискретні інструкції, присвоєння певного розташування пам'яті всім командам та управління буферами вводу-виводу. Після виконання цих кроків для реалізації математичних підпрограм, бібліотек підпрограм і програм сортування, нашим завданням було розглянути сам процес програмування. Нам потрібно було зрозуміти, як ми можемо повторно використовувати перевірений код і мати машинну допомогу в програмуванні. Під час програмування ми вивчали процес і намагався придумати способи абстрагувати ці кроки, щоб включити їх у мову вищого рівня. Це призвело до розробки інтерпретаторів, асемблерів, компіляторів і генераторів – програм, призначених для роботи з іншими програмами, тобто автоматичного програмування." [3].

Одним з прикладів автоматичного програмування є генеративне програмування або як його можна ще називати – перетворення програм. Генеративне програмування та відповідний термін метапрограмування [4] – це поняття, згідно з якими програми можуть бути написані "для автоматичного створення програмних компонентів" [5]. Тобто він є аналогією автоматизації виробництва традиційних товарів, таких як швейні вироби, автомобілі, хімікати, та електроніки, але вже в контексті створення програмного забезпечення [6].

Метою такого підходу є підвищення продуктивності розробників [7]. Іншими словами – заміна ручного пошуку, адаптації та створення компонентів на автоматичну генерацію необхідних компонентів за вимогами. Також для підвищення продуктивності, якості та часу виходу на ринок при розробленні програмного забезпечення завдяки створенню як стандартних комплектуючих, так і автоматизації виробництва в загальному. Одне з важливих змін парадигми, яке тут мається на увазі, полягає в тому, щоб будувати програмні системи із стандартних компонентів, а не "винаходити колесо" щоразу.

Генеративне програмування означає для розробника додатків те, що він може абстрактно викласти, що хоче отримати в результаті, а генератор створює потрібну систему або компонент. Це працює лише за умови, якщо:

1. Формалізований опис компонента відповідає необхідній структурі.

2. Певні конфігурації визначають, як перевести абстрактні вимоги в конкретні компоненти.

3. Впровадити знання про конфігурацію за допомогою генераторів.

### 1.2. Аналіз існуючих інструментів генерування коду

Генерація коду дозволяє розробнику зосередитися на високому рівні абстракції під час розробки. Він перетворює код високого рівня, написаний 10 людиною, у мову низького рівня. Іншими словами, він генерує вихідний код на основі опису або моделі проєкту. Це дозволяє економити час під час розробки додатків та захищає код від людських помилок, які неминучі навіть для найбільш досвідчених розробників.

Генерування коду під час налаштування збірки дозволяє використовувати результати генерації під час компіляції. Найпопулярнішим інструментом для генерації коду є компілятор. Тобто можна вільно використовувати будь-який компілятор, який вважається ефективним для поставленого завдання. Однак існують спеціальні інструменти для перетворення коду високого рівня в код низького рівня. Ось деякі найкращі інструменти генерації коду під час проєктування:

* розширювані шаблони мовної таблиці стилів (XSLT);
* інструменти на основі UML;
* Razor Generator;
* Metadrone;
* Reegenerator;
* шаблони інструментарію для трансформації текстових шаблонів (Text Template Transformation Toolkit або Т4);
* Radzen;
* генератор CodeSmith;
* ASP.Net Zero.

Всі ці інструменти корисні, коли потрібно створити простий повторюваний код або будь-який текст відповідно до шаблону.

Відомо, що вибір засобу генерації коду залежить від:

* використовуваної мови;
* типу програми або компонента, які потрібно створити;
* середовища розробки, що використовується.

Далі пропонується детальніше розглянути різні інструменти, які можна використовувати для генерації коду.

#### 1.2.1. Шаблони трансформації XSL

eXtensible Stylesheet Language (XSL) – мова стилів для XML- документів. Розширювана мова перекладу таблиць стилів (XSLT) є частиною цієї мови, яка відповідає за перетворення файлів XML в інші формати. Це рішення для створення шаблону є частиною стандарту W3C XSL. Ключовою перевагою XSLT у порівнянні з іншими подібними інструментами є його гнучкість [8].

XSLT призначений для перетворення ієрархічної структури XML- документа в HTML, PDF, текст, вихідний код тощо. Мова XSLT є потужним інструментом для обробки даних та інформації в ієрархічній структурі.

XSLT оптимізований для створення правил перетворення. Він складається з набору правил, позначених тегами шаблону. Правило складається із статичного тексту та ряду тегів, що нагадує конструкції логічних мов: значення відображення, цикл, умова тощо.

На відміну від класичних мов програмування, XSLT описує перетворення не як набір дій, а як набір правил, що застосовуються до вузлів вхідного XML. Кожне правило містить логічну функцію (предикат). Визначити, чи функція відповідає поточному вузлу, можна за допомогою обчислення цієї функції.

У XSLT такі функції описуються мовою XPath. Якщо предикат є істинним, правило виконується.

Модель XSLT включає:

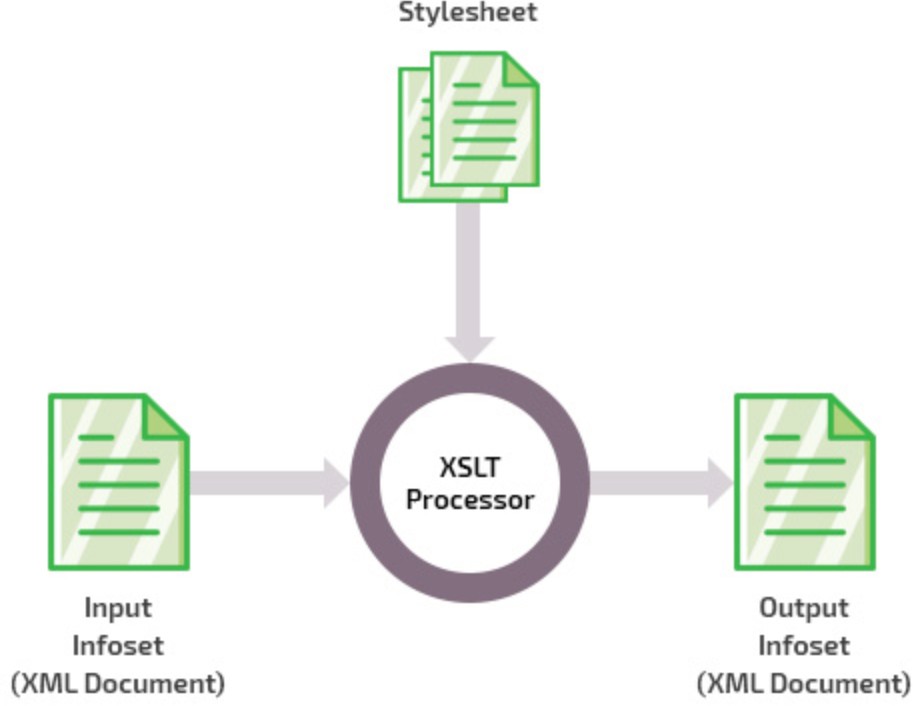
1. XML-документ – вхідні дані для перетворення в інші типи документів.
2. Таблиця стилів XSLT – сформований XML-документ, який містить набір правил перетворення і який використовується як шаблон для перетворення.
3. Процесор XSLT – додаток, який отримує XML-документи та стилі XSLT як вхідні дані та виконує перетворення, застосовуючи правила зі стилів XSLT до XML-документів.
4. Вихідний документ – результат перетворення.

Рис. 1.1. Графічне представлення моделі XSLT

Таким чином метод із використанням XSLT виконує наступні дії:

1. XSLT Processor обробляє таблицю стилів XSLT і застосовує правила перетворення до цільового XML-документу.
2. XSLT Processor генерує відформатований документ у форматі XML, HTML або текстовому форматі.
3. Засіб форматування XSLT генерує фактичне виведення, яке подається на вихід користувачеві.

#### 1.2.2. Інструменти на основі UML

Перетворити код на цільову мову можна за допомогою інструментів на основі моделей уніфікованої мови моделювання (UML) [9]. Вони генерують вихідний код певною мовою з класів UML і дозволяють моделі UML відображати будь-які зміни у вихідному коді.

Двостороння інтеграція допомагає синхронізувати вихідний код та UML. Це означає, що кожного разу, коли ви створюєте фрагмент коду або оновлюєте модель UML, ці зміни об’єднуються.

Генерація коду за допомогою моделей UML вбудована в Microsoft Visual Studio. Він генерує код, написаний на C#, із діаграм класів UML. Це дозволяє зосередитись на бізнес-логіці та архітектурі проєкту, замість того, щоб писати інфраструктурний код низького рівня. Це також допомагає уникнути помилок у коді, які неминуче трапляються під час ручного кодування, і займає багато часу для налагодження.

Для того, щоб генерувати код C# із діаграм класів UML, використовуйте команду «Створити код». За замовчуванням він створює код C# для кожної вибраної діаграми UML. Ви можете змінити або масштабувати цю поведінку, редагуючи або копіюючи текстові шаблони, що генерують код. Ви також можете вибрати будь-яку іншу поведінку для типів, що входять до різних пакетів моделей.

Існують також незалежні інструменти для генерації коду за допомогою моделей UML: UModel, Visual Paradigm, Modeliosoft, Enterprise Architect, erwin Data Modeler тощо. Ці інструменти підтримують генерацію коду на різних мовах програмування, включаючи Java, C++ та Python. Більшість із цих модельних середовищ можна інтегрувати з такими середовищами розробки, як Eclipse, NetBeans, IntelliJ IDEA, Visual Studio та Android Studio.

#### 1.2.3. Razor Generator

Razor Generator (слід не плутати з механізмом Razor) – це інструмент із відкритим кодом, написаний на C#, який підтримує Visual Studio 2019 [10]. Він дозволяє розробнику обробляти файли Razor під час проєктування, а не під час виконання. Завдяки цьому можна інтегрувати файли Razor у свою збірку, щоб полегшити їх повторне використання та розповсюдження.

Цей інструмент надає розробнику більше часу для запуску та дозволяє тестувати подання Razor.

#### 1.2.4. Metadrone

Metadrone – це безкоштовний інструмент, який використовує простий синтаксис шаблону для виведення тексту на основі схеми бази даних [11]. Це дозволяє розробнику зменшити ручне кодування інтерфейсів екранів, зіставлення ORM у фреймворках, збережених процедурах, класах API тощо. Metadrone підтримує бази даних SQL Server, MySQL, Oracle та PostgreSQL.

#### 1.2.5. Reegenerator

Reegenerator – це безкоштовний інструмент генерації коду, інтегрований у Microsoft Visual Studio [12]. Він може використовувати будь- який тип файлу як вхідні дані та генерувати будь-який тип файлу як вихідний файл, наприклад:

* файли даних: XML, JSON;
* файли коду: CS, .VB;
* бази даних.

Reegenerator використовує кілька генераторів в одному файлі. Генератори – це звичайні класи C# / VB.NET у звичайній бібліотеці класів .NET.

#### 1.2.6. Text Template Transformation Toolkit або Т4

Набір інструментів для перетворення текстових шаблонів (T4) – це генератор коду, вбудований у Microsoft Visual Studio з 2008 року [13]. Текстовий шаблон – це елемент T4, який генерує вихідні дані у проєкті Visual Studio для кожної збірки. Логіка шаблону може бути записана на C# або VB.NET. Трансформація шаблонів T4 під час проєктування відбувається під час компіляції. Ці шаблони зазвичай використовуються для генерації коду під час компіляції поточного проєкту. Після генерації T4 надає користувачеві шаблони. Ці шаблони можуть бути використані повторно шляхом успадкування або включення.

Шаблони тексту складаються з:

1. Директиви – Елементи, що керують обробкою шаблонів.
2. Текстові блоки – вміст, який потрібно скопіювати у вихідний файл.
3. Блоки управління – програмний код, який вставляє значення змінних у текст та управляє умовними та повторюваними частинами тексту.

На рис. 1.2. представлено приклад роботи T4, де кроки 1 і 2 показують перетворення команди, а крок 3 – кінцевий результат програми.

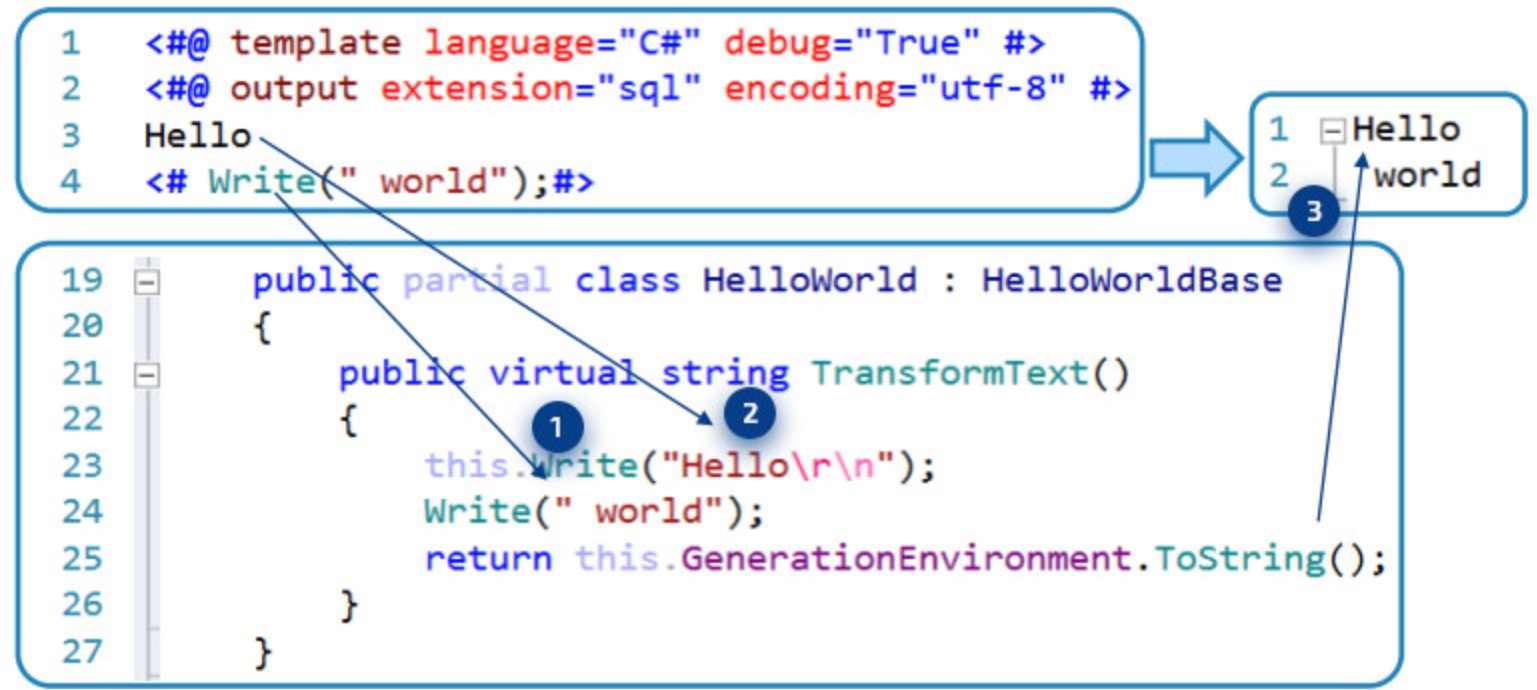


Рис. 1.2. Приклад роботи Т4

#### 1.2.7. Radzen

Radzen – це інструмент для розробки веб-додатків, який генерує код для Angular framework [14]. Ви можете працювати з файлами Radzen у Visual Studio Code. Цей інструмент особливо корисний для невеликих проєктів через обмежену функціональність.

Бекенд Radzen написаний на C#. Коли додається джерело даних з Microsoft SQL Server, MySQL або PostgreSQL, Radzen створює серверну програму ASP.NET Core. Її сторінки представлені за зразком дизайну Model

– View – Controller. Також можна редагувати такі сторінки, використовуючи часткові класи та методи C#.

Radzen підтримує кілька баз даних, таких як:

* MS SQL Server;
* MySQL;
* PostgreSQL;
* OData;
* Swagger.

#### 1.2.8. Генератор CodeSmith

Генератор СodeSmith є частиною інструментів CodeSmith. Це генератор вихідного коду на основі шаблонів. Він автоматизує генерацію коду для будь-якої мови [15]. Генератор CodeSmith містить набір корисних шаблонів, включаючи шаблони для роботи з перевіреними архітектурами (netTiers, SLA, NHibernate, PLINQO, Entity Framework, Kinetic Framework тощо). Є можливість легко змінити шаблон за замовчуванням або створити власний. Коли генерація завершена, CodeSmith Generator надає результати. Генератор CodeSmith підтримує мови C#, Java, VB, PHP, ASP.NET,

SQL та інші. Шаблони можуть генерувати код будь-якою мовою на основі ASCII.

CodeSmith Generator взаємодіє з базами даних за допомогою SchemaExplorer. Цей інструмент надає генератору типи взаємодій із даними сервера SQL або ADO та засоби проєктування для доступу до цих типів із генератора CodeSmith. Крім того, CodeSmith Generator включає багато шаблонів баз даних.

Окрім генератора, набір інструментів CodeSmith включає:

1. Exceptionless – створює звіти про помилки, функції та журнали в режимі реального часу та працює з програмами, написаними в ASP.NET, Web API, WebForms, WPF, Console та MVC.
2. CodeSmith Frameworks – надає розробнику фреймворк PLINQO, набір шаблонів CodeSmith, які генерують скелети об’єктного реляційного картографування за допомогою відповідних шаблонів дизайну. Фреймворки PLINQO включають

#### 1.2.9. ASP.Net Zero

ASP.NET Zero – це рішення для Visual Studio. Воно базується на багатошаровій архітектурі, тоді як його структура коду базується на SOLID. Воно також оснащено інтерфейсом Metronic, який забезпечує зручний інтерфейс [16].

ASP.NET Zero забезпечує високопродуктивну та масштабовану архітектуру та попередньо створені сторінки. Цей генератор має адміністративну панель з усіма функціональними можливостями для адміністрування системи та користувачів, включаючи додавання записів журналів до бази даних. Динамічна локалізація дає змогу запустити ваш проєкт у кількох країнах, а динамічний інтерфейс дозволяє адаптувати його до різних класів користувачів.

ASP.NET Zero надає різні варіанти фреймворку:

1. Рішення ASP.NET Core 2.x та Single-Page Application (SPA) на Angular 7.x.
2. Рішення ASP.NET Core 2.x та ModalViewController (MVC) архітектура на jQuery.
3. Рішення ASP.NET MVC 5.x, веб-API ASP.NET SPA на основі AngularJS 1.x.
4. ASP.NET MVC 5.x, веб-API ASP.NET та рішення на основі jQuery.
5. Мобільний додаток Xamarin, інтегрований із серверним рішенням (лише для версій ASP.NET Core (MVC або Angular UI); підтримує iOS та Android).
6. Додаток на основі ASP.NET MVC.

Шаблони запуску ASP.NET Zero працюють із SQL Server за замовчуванням. Ви можете адаптувати їх для інших сховищ даних вручну. Крім того, якщо використовується інтегрований EntityFramework, можна

адаптувати шаблони MySQL. В офіційній документації є посібник для цього процесу. Якщо ж використовувати Entity Framework Core, можна інтегрувати його з MySQL, PostgreSQL або SQLite. Також можливо інтегрувати його з іншими типами баз даних, але немає вказівок щодо того, як це зробити.

ASP.NET Zero надає нам вихідний код після придбання ліцензії, але існує два винятки:

1. Інструмент містить багато безкоштовних бібліотек із відкритим кодом як компонентів NuGet. Вони не входять до коду, який ви отримуєте після покупки, оскільки це займе занадто багато місця і буде важко оновити ці бібліотеки.
2. У ASP.NET Zero є компонент NuGet із закритим вихідним кодом. Він захищає правила ліцензування.

Документація до кожної версії ASP.NET Zero доступна на офіційному веб-сайті. Він не є обширним порівняно з документацією для інших інструментів, а також відсутні додаткові вебінари, блоги чи відеоуроки. Але є офіційно підтримана громада. Ці функції особливо корисні для розробників, що працюють для малого та середнього бізнесу, які мають обмежений час та ресурси для створення програмного забезпечення. Вони також корисні для розробки додатків SaaS.

### 1.3. Методи та алгоритми генерування програмного коду

На сьогоднішній день існує достатньо велика кількість методів для генерування програмного коду. Автором пропонується розглянути деякі з них для більш точного розуміння даного процесу.

Результатом роботи першого методу є лінеаризоване проміжне подання вихідної програми, або intermediate representation (далі IR). IR складається з послідовності виразів префіксів без дужок. У такому поданні за операторами слідують їх операнди. Кожна інструкція комп'ютера 19 описується префіксом разом із певною "семантичною" інформацією та шаблоном – машинною мовою.

Алгоритм кодування перевіряє збіг шаблонів. Такий процес дуже подібний до синтаксичного аналізу, в якому IR-послідовність виразів префіксів переводиться у послідовність інструкцій, однак існують деякі відмінності від синтаксичного аналізу в деяких аспектах. Оскільки більшість операторів можуть отримати доступ до своїх операндів різними способами, опис цільової машинної програми, як правило, неоднозначний. Дійсно, важливим фактором генерації коду є спосіб вирішення цих неоднозначностей. По-друге, процес генерування коду у такому методі набагато складніший, ніж при синтаксичному аналізі, оскільки він вибирає серед різноманітних інструкцій або послідовностей команд на основі як синтаксичної, так і семантичної інформації.

Таким чином можна представити алгоритм, яким використовує зазначений метод, у такому вигляді:

1. Вхідні дані: функції ACTION та NEXT (представлені у вигляді матриць), опис машинної мови М та лінеаризоване проміжне подання програми Р.
2. Вихідні дані: асемблерна програма для мови P машинною мовою М.
3. Метод: виконання синтаксичного розбору Shift-Reduce вхідної IR-послідовності, видаючи цільові інструкції щоразу, коли виконуються обчислення.

### 1.4. Висновки до розділу 1

У даному розділі було розглянуто поняття генерації коду. Під час проєктування дозволяє розробникам економити час та уникати помилок при написанні коду низького рівня. Крім того, дозволяє використовувати результати генерації коду під час розробки різних додатків.

Також у даному розділі було розглянуто та проаналізовано існуючі інструменти для генерування програмного коду, такі як: XSLT, інструменти на основі UML, Razor Generator, Metadrone, Reegenerator, Text Template Transformation Toolkit, Radzen, генератор CodeSmith та ASP.Net Zero. Були досліджені та проаналізовані підходи до реалізації даного процесу у кожній із зазначених системах, їх структуру та особливості. Деякі з них вбудовані в популярні середовища розробки, такі як Microsoft Visual Studio, а інші – незалежні рішення. Вибір між ними залежить від безлічі параметрів, а саме:

1. Яке середовище, мова та структура використовується?
2. Які функції потрібні?
3. Який бюджет проєкту?

Описані системи генерування програмного коду є достатньо повними у наявних функціональних можливостях, проте такі системи зазвичай або коштують великих грошей і є недоступними для простого користувача сервісу, або ж спрямовані на використання у обмежених середовищах розроблення. При цьому більшість вимагає певної структури файлів, які подаються для опису того, що має бути згенеровано. Також якщо розглядати дані системи як спосіб генерування коду шаблонів компонентів у веб- застосунках, то більшість не дозволить отримати необхідний результат.

Також зазначені існуючі інструменти були проаналізовані за критеріями ресурсоємності, можливості налаштування, орієнтованості на веб-застосунки та універсальність щодо мов програмування. Результати даного аналізу зведені у табл. 1.1.

Таблиця 1.1

Аналіз існуючих рішень

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва рішення | Ресурсоєм- ність | Можливість налашту- вання | Орієнтова- ність на веб- застосунки | Універса- льність |
| XSLT | + | + | + | +/– |
| UML | – | + | +/– | +/– |
| Razor Generator | – | – | – | – |
| Metadrone | + | + | – | – |
| Reegenerator | + | + | – | + |
| Т4 | + | + | – | – |
| Radzen | – | + | + | +/– |
| Генератор CodeSmith | – | + | – | + |
| ASP.Net Zero | + | – | +/– | + |

Бачимо що досить мало рішень спрямовані власне на веб-розробку , а отже це ще раз підтверджує актуальність дисертаційної роботи.

Таким чином, з огляду на вищезазначене, розроблення нового методу генерації коду шаблонів компонентів веб-застосунків з метою пришвидшення створення програмного продукту є актуальною задачею.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Ricardo A. Automatic Inductive Programming [Електронний ресурс] / Aler Mur Ricardo // ICML – Режим доступу до ресурсу: <http://www.sigplan.org/Conferences/GPCE/>– Дата доступу: жовтень 2020.
   * Назва з екрану.
2. Parnas D. L. Software Aspects of Strategic Defense Systems [Електронний ресурс] / Parnas – 1985 November. – Режим доступу до ресурсу: <http://web.stanford.edu/class/cs99r/readings/parnas1.pdf> – Дата доступу: жовтень 2020. – Назва з екрану.
3. Chun. On Software, or the Persistence of Visual Knowledge / Chun, Wendy. – Boston: Grey Room 18, 2004. – 30 с.
4. About Generative Programming [Електронний ресурс] – Режим доступу: https://scala-lms.github.io/tutorials/01\_overview.html – Дата доступу: жовтень 2020. – Назва з екрану.
5. Cointe P. Towards Generative Programming / Cointe., 2005. – 315-325 с. – Дата доступу: жовтень 2020.
6. Generative Programming: Concepts and Experiences (GPCE) [Електронний ресурс] – Режим доступу: <http://www.sigplan.org/Conferences/GPCE/>– Дата доступу: жовтень 2020. – Назва з екрану.
7. Wilcox J., Paying Too Much for Custom Application Development/ James Wilcox – March 2011.
8. Blokdyk G. Extensible Stylesheet Language XSL A Complete Guide Paperback / Gerardus Blokdyk., 2018. – 290 с.
9. Stevens P. Using UML: Software Engineering with Objects and Components

/ P. Stevens, R. Pooley., 1999. – 256 с.

1. Chadwick J. Programming Razor: Tools for Templates in ASP.NET MVC or WebMatrix / J. Chadwick, R. Pooley., 2011. – 120 с.

11. Metadrone [Електронний ресурс] – Режим доступу: http://www.metadrone.com/documentation/ – Дата доступу: жовтень 2020. – Назва з екрану.

12. Child A. The Regenerator / A. Child, R. Pooley., 2015. – 198 с.

13. Hazzard K. Meta programming in .NET / K. Hazzard, J. Bock., 2012. – 360 с.

14. Radzen [Електронний ресурс] – Режим доступу: https://www.apriorit.com/dev-blog/629-web-radzen-app-development- platform-review – Дата доступу: жовтень 2020. – Назва з екрану.

15. CodeSmith [Електронний ресурс] – Режим доступу: https://codesmith.io/ – Дата доступу: жовтень 2019. – Назва з екрану.

16. ASP.NET [Електронний ресурс] – Режим доступу: https://aspnetzero.com/

– Дата доступу: жовтень 2019. – Назва з екрану.

17. Mandic D. Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability / D. Mandic, J. Chambers., 2017. – 102 с.

18. K.S.Tai. Improved semantic representations from tree-structured long short- term memory networks. In Proceedings of the Annual Meeting of the Association for Computational Linguistics / K.S.Tai, R.Socher, C.D.Manning., 2015.

19. S. Karaivanov, V. Raychev, and M. Vechev. Phrase-based statistical translation of programming languages. In Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, pages 173–184. ACM, 2014.

20. T. Nguyen, T. T. Nguyen, and T. N. Nguyen. Lexical statistical machine translation for language migration. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, pages 651–654. ACM, 2013.